Core-Periphery structure analysis for Bitcoin data

Jianqiao Liu, Wen-Chih Li, Parikshit Panwar
Computer Science – Graduate School
Washington State University
Pullman, WA, USA
jianqiao.liu@wsu.edu, wen-chih.li@wsu.edu, parikshit.panwar@wsu.edu

Abstract*—The core-periphery structure is a network theory model that can be used to detect the core nodes and their peripheries. For the detection, we use KM_ config in our project to find the core nodes and their peripheries. Moreover, we implement our method to Erdos-Renyi random graph, Karate Club network, and two Bitcoin datasets. Our goal is to know which Bitcoin trading platform is more trustable. Thus, we come up with a method to measure the trustworthiness of a Bitcoin trading platform based on core-periphery structure.

***Keywords*—core-periphery, Bitcoin, Erdos-Renyi, Karate-Club, KM_config**

I. Introduction

Nowadays, Bitcoin transactions are becoming extremely popular. However, since the Bitcoin transactions are all anonymous, it also brings a lot of fraud risks. Therefore, for new Bitcoin transaction users, choosing a trustworthy Bitcoin trading platform becomes particularly important. However, there is currently no such standard or evaluation to measure the trustworthiness of a Bitcoin trading platform. Thus, in our project, we propose a method to obtain the rating of trustworthiness of the trading platform to help new users find a more reliable Bitcoin trading platform.

In our project, we tested the KM_config core-periphery structure detection algorithm in E-R random graph and Karate Club Network. For these experiments, we would like to see if this algorithm is suitable for some popular datasets. We found that KM_config is not suitable for E-R random graph but works well on Karate-Club network.

Also, we came up with a rating based on the core-periphery structure to find whether the Bitcoin trading platform that has a trust-weighted signed network is trustworthy. We used two Bitcoin trust-weighted signed networks, including Bitcoin OTC and Bitcoin Alpha. We applied the KM_config algorithm on these two Bitcoin network to detect core-periphery structures and find core nodes. Then, we generated two ratings representing the trustworthiness of Bitcoin OTC and Bitcoin Alpha respectively based on the weights involving the core nodes. The higher the rating, the more trustworthy the Bitcoin trading platform. For the test results, we found that Bitcoin Alpha is more trustworthy than Bitcoin OTC. Furthermore, applying this method on Bitcoin trust weighted signed networks, we can know which trading platform is more suitable for new users to participant in Bitcoin transactions.

II. Problem Definition

Bitcoin is a cryptocurrency that features a distributed, decentralized and trustworthy mechanism, which has made Bitcoin a popular global transaction platform [1]. However, due to the anonymity of Bitcoin trading platform users, Bitcoin transactions have generated a lot of fraudulent behaviors. Thus, it is important to choose a trustworthy trading platform to conduct Bitcoin transactions.

Core-periphery structures can help in the detection of anomalies in transactions. A core-periphery structure could have a core node and its peripheries, and even multiple core nodes. In Bitcoin transactions, we would like to come up with a method that can generate a rating to measure the trustworthiness of the transaction platforms. If the rating can measure out the more trustworthy platform, the users can choose the transaction platform based on the ratings. However, there is currently no measurement or standard to tell which bitcoin transaction platform is trustable. Thus, we want to conduct our project to obtain the measurement

III. Algorithms

In this section, we will discuss some concepts and algorithms used in our project.

A. Core-Periphery Structure

Core-periphery structure is a structure within a network containing core nodes and the periphery nodes. The core nodes are the nodes that are densely interconnected, whereas the periphery nodes are the nodes that are not interconnected but densely connected to the core nodes. Core-periphery structures can be used to understand some properties of the network that cannot be interpreted by studying the individual nodes and edges, such as using centrality measures. Some of the known networks in which these core-periphery structures are found in are neural network, protein-protein interaction network, social network, transportation network, etc. [2].

B. Core-Periphery Structure Detection Algorithm

In this subsection, we shall discuss the algorithm that is used for finding the core-periphery structure in the network. According to the previous subsection, a network consists of a core-periphery structure, however, there may be multiple such core-periphery pairs in a network (as shown in Fig. 1 below, each color represents a core-periphery structure), each pair consisting a group of core node and a periphery node.

