Core-Periphery Structure Analysis and Visualization

Cpt_S 591 Project Final Report

Xinyi Wang, Yu-Chieh Wang, Jingyuan Huang
11653817, 11641327, 11644540
Computer Science – Graduate School
Washington State University
Pullman, WA, USA
xinyi.wang@wsu.edu, yu-chieh.wang@wsu.edu, jingyuan.huang@wsu.edu

Abstract*—The core-Periphery structure is able to help understand some property which cannot be found by only studying nodes and their neighbors locally in a network. Also, it identifies the relationships between cohesive core clusters surrounded by sparse peripheries. In this paper, we will simply explain what is Core-Periphery structure. Then, using two methods on three different datasets(random, karate-club, and bitcoin trust network) to generate graphs. Finally, we will analyze the result graph in detail.

***Index Terms*—Core-Periphery, Network Science, Graph**

I. Introduction

Today, most network algorithms focus on core blocks, and few algorithms discuss the intersection of two cores. For example, here are three articles X, Y, and Z. X is related to medical topics. Y is related to the subject of the project. Z is also related to medical and engineering topics. Here, the relationship between Z and medical and engineering is not as strong as X and Y. This has a relatively weak connection with more than two topics at the same time, we call it periphery.

Going back to the aforementioned algorithm, if the general algorithm is used on the article Z, it does not belong to any type of topic, so it is easy to be ignored or randomly assigned during analysis. However, article Z is as important as X and Y, and should not be ignored because it cannot be classified. Therefore, the algorithms based on core-periphery structure play a considerable role in the analysis of such problems. They can not only help to understand some property which cannot be found by only studying nodes and their neighbors locally in a network, but also identify the relationships between cohesive core clusters surrounded by sparse peripheries.

In this project, we use three different size but the same structure databases to help analysis. We hope to start with the smallest database and use the simplest graphics to explain in detail the differences between the algorithms based on the core-periphery structure and the advantages over traditional algorithms. Next, we will use more complex databases step by step to slowly guide everyone to understand the real-world network model-Bitcoin's trust network. Through the previous explanation, it can not only help to understand more easily but also effectively analyze the network.

It is worth mentioning here that in the following experiments we will use two algorithms KM_ER and KM_config. Moreover, the algorithm of KM_ER is applied to the random graph, because this algorithm and this graph are bound to each other. Only this algorithm can efficiently calculate random graphs. Also, we use the KM_config algorithm for the karate and bitcoin databases. Finally, we will analyze the results of these three databases.

II. Problem definition

Core-periphery structures help in detection of anomalies in transactions, or observe network-level transaction trends in crypto-currencies.

It is important to maintain a who-trusts-whom network for blockchain based technologies such as bitcoins. We are able to use the result of the core-periphery structure algorithm to identify different clusters of users in order to apply policy or measures to manage the transaction network, like giving the clusters of people with high levels of trust more authority, or impose stricter regulations on clusters of people with low levels of trust.

III. Data

How do we collect the data? We form a dataset like this. For each tuple in the dataset, it contains three attributes which are sources, target, and weight. For example, this figure is about the dataset of bitcoin transactions. In the figure 1, source and target represent the buyer and seller, then the weight means the rating of this transaction. It has a fixed range so that buyers can choose whatever points to give back to the seller.

	A	B	C
1	source	target	weight
2	user6	user2	4
3	user6	user5	2
4	user1	user15	1
5	user4	user3	7
6	user13	user16	8
7	user13	user10	8
8	user7	user5	1
9	user2	user21	5
10	user2	user20	5

Figure 1 The Dataset of Bitcoin Network

A. Datasets

1) karate-club network:

Social network of friendships between 34 members of a karate club at a US university in the 1970s. which is node=34, edge=78.

2) Erdos-Renyi random graph with n=80, p=0.1:

It is a database that comes with a python package called networkx. We use use it to generate a ER random graph.

3) Bitcoin Transaction Network:

The dataset is originally from SNAP Stanford Large Network Dataset Collection. The format of the dataset is csv, and the details of it are in Figure 2 .

