Exploring practical use of Hierarchical Core-Periphery structures in networks

Social Network Analysis for Computer Scientists — Course paper

Davide Baggio
d.baggio.2@umail.leidenuniv.nl
LIACS, Leiden University (Erasmus+)
University of Padua

Maksim Starostenko
m.starostenko@umail.leidenuniv.nl
LIACS, Leiden University

ABSTRACT

Core-periphery structures provide valuable insights into network connectivity by identifying central and peripheral nodes. This research explores a hierarchical approach to core-periphery detection, using a Monte Carlo algorithm to analyze network structures across diverse datasets. Our study examines networks from various domains, including social networks, literary critic interactions, fictional character relationships, and organizational structures. Using an algorithm that allows detection of up to five core-periphery groups, we applied the method to eleven different networks ranging from small (16 nodes) to medium-sized (115 nodes). The experiments revealed the algorithm's capability to identify complex network structures, though performance varied across datasets and highlighted challenges in precise node categorization. The study contributes to understanding network complexity and suggests future improvements in hierarchical network analysis techniques.

KEYWORDS

hierarchical core-periphery,clustering, social network analysis, network science

ACM Reference Format:
Davide Baggio and Maksim Starostenko. 2024. Exploring practical use of Hierarchical Core-Periphery structures in networks: Social Network Analysis for Computer Scientists — Course paper. In *Proceedings of Social Network Analysis for Computer Scientists Course 2024 (SNACS '24).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

The core-periphery structure is a common type of clustering method used to detect underlying features in network graphs. Its main objective is to identify clusters in the network by extracting graph elements (nodes) based on their features and behavior. In this structure, the core nodes are considered more central and important, while the peripheral nodes are less important, but still depend on the core nodes.
The core-periphery detection algorithm is essential for understanding the structure of various networks, offering valuable insights across a wide range of applications, from small-scale networks to large ones. It is particularly useful in analyzing and explaining relationships in diverse domains such as economic activity between countries or regions, as well as in social, biological, and financial networks.
The traditional core-periphery structure consists of two distinct groups: the core and the periphery. Nodes in the core are typically more densely interconnected together, while peripheral nodes have fewer connections between themselves and rely more heavily on the core nodes for connectivity. The connection between core and periphery can also be either highly concentrated or low-concentrated edge-wise. When the structure of a graph is extended to a multiple number of groups, the algorithm provides a flexible shift to a more complex structure of a network, where it accounts for all different kinds of possible structures of the network, meaning that there can be multiple cores, that are parallel, nested and located at different hierarchical levels of the network.
The following algorithm ensures a great generalization on a number of different possible structures, while other similar algorithms that try to identify core-periphery structure lack on capabilities of detecting special circumstances. For example, k-core decomposition algorithm builds a structure based on the degree distribution of the nodes, which may not align well with an actual core-periphery structure. Other popular algorithms are "Quality Function" and "Stochastic Block Models". These algorithms are highly sensitive to parameters selection, making them challenging to optimize and susceptible to errors.
Although the proposed algorithm appears to be both innovative and highly effective, offering generalization and optimization to accommodate various structural variations, this claim requires further validation. Therefore further experiments and testing are needed: the algorithm will be tested on 11 datasets in order to analyze and assess its performance across diverse structures and settings.

2 RELATED WORK

Over the years, researchers have explored various methodologies and perspectives on core-periphery structures, deepening our understanding of how these configurations influence network connectivity, robustness, and community organization.
One of the first studies to introduce core structures was the paper by Seidman [15] in 1983, which proposed a method for identifying core nodes in social networks based on their connectivity, defining a k-core as a maximal subgraph where each node has at least k connections to other nodes within the subgraph. Subsequently, Borgatti and Everett [3] in 1999 formally introduced the core-periphery model, defining cores as densely connected nodes and peripheries

as sparsely connected nodes linked primarily to the core.
An important change was introduced by Rombach et al. [13] in 2013, who proposed a method for detecting hierarchical core-periphery structures, allowing for the identification of multiple nested core layers within a single network. This approach provides a more nuanced understanding of network organization, enabling the analysis of multilevel interactions within complex networks.
A key reference for this study is the 2023 paper by Polanco and Newman, *Hierarchical Core-Periphery Structure in Networks* [12]. This work advances the understanding of core-periphery configurations by introducing a method for detecting nested core and periphery regions across multiple hierarchical levels, offering a richer perspective on the layered complexity within network structures.
Building on these foundational studies, our research aims to assess the effectiveness of hierarchical core-periphery detection across a diverse set of networks, contributing to the ongoing exploration of core-periphery structures in complex networks.

3 PRELIMINARIES

In this model, nodes are organized into overlapping groups, allowing for flexible membership across multiple categories. The rules for determining group membership and the probabilities of forming edges between nodes are described below.

- Nodes can belong to any of the k groups, which are labeled by $r = 0, ..., k-1$ and all of them must also belong to group 0.
- The notation used to describe if a node belongs to a specific group is the following: $g_u^r = 1$ if node u belongs to group r, and 0 otherwise.
- We use a set of probabilities, ω_r, for each group including group 0. Edges are placed between node pairs independently with probability ω_r, where r is the highest common group that both nodes in the pair belong to, meaning the one with the highest number. For example, if one node belongs to groups 0, 1, 2 and another belongs to 0, 1, 3, then there is an edge between them with probability ω_1.