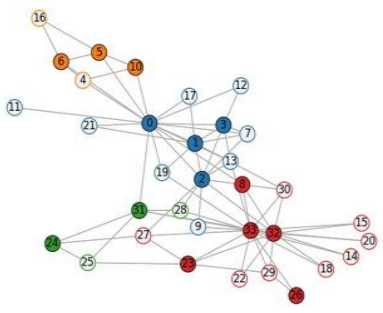

Fig. 1. Multiple core-periphery pairs in a network.

The algorithm, KM-Config, which is from the Python package cpnet, is used in this project to find such multiple core-periphery pairs in a network. This algorithm provides functionalities, get_pair_id() and get_coreness(), to obtain the Id's of each pair of core-periphery structure in a network and get the coreness value of each node respectively [2]. The coreness value is a node denotes or signifies whether a node in a network belongs to the core node or the periphery node. The coreness value is a floating value between 0 to 1. If the coreness value of a node is higher, it signifies that the node belongs closer to the core than the periphery [2].

The KM_config algorithm is known to provide better results than previous algorithm developed by Borgatti and Everett [1]. The Borgatti-Everett algorithm was based on an approach that the nodes that have higher degree will form the core and those of lower degree will form the periphery, which is not always the case [2]. Therefore, this algorithm proposed that a network composed of a single core-periphery structure where all the nodes in the network were partitioned into the core and periphery nodes.

IV. Data

In this section, we are going to introduce the networks and datasets we used in our project, including Erdos-Renyi random graph, Karate-Club network and two Bitcoin datasets.

A. Erdos-Renyi Random Graph

We use the Erdos-Renyi random graph with n = 80 nodes, p = 0.1 probability for edge creation in our project. It is a random graph that comes from python packages named "NetworkX". We use it to generate an E-R random graph and implement an algorithm into it.

B. Karate-Club Network

A social network of a karate club was studied by Wayne W. Zachary for a period of three years from 1970 to 1972 [3]. The network use nodes = 34 and edge = 48 members of a karate club, documenting links between pairs of members who interacted outside the club.

C. Bitcoin Datasets

In our project, we used two Bitcoin datasets from SNAP Stanford Large Network Dataset Collection [4]. These two Bitcoin datasets are from two Bitcoin trading platforms called Bitcoin OTC and Bitcoin Alpha. These are who-trusts-whom networks.

Since the users in the Bitcoin trading platform are anonymous, in order to prevent transactions with fraudulent and risky users, users of Bitcoin trading platform will rate other users in a scale of -10 (distrust) to +10 (trust). Thus, a trust weighted signed network is formed. Two dataset statistics can be shown in table 1. The node denotes the platform user, the edge denotes the rating direction, and the weight denotes the rating.

TABLE I. Dataset Statistics

Dataset name	Nodes	Edges	Range of edge weight
Bitcoin OTC	5,881	35,592	-10 to +10
Bitcoin Alpha	3,783	24,186	-10 to +10

	A	B	C	D
1	6	2	4	1.29E+09
2	6	5	2	1.29E+09
3	1	15	1	1.29E+09
4	4	3	7	1.29E+09
5	13	16	8	1.29E+09
6	13	10	8	1.29E+09
7	7	5	1	1.29E+09
8	2	21	5	1.29E+09
9	2	20	5	1.29E+09
10	21	2	5	1.29E+09
11	21	1	8	1.29E+09
12	21	10	8	1.29E+09
13	21	8	9	1.29E+09
14	21	3	7	1.29E+09
15	17	3	5	1.29E+09

Fig. 2. Bitcoin Dataset Format

As shown in the Fig 1, the data format of the dataset is that each line has one rating, and the format is as follows:

SOURCE, TARGET, RATING, TIME

Where SOURCE denotes the node id of source (i.e., rater), TARGET denotes the node id of target (i.e., ratee), RATING denotes the source's rating for the target, ranging from -10 to +10, TIME is the timing of the rating, measured as seconds since Epoch.

V. Implementation

In this section, we are going to show our implementation of using KM_config core-periphery structure detection algorithm on three different networks: Erdos-Renyi Random Graph, Karate-Club network, and Bitcoin datasets. The experimental setup will mainly use Python with VS Code as the code editor. The python packages we used are cpnet, NetworkX, Numpy, Pandas, and Matplotlib in our implementation.

A. Erdos-Renyi Random Graph

We applied the KM_config algorithm using E-R random graph to see if the result is clear and can also be used for the analysis of bitcoin datasets. In the E-R random graph, we used n=80 (n represent nodes) and p=0.1 (p represent probability for edge creation) to generate the graph. Furthermore, we used the function "erdos_renyi_graph" from NetworkX to generate the graph [5].