Dataset statistics	
Nodes	5,881
Edges	35,592
Range of edge weight	-10 to +10
Percentage of positive edges	89%

Figure 2 Information of Bitcoin Network

B. Tools

1) Program Language: Python

In order to get more accurate algorithm results and generate graphics, we use a language that is easier to interface with other tools: python

2) Python package: cpalgorithm

cpalgorithm is a python tool that specifically implements the Core-Periphery algorithm. It contains a variety of algorithms for one-way indicators and two-way indicators, such as BE, MINRES, SBM, LowRankCore, LapCore, LapSgnCore, Surprise, Divisive, KM_ER, and KM_config.

C. Methods

In this project, we mainly use two Core-Periphery algorithms to do analysis: KM_ER, and KM_config.

1) KM_ER

This algorithm finds multiple core-periphery pairs in networks. In the detection of core-periphery pairs, the Erdos-Renyi random graph is used as the null model.

2) KM_config

This algorithm finds multiple core-periphery pairs in networks. In the detection of core-periphery pairs, the configuration model is used as the null model.

IV. Algorithms

Now we come to the methodology section. The overall methodology is based on a greedy growth algorithm. To identify each core-periphery pair, we use edge weight density in the form of the mean of edges to build peripheries around cores. The cohesion score for a cluster C is defined as this:

$$Cohesion(C) = \left(\frac{\sum(w_{in}(C))}{\sum(w_{in}(C)) + \sum(w_{out}(C)) + p \times |C|} \right)$$

Figure 3 Cohesion Score

Here W-in denotes the internal edge weights ∈ C and W-out denotes the weight of outgoing edges from C. p ∗ |C| is a penalty term used to model the existence of yet undiscovered edges. Further, a user defined parameter called MEAN_ OFFSET is defined to ascertain the difference in edge weight density of prospective core clusters and their surrounding periphery clusters. A value of 0.2 is used as a default value for MEAN_OFFSET. The algorithm operates in four phases as described next.

A. *Initialization Phase*

The initialization phase begins with ordering the nodes of the graph for cluster initiation in the greedy growth phase. The nodes of the graph are ranked according to their weighted degree. This order is defined as the cluster initiator order.

B. Greedy Growth Phase

In the greedy growth phase, cluster initiators not previously

assigned a cluster label are chosen as seeds for cluster growth. The seed nodes are allowed to grow their cluster as long as the structural density measure, Cohesion of the cluster keeps on increasing. The growth process checks for node addition or deletion from the growing cluster, whichever leads to a greater increase in cohesion. A growing seed stops further growth when a locally optimal cohesive group, say Li has been formed. At this point, Li is used to extract its boundary nodesets. The growth phase uses boundary nodesets to facilitate the growth of prospective periphery clusters in the boundary of relatively dense prospective core clusters. Next we describe the extraction of boundary nodesets and how they are used in the growth phase in the next two subsections.

C. *Overlap Phase*

The locally optimal cohesive overlapping clusters formed in the growth phase might have a high overlap, such that merging them will avoid duplicate information. Thus, the merge phase merges highly overlapping pairs of clusters from the growth phase with an overlap score greater than an overlap threshold (= 0.5). The overlap score between two clusters Ci and Cj is calculated as follow:

$$OverlapScore(C_i, C_j) = \frac{|C_i \cap C_j|^2}{|C_i| \times |C_j|}$$

D. *Extraction of Core Periphery associations*

In the growth phase, cluster initiators are chosen as per their weighted degree. This naturally leads to the formation of structurally denser clusters before the relatively sparser clusters. Therefore, clusters which are formed earlier in the growth phase have a greater chance to become cores than clusters which are formed later in the growth phase. We exploit this property to look for core periphery relationships in the natural cluster formation order. All clusters, Oi, are traversed in the order in which they were formed to check for their boundary nodesets inrange and low. It is to be noted that by now, each node in these boundary nodesets should also have been assigned a cluster label, say, Pj , in the growth phase. Core periphery relationships are made between Oi and each cluster Pj , if mean(W-in(Oi)) is greater than mean(win(Pj)) for low boundary clusters, and cohesion(Oi) is greater than
cohesion(Pj) for inrange boundary clusters.

Type of Core periphery relationship: Each core periphery relationship (Oi, Pj) is given a type (1 or 2) based upon the majority nodeset type in the periphery cluster. Type 1 peripheries have majority of nodes belonging to nodeset inrange, while type 2 peripheries have majority of the nodes belonging to nodeset low.

V. Implementation

In this section, we use the algorithm to implement on three different datasets: karate club network, Erdos-Renyi random graph and bitcoin transaction network. For the API commands, we use KM-config for karate club network and bitcoin transaction network, and KM-ER for the Erdos-Renyi random graph. The results will include the table of relation between nodes and their cp-pair plus the coreness attribute, and the plots of network.

The experimental setup we are using is Python with specific packages named of matplotlib, pandas, cpalgorithm, networkx.

For better visualization, we use the result of karate club network as the experimental evaluation in Figure 3 and 4.

From above result, we successfully identify four core clusters and their peripheries with colors. The Figure 3 lists the node ID, cp-pair and coreness.

We simply take a look at the cp-pair 0. Since the table has sorted the nodes by the sequence of cp-pair, all nodes from cp-pair 0 are going to be seen from the table. We can see a set of nodes(0, 1, 2, 3, 7, 9, 11, 12, 13, 17, 19, 21) are all from cp-pair 0, and from the plot of the dataset, those set of nodes are all represented into the same red color.

Moreover, the coreness of the nodes from the table is the same from the plot: nodes(0, 1, 9) are all from cp-pair 0 periphery, and the rest of the nodes are all from cp-pair 0 core. It should be noted that we're using the same sort order of the cp-pairs as the sort order of the primary colors which include only 7 colors, so the cp-pair 0 here corresponds to red color.

For the other datasets we mentioned before, the implement and result are all same as this one.

	Name	PairID	Coreness
2	0	0	0.000000
3	1	0	0.000000
4	2	0	1.000000
5	3	0	1.000000
6	7	0	1.000000
7	9	0	0.000000
8	11	0	1.000000
9	12	0	1.000000
10	13	0	1.000000
11	17	0	1.000000
12	19	0	1.000000
13	21	0	1.000000
14	4	1	0.000000
15	5	1	0.000000
16	6	1	1.000000
17	10	1	1.000000
18	16	1	1.000000
19	8	2	0.000000
20	14	2	0.000000

Figure 3 Relation between Nodes and cp-pairs of Karate Club Network

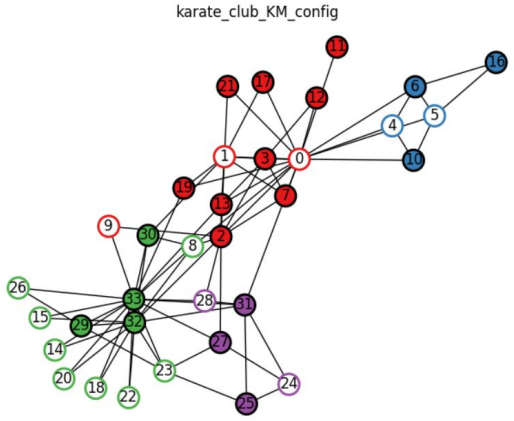

Figure 4 Relation between Nodes and cp-pairs of Karate Club Network

VI. Results and discussion

A. *Karate-club network*

For the Karate-club network, Figure 5 is the relation table between nodes and c-p pairs and the attribute of nodes(the c-p is core-periphery). Pair 0 means the node belongs to the No.0 c-p pair and the coreness means whether the node is in core or not.