B. Karate-Club Network

We also applied the KM_config to the Karate-Club network. Eventually, Fig 3 shows that we found four core clusters and their peripheries with different colors. As a result, we could clearly look at the core cluster and its peripheries so that we could conclude that the KM_config is suitable for the Karate club network. This is a method that can be trusted in finding multiple core nodes and their peripheries. Therefore, we could apply this method to analyze bitcoin transaction datasets.

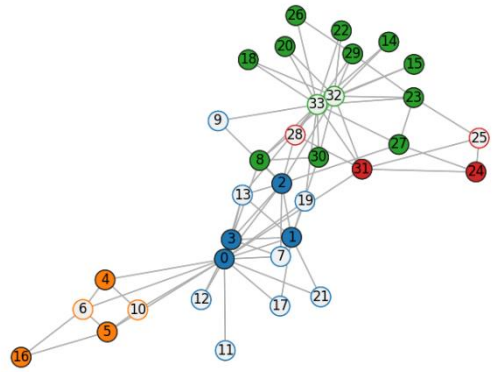

Fig. 3. Karate Club – KM_config

C. Bitcoin Dataset

In order to find out which Bitcoin dataset is more trustworthy, we need to do these steps to get the results.

1) Find core users of the Bitcoin trading platform

In order to evluate the trustworthiness, finding the core users of the Bitcoin trading platform is very important. Since these core users use the platform more often, they will conduct more Bitcoin transactions with other users, so the number of trust ratings involving these core users will be greater. Thus, these ratings involving core users can be used as a measure of the trustworthiness of the Bitcoin trading platform.

2) Find core nodes of the Bitcoin transaction network

Since we know that we need to find the core users of the Bitcoin, but how can we make it? We can find the core nodes of the Bitcoin transaction network with KM_config core-periphery detection method, and these core nodes can be considered as core users of the Bitcoin platform.

However, how can we know that the core nodes are the core users? We can do a series of calculation to prove it. Let us take the Bitcoin OTC platform for an example. OTC network has 5,881 nodes and 35,592 edges.

a) First, apply KM_config algorithm to OTC network, so we can know which are the core nodes and which are the periphery nodes.

b) Then, traverse the result and obtain the number of core nodes, N_C and the number of periphery nodes N_P. In this example, $N_C = 2051$ and N_P is 3830.

c) Next, traverse all edges of the network and get the number of edges involving the core nodes E_C. In this example, $E_C = 26895$, so the number of edges do not involve core nodes is $E_P = 8697$

d) Thus, for each core node, the average number of edges involving the core node is $\frac{E_C}{N_C} \approx 13.11$, but for each periphery node, the average number of edges is $\frac{E_P}{N_P} \approx 2.27$. Since each node denotes a user and each edge denotes one Bitcoin transaction rating, we can see that the core nodes have much more edges than periphery nodes on average. In other words, the average number of Bitcoin transactions for users denoted by the core nodes is far more than the average number of Bitcoin transactions for users denoted by peripheral nodes.

In conclusion, the result is in line with the assumption. Thus, we can get a list of core nodes which represent a list of core users.

3) Calculate average rating involving core users

Since we have obtained the list of core nodes representing core users, we can traverse the network and get a list of edges involving these core nodes. Because we know that each edge has a weight, i.e. rating, we can calculate the summation of the weights of the edges involving the core nodes (i.e., W_c), and then divide by the number of these edges (i.e., $\mathrm{E_C}$) to get an average weight (i.e., W_{avg}). That is,

$$W_{avg} = \frac{W_c}{\mathrm{E_C}}$$

Thus, we can get the average rating involving core users which is equal to W_{avg}.

4) Compare the average ratings of two Bitcoin datasets

As mentioned in 1), ratings involving core users can be as a measure of the trustworthiness of the Bitcoin trading platform. Therefore, we can compare the size of the two average ratings to determine which Bitcoin network is more trustworthy. The larger the W_{avg}, the higher the average rating, the more trustworthy the Bitcoin trading platform.

VI. Results and Discussion

In this section, we introduce the results of applying KM_config on Erdos-Renyi random graph, Karate-Club Network and Bitcoin datasets and some discussion about our method.

A. Erdos-Renyi Random Graph

Fig 3 shows the result of the E-R random graph. We can easily view that the classification distribution of the core-periphery pairs is not clear enough for the user to identify. A considerable amount of nodes that belong to one color is not gathered as a real cluster. Even if they are gathering as a real cluster, we cannot view it properly through Fig 4.