	Name	PairID	Coreness
2	0	0	0.000000
3	1	0	0.000000
4	2	0	1.000000
5	3	0	1.000000
6	7	0	1.000000
7	9	0	0.000000
8	11	0	1.000000
9	12	0	1.000000
10	13	0	1.000000
11	17	0	1.000000
12	19	0	1.000000
13	21	0	1.000000
14	4	1	0.000000
15	5	1	0.000000
16	6	1	1.000000
17	10	1	1.000000
18	16	1	1.000000
19	8	2	0.000000
20	14	2	0.000000

Figure 5 Distribution Table of Karate-club network

Figure 6 shows that it is separated into four colors depending on the winning rate by using KM_config. The red core cluster means the players who have a high winning rate, the color purple is the opposite:

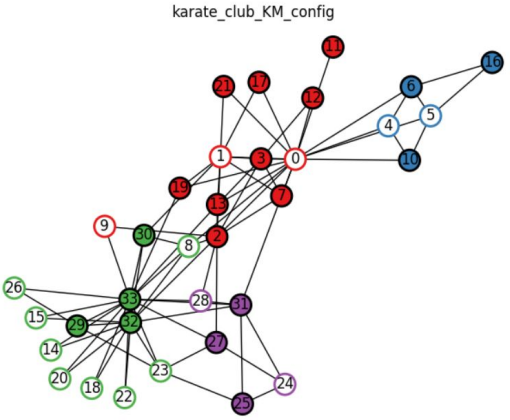

Figure 6 Plot of Karate-club network

B. *Erdos-Renyi random graph*

Similarly, the relation table of E-R random graph with n=80, p=0.1 is shown below:

	Name	PairID	Coreness
1	0	0	0.000000
2	2	0	1.000000
3	7	0	1.000000
4	13	0	0.000000
5	22	0	1.000000
6	30	0	1.000000
7	55	0	1.000000
8	71	0	0.000000
9	1	1	1.000000
10	16	1	1.000000
11	33	1	1.000000
12	39	1	0.000000
13	66	1	0.000000
14	67	1	1.000000
15	79	1	0.000000
16	3	2	0.000000
17	6	2	1.000000
18	29	2	1.000000
19	35	2	0.000000

Figure 7 Distribution Table of E-R Random Graph

An example plot of E-R random graph by using KM_ER is shown below:

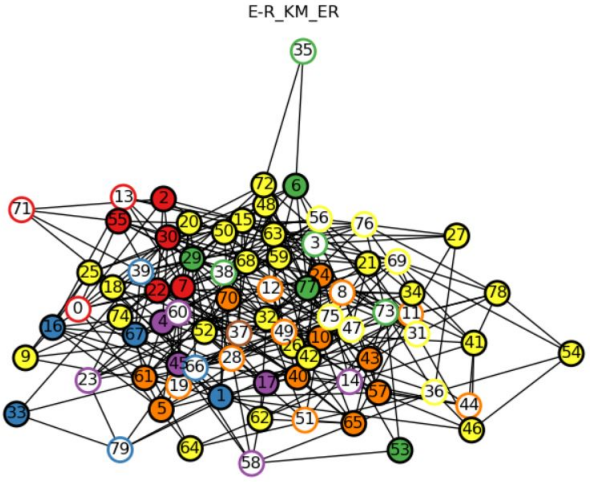

Figure 8 Plot of E-R Random Graph

C. *Bitcoin Transaction Network*

An relation table example of Bitcoin Network by using KM_ config shown as Figure 9 below:

	Name	PairID	Coreness
1	user6	0	1.000000
2	user2	0	1.000000
3	user5	0	0.000000
4	user1	0	1.000000
868	user5844	0	0.000000
869	user13	1	1.000000
870	user16	1	0.000000
871	user25	1	1.000000
872	user57	1	1.000000
1604	user5987	1	1.000000
1605	user26	2	1.000000
1606	user1863	2	0.000000
1607	user1985	2	0.000000
1608	user2130	2	0.000000
4754	user2572	227	0.000000
4755	user2573	227	0.000000
4756	user2568	228	1.000000
4757	user2574	228	1.000000
4758	user2628	228	1.000000

Figure 9 Distribution Table of Bitcoin Network

Figure 10 is shown as an example plot of Bitcoin Network by using KM_config:

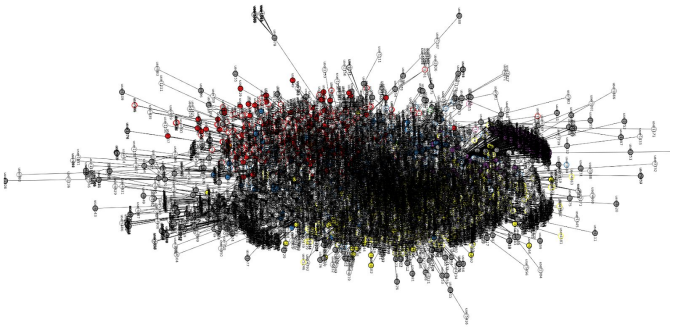

Figure 10 Plot of Bitcoin Network

For the karate club network, we can see clearly it is separated into four colors depending the winning rate. The red core cluster means the players who have high winning rate, the color purple is the opposite. For the Erdos-Renyi random graph with n=80, p=0.2. We can see the classification distribution of c-p pair is not that clear. A considerable amount of nodes that belongs to one color is not gathering as a real cluster. For the Bitcoin transaction network, this is a dataset with over 30,000 transaction records. We are able to see that there are many nodes classified into one c-p pair. We have about 870 nodes that are classified into cp pair 0 and about another 800 nodes that are classified into cp pair 1. That means the result can be highly-centralized. The red nodes and yellow nodes represent cp pair 0 and 1, which are actually centralized.

It is important to build a who-trust-whom network in order to make visualization on transaction network, sports game and any real-world based network which can be implied to a weighted graph. Core-Periphery structure is indeed a remarkable methodology to make this goal come true. With the output of core-periphery pairs figure, we can always get information we want. The result of plot is generally straightforward for user to understand who can be trust or not.

It also has limitation. As we saw previously, when it applied to random graph, the result is not that perfect. And, sometimes, the transition between a real-world network and weighted graph is not straightforward. Additional measures

are needed to be found when it cannot be transformed (e.g., A powerful algorithm is needed if you really want to do that). We saw the result of bitcoin network, it is a little bit hard to identify.

VII. Related Work

According to the article "Core Periphery Structures in Weighted Graphs using Greedy Growth[1]", it clearly introduces the core peripheral structure and uses the protein structure to explain why the core peripheral is so important. The structure of a protein is similar to the network we usually see, with a core and a core-periphery. But can we analyze proteins using ordinary analysis networks? The answer is no because, in the protein, every role is indispensable.

The points mentioned in this article are often overlooked. In the past, we focused on the core analysis and regarded the core-periphery as an exception or a less relevant thing. However, in some situations and environments, all components are not negligible, which also inspired us to want to do this Theme idea.

In complex networks, community structure is a widely used property. To understand many real-world networks in-depth, we need to analyze the community structure of complex networks. Overlapping clusters occur when nodes have membership in multiple communities. Therefore, finding overlapping clusters helps us to discover connections between online communities. Under this prerequisite, the core-periphery algorithm of a single cluster came into being. The single-cluster core-periphery algorithm describes that high-density cores are closely connected and sparsely and loosely connected around. Experiments have proved that due to the existence of overlapping communities in the real world network, the core-periphery structure will naturally appear.

We also use single core-periphery structure function to analyze our datasets. As Figure 11 shown below, it is an example of using BE function to analyze the Karate-club network. According to the Figure 11, we can see the red nodes represented the core of this dataset, and else nodes represented the periphery of this dataset.

Based on the core-periphery algorithm of a single cluster, the researchers construct an algorithm based on greedy growth, which can find the core-periphery structure of multiple clusters in a weighted network. Under this kind of core-periphery structure: the nodes belonging to the core are connected. We call this density structure-based density, and the nodes belonging to the periphery are loosely connected to the nodes belonging to the core.

Through the core-periphery structure of multiple clusters, we can discover the importance of dense cores and peripherals in real-world networks. It is not observed by simple overlapping clusters found by the core-periphery algorithm of a single cluster.