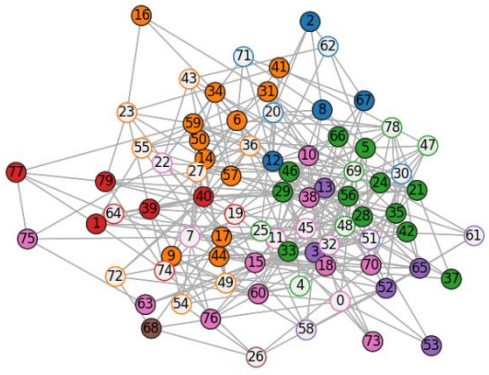

Fig. 4. Erdos-Renyi Random Graph – KM_config

B. *Karate-Club Network*

Fig 5 shows that the result of pair id of the Karate Club Network using KM_config. Take node 0 for example, node 0 is a core node so that the coreness is 1 and it has the share the same pair with nodes (1,2,3,7,9,11,12,13).

Fig 6 shows that the result of the Karate Club Network. We use four different colors to represent different core nodes and their peripheries. Moreover, the result shows that we have successfully identified four cluster and their peripheries within four colors.

We can simply take core-periphery pair 1 as an example. Pair 1 contains nodes (4, 5, 6, 10, 16) as shown in Fig 6. Then, we look at Fig 5 you can view that all the nodes in pair 1 are share with the same color orange. Different pairs clearly share different colors. And the coreness of nodes 4, 5 and 16 are all 1 which means that they are core nodes, while in Fig 5, node 4, 5, 6 are in deep color which denote they are the core nodes.

We can conclude that compare with E-R random graph, Karate Club Network has clearer cluster and its peripheries. It proves the KM_config is more suitable in Karate Club Network.

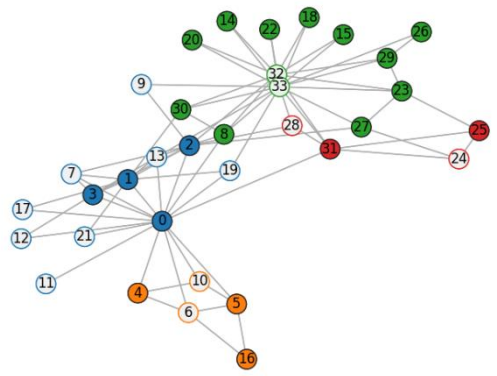

Fig. 5. Karate-Club Network – KM_config

	A	B	C
1	Node ID	Pair ID	Coreness
2	0	0	1
3	1	0	1
4	2	0	1
5	3	0	1
6	4	1	1
7	5	1	1
8	6	1	0
9	7	0	0
10	8	2	1
11	9	0	0
12	10	1	0
13	11	0	0
14	12	0	0
15	13	0	0
16	14	2	1
17	15	2	1
18	16	1	1

Fig. 6. Karate-Club pair_ids

C. *Bitcoin Dataset*

1) *Get core nodes*

First step is to import the Bitcoin network, we used the Python library pandas to complete this job.

Next, we applied the KM_config algorithm to detect the core-periphery structures of the network. However, during the implementation, we found that the algorithm has randomness, it means that the algorithm will provide different results in different runs. Fortunately, the algorithm developer provides a parameter num_runs to KM_config to customize the number of runs of the algorithm, which can reduce the randomness of the algorithm and obtain a result with less error. So, we increased the number of runs by passing "num_runs" parameter by “KM_config(num_runs = 50)”.

Then, we retrieved the results through the function “get_pair_id()” to get the core-periphery structure pair ID, and the function “get_coreness()” to get the coreness of the node. We integrated the result and export to a csv file. So, we got the pair ID and coreness of all 5,881 nodes in Bitcoin OTC network, and the pair ID and coreness of all 3,783 nodes in Bitcoin Alpha network.

The core-periphery structure detection results of two Bitcoin datasets are as follows (since the output is random, we picked one of the results to show). After that, we can traverse the results and get the list of core nodes.

	A	B	C
1	Node ID	Pair ID	Coreness
2	6	106	1
3	2	106	1
4	5	106	0
5	1	106	1
6	15	106	1
7	4	106	1
8	3	106	1
9	13	106	1
10	16	106	0
11	10	106	1
12	7	106	1
13	21	106	1
14	20	106	1
15	8	106	0

5873	5996	158	0
5874	5992	0	0
5875	5997	0	0
5876	5998	0	0
5877	5999	113	0
5878	6000	161	0
5879	6002	161	1
5880	6003	106	1
5881	6004	0	0
5882	6005	0	0
5883			

Fig. 7. Bitcoi OTC Core-Periphery Detection Results

	A	B	C
1	Node ID	Pair ID	Coreness
2	7188	7	0
3	1	7	1
4	430	7	1
5	3134	7	0
6	3026	7	0
7	3010	7	0
8	804	7	0
9	160	7	1
10	95	7	1
11	377	0	1
12	888	7	1
13	89	7	1
14	1901	7	0
15	161	7	1
16	256	7	1

3773	3314	7	1
3774	3394	7	1
3775	5029	7	0
3776	7405	7	0
3777	7407	7	0
3778	7430	7	1
3779	5837	87	1
3780	7465	87	0
3781	7363	7	1
3782	7376	7	1
3783	7461	7	0
3784	7466	7	1
3785			

Fig. 8. Bitcoi Alpha Core-Periphery Detection Results

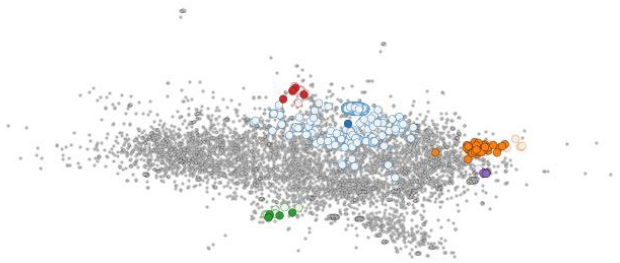

Fig. 9. Bitcoi OTC Core-Periphery

In addition, as shown in Fig 9, we also visualized the core-periphery structure of the Bitcoin dataset based on NetworkX. However, since the number of nodes in the dataset is quite huge, the visualization looks messed up, and there are not enough colors to show the different pairs, because there may be hundreds of pairs. Thus, the visualization of Bitcoin dataset is not applicable.

2) Calculate average weight involving core nodes

Since we obtained the two lists of core nodes, we traversed the original Bitcoin dataset and get the list of edges whose source or target node is core node. However, when we wanted to calculate the average weight of these edges, we cannot directly get the weight of the edges because each edge does not have weight attribute. Therefore, we did one more step to convert the format of the original Bitcoin dataset. For each edge, we granted the rating of the edge to the weight attribute. We used the function "add_weighted_edges_from" provided by NetworkX to implement this step, then we obtained the new dataset format:

	A	B	C
1	source	target	weight
2	6	2	{'weight': 4}
3	6	5	{'weight': 2}
4	6	4	{'weight': 2}
5	6	7	{'weight': 5}
6	6	114	{'weight': 2}
7	6	32	{'weight': 1}
8	6	173	{'weight': 3}
9	6	258	{'weight': 1}
10	6	268	{'weight': 3}
11	6	219	{'weight': 4}
12	6	198	{'weight': 2}
13	6	35	{'weight': 4}
14	6	664	{'weight': 1}
15	6	937	{'weight': 1}

Fig. 10. Transformed Bitcoin dataset

Then, we summed up the weights of edges whose source or target node is core node and divided by the number of edges involving core nodes, so we obtained an average rating representing the trustworthiness of the Bitcoin network. We did this process for 5 times and calculate the average to reduce error, and the results are as follows:

TABLE II. BITCOIN OTC RESULTS

Test	**Bitcoin OTC**			
	Core nodes number	***Total number of edges involving core nodes***	***Total weight of edges involving core nodes***	***Average weight of edge involving core nodes***
1	1839	26517	21867	0.8246
2	2292	25788	21535	0.8351
3	1872	26184	24838	0.9486
4	1907	26106	24448	0.9365
5	1821	26315	22157	0.8420
Average	1946.2	26182	22969	**0.8774**

TABLE III. BITCOIN ALPHA RESULTS

Test	**Bitcoin Alpha**			
	Core nodes number	***Total number of edges involving core nodes***	***Total weight of edges involving core nodes***	***Average weight of edge involving core nodes***
1	1353	17988	24763	1.3766
2	1312	17869	24950	1.3963
3	1285	17728	24995	1.4099
4	1344	17562	24562	1.3986
5	1392	18104	25524	1.4099
Average	1337.2	17850.2	24958.8	**1.3983**

As we can see, the average weight of edge involving core nodes of Bitcoin Alpha is larger than Bitcoin OTC. As we mentioned in implementation section, the larger average weight, the more trustworthy the Bitcoin trading platform. Since the average weight of edge involving core nodes of OTC platform is 0.8774, and the average weight of edge involving core nodes of Alpha platform is 1.3983. Thus, we can tell that the Bitcoin Alpha trading platform is more trustworthy.