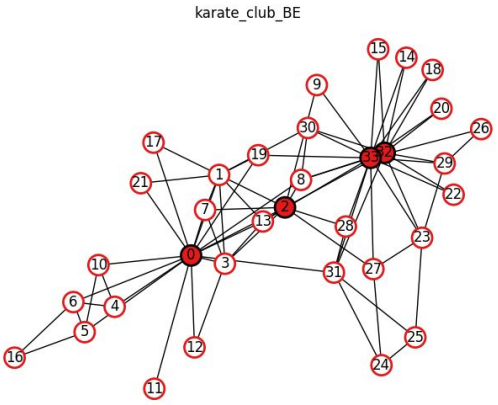

Figure 11 Plot of BE on Karate-club Network

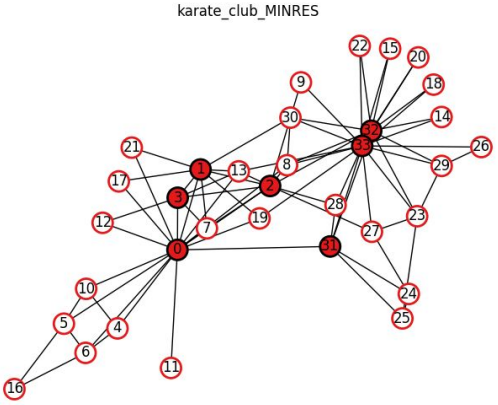

Figure 12 Plot of MINRES on Karate-club Network

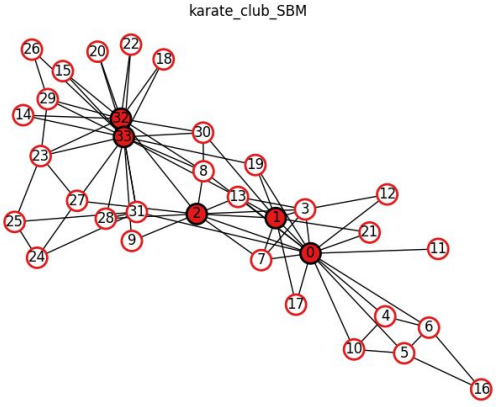

Figure 13 Plot of SBM on Karate-club Network

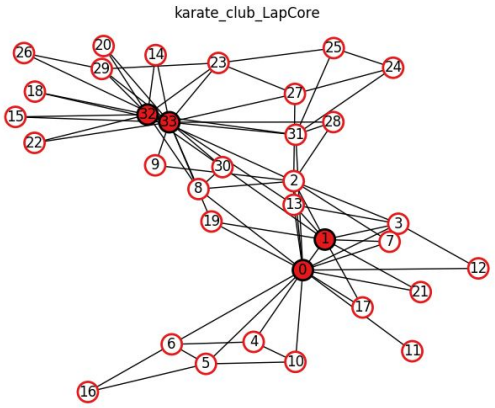

Figure 14 Plot of LapCore on Karate-club Network

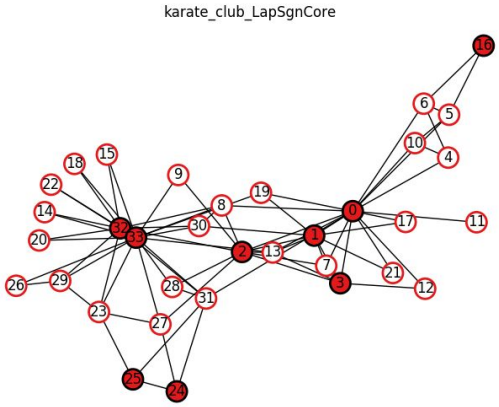

Figure 15 Plot of LapSgnCore on Karate-club Network

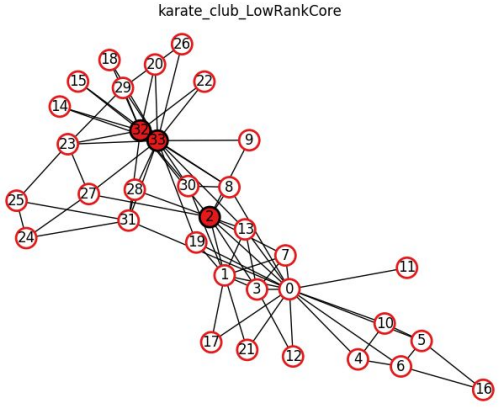

Figure 16 Plot of LowRankCore on Karate-club Network

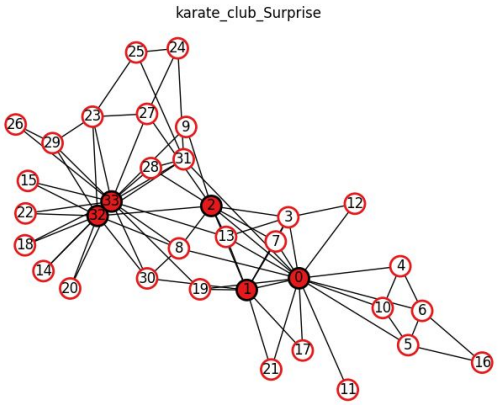

Figure 17 Plot of Surprise on Karate-club Network

As Figure 11-17 shown, we used each core-periphery algorithm of a single cluster to implement the karate-club network. Based on the plots we obtained, we can find that the core-periphery algorithm of a single cluster has limitations. The core-periphery algorithm of a single cluster only uses a structural density measure called cohesion for the growing clusters. Moreover, in combination with real-world networks, it is not practical to make a single cluster. For example, the bitcoin transaction network we analyzed in this project has a large number of nodes and edges, of which 89% are positive edges, and each edge has a weight range from -10 to 10. For bitcoin transactions with a high edge weight, they can be regarded as having a high degree of credibility. Correspondingly, the transactions may obtain a high authority. For example, the transaction can be completed without passing verification. On the contrary, assuming the edge weight is low, we consider the transaction to be of low credibility. It may not only need to complete primary verification but also need to meet more restrictive conditions to complete the transaction. The implementation mentioned above is not possible through a single cluster.

VIII. Conclusion

Generally, we use KM-config implement on two different scale dataset karate club network and bitcoin transaction network, and KM-ER implement on one Erdos-Renyi random graph. The KM-config implementation work well on both datasets, and we are able to see how it will separate the network into multiple cp-pair depending on different scale of networks. However, for the KM-Er implementian on random graph, the result is not satiable. Random graph do not apply to this algorithm.

For the future work that we need to focus on is how to eliminate the noisy nodes when the dataset is tremendously big. For the reason of the appearance of irregular data, it is supposed to have some impact on the formation of the graph.

We need to try to reduce the weight of those nodes kind of by giving coefficient that can be calculated by a specific algorithm to every node to do verification on overall nodes in the network.

References

[1] D. Sardana and R. Bhatnagar, "Core Periphery Structures in Weighted Graphs Using Greedy Growth," 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Omaha, NE, 2016, pp. 1-8.

[2] S. Kojaku and N. Masuda. Finding multiple core-periphery pairs in network. Phys. Rev. 96(5):052313, 2017.

[3] S. Kojaku and N. Masuda. Core-periphery structure requires something else in the network. New Journal of Physics, 20(4):43012, 2018.

[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] S. P. Borgatti and M. G. Everett. Models of core/periphery structures. Soc.~Netw., 21(4):375–395, 2000.

[6] J. P. Boyd, W. J Fitzgerald, M. C. Mahutga, and D. A. Smith. Computing continuous core/periphery structures for social relations data with MINRES/SVD. Soc.~Netw., 32:125–137, 2010.

[7] S. Z. W.~ Lip. A fast algorithm for the discrete core/periphery bipartitioning problem. arXiv, pages 1102.5511, 2011.

[8] X. Zhang, T. Martin, and M. Newman. Identification of core-periphery structure in networks. Phys. Rev. E., 91(3):032803, 2015.

[9] M. Cucuringu, P. Rombach, S. H. Lee, and M. A. Porter Detection of core-periphery structure in networks using spectral methods and geodesic paths. Euro. J. Appl. Math., 27:846–887, 2016.

[10] P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha. Core-Periphery Structure in Networks (Revisited). SIAM Review, 59(3):619–646, 2017

[11] J. van Lidth de Jeude, G. Caldarelli, T. Squartini. Detecting Core-Periphery Structures by Surprise. Preprint arXiv:1810.04717 (2018).

APPENDIX

Results and codes are all on Github:
https://github.com/angelxd84130/Core-Periphery-Structures-