D. Discussion

In this part, we introduce a method to find out which Bitcoin trading platform is more trustworthy based on comparing the value of the average weight of edge involving core nodes. This could be a reference for new users to choose a more credible Bitcoin trading platform.

However, there are still some weaknesses in our method. For instance, average weight of edge involving core nodes the two Bitcoin transaction networks we choose is quite different, so it is easy to compare the size. But the core-periphery structure detection algorithm has randomness, so the results are not stable. What if the average weights of two networks are roughly same or the two values vary within an interval? It might be hard to measure the trustworthiness of the two Bitcoin trading platforms. Thus, it may take more calculations to get a more accurate value.

VII. RELATED WORK

According to the article "Models of core periphery structures" [6], it clearly introduced the ideal image of the core-periphery structure. Furthermore, this paper mentions the core-periphery structure are used in economic and organization studies. In our project, we going to use the core-periphery structure to detect the core nodes and their peripheries. After that, we will generate the score in order to

measure the trading platform.

As described in the earlier section, some of the earlier work on core-periphery structure was done by Borgatti and Everett who were the first to study the core-periphery structure in a network. The Borgatti-Everett algorithm followed the approach that the network itself is a core-periphery structure consisting of the core nodes and the periphery nodes where core nodes are nodes with higher degree and the periphery nodes are nodes with lower degree. In comparison with the algorithm that we have used in this project, the KM_config algorithm is able to find multiple core-periphery pairs in a network.

Other related work, as described in [7], consists of implementing an algorithm, ClusterONE-CP. This algorithm is used to find the core-periphery pairs of a weighted network based on a greedy approach. Furthermore, this algorithm scores the core-periphery structures and divides it into types: Type 1 and Type 2, based on "how many standard deviations away is the mean of periphery edges from the core." [7]. However, the KM_config algorithm is only able to find multiple core-periphery pairs but does not further classify the core-periphery pairs.

VIII. Conclusion

Generally, we apply the core-periphery detection algorithm KM_config on three different networks, including Erdos-Renyi random graph, Karate-Club network and Bitcoin transaction networks. After the implementation, we found that KM_config algorithm is not suitable for Erdos-Renyi random graph but appropriate for Karate-Club network. Next, we applied KM_config algorithm on two Bitcoin transaction networks from Bitcoin OTC trading platform and Bitcoin Alpha trading platform. For each network, we traversed the result and obtained the list of core nodes. Then, we got the list of edges involving these core nodes and calculated the average weight which could represent the rating of the trustworthiness of the network. By comparing the ratings of two network, we found that Bitcoin Alpha trading platform is more trustworthy.

For the future work, since what we have done in this paper is to find a trustworthy Bitcoin trading platform, we also want to focus on a single platform. We wonder if it is possible to judge the trustworthiness of a group users or a single user in the platform through core-periphery structures. For instance, if the administers of the platform know the trustworthiness of users, they could grant more authorities to users with high trustworthiness or reduce the authorities of users with low trustworthiness. It might reduce the cost of maintaining the platform.

Bibliography

[1] Y.-J. Lin, P.-W. Wu, C.-H. Hsu, I.-P. Tu, and S.-W. Liao, "An evaluation of bitcoin address classification based on transaction history summarization," in 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2019, pp. 302–310.

[2] S. Kojaku, N. Masuda. "Core-periphery structure requires something else in the network". 2018.

[3] Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. Journal of Anthropological Research, 33(4), 452–473.

[4] S. Kumar, F. Spezzano, V.S. Subrahmanian, C. Faloutsos. Edge Weight Prediction in Weighted Signed Networks. IEEE International Conference on Data Mining (ICDM), 2016.

[5] Networkx.generators.random_graphs.erdos_renyi_graph — NetworkX 2.5 documentation. (n.d.). Retrieved May 2, 2021, from Networkx.org website:https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html.

[6] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," Soc. Networks, vol. 21, no. 4, pp. 375–395, 2000.

[7] D. Sardana and R. Bhatnagar, "Core periphery structures in weighted graphs using greedy growth," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2016, pp. 1–8

Appendix

GitHub project: https://github.com/Jianqiao-WSU/Core-periphery-Structures