The end result is a graph that has the following characteristics:

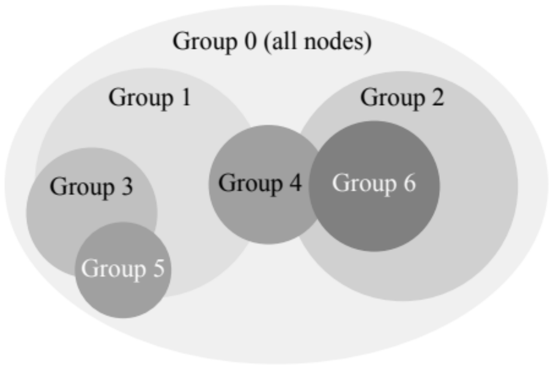

Figure 1: General core-periphery structure.

An important theorem which is the basis of the algorithm created by the authors of hcp is the Monte Carlo algorithm. It is a computational technique used to explore and sample a solution space through random trials and is especially useful for problems involving optimization over large complex spaces. The core idea is to use random sampling to propose changes to the current state of the system and decide whether to accept these changes based on a probabilistic criterion. For the fixed number of groups, the algorithm iteratively modifies group assignments by randomly adding or removing nodes from groups. It evaluates each proposed change using an acceptance probability based on the ratio of the likelihoods of the new and current assignments. The algorithm for a varying number of groups dynamically adjusts the number of groups during each iteration. Similarly, as in the fixed-number-of-groups case, changes such as adding or removing groups and reassigning nodes are probabilistically accepted based on the likelihood ratio.

On the following page, we present the pseudo-code for two scenarios: one with k fixed and one with k variable. Both algorithms aim to identify optimal group structures while accounting for hierarchical, overlapping, or other complex relationships among nodes.

Algorithm 1 Monte Carlo algorithm for the case of fixed k

```
1:  Input:
2:      $n$: Total number of nodes
3:      $k$: Total number of groups
4:      $P(A|g,k)$: Probability function for the current group assignments
5:      $g$: Initial group assignments (array where $g[i]$ is the group of node $i$)
6:  repeat
7:      Choose a group $s$ uniformly at random from the range $[1, k-1]$.
8:      With probability 0.5:
9:      if remove a node from group $s$ then
10:         if group $s$ is not empty then
11:             Select a node uniformly at random from those currently in group $s$.
12:             Store $g'$ by removing the selected node from group $s$ in $g$.
13:         else
14:             Skip this proposal and move to the next iteration.
15:         end if
16:     else
17:         if group $s$ is not full then
18:             Select a node uniformly at random from those currently not in group $s$.
19:             Store $g'$ by adding the selected node to group $s$ in $g$.
20:         else
21:             Skip this proposal and move to the next iteration.
22:         end if
23:     end if
24:     Compute the acceptance probability for the proposed move:
            $$\alpha(g \to g') = \min\left(1, \frac{P(A|g',k)}{P(A|g,k)}\right)$$
25:     Generate $u = Uniform(0,1)$.
26:     if $u < \alpha(g \to g')$ then
27:         Accept the move, and set $g = g'$.
28:     else
29:         Reject the move, and revert to $g = g$.
30:     end if
31: until convergence
```

Algorithm 2 Monte Carlo algorithm for varying k

```
1:  Input:
2:      $n$: Total number of nodes
3:      $k$: Total number of groups
4:      $P(A|g,k)$: Probability function for the current group assignments
5:      $g$: Current group assignments (array where $g[i]$ is the group of node $i$)
6:  repeat
7:      Choose a move type with probability:
8:      if $Uniform(0,1) < 1 - \frac{1}{2k(n+1)}$ then
9:          if $k = 1$ then
10:             do nothing
11:         else
12:             Choose a group $s$ uniformly at random from $[1, k-1]$.
13:             With probability $\frac{1}{2}$:
14:             if add a node to group $s$ then
15:                 if group $s$ is not full then
16:                     Select a node uniformly at random from those not currently in group $s$.
17:                     Store $g'$ by adding the selected node to group $s$ in $g$.
18:                 else
19:                     do nothing
20:                 end if
21:             else
22:                 if group $s$ is not empty then
23:                     Select a node uniformly at random from those currently in group $s$.
24:                     Store $g'$ by removing the selected node from group $s$ in $g$.
25:                 else
26:                     delete group $s$ and reduce $k$ by 1
27:                     adjust labels of groups above $s$
28:                 end if
29:             end if
30:         end if
31:     else
32:         Choose a group index $s$ uniformly at random from $[1, k]$.
33:         Increase the labels of all groups $s$ and greater.
34:         Create a new empty group with label $s$.
35:         Increase $k$ by 1.
36:     end if
37:     Compute the acceptance probability for the proposed move:
            $$\alpha(g, k \to g', k') = \min\left(1, \frac{P(A|g',k')}{P(A|g,k)}\right)$$
38:     Generate $u = Uniform(0,1)$.
39:     if $u < \alpha(g, k \to g', k')$ then
40:         Accept the move, and set $g = g'$ and update $k = k'$.
41:     end if
42: until convergence
```

4 APPROACH

To facilitate the use of the program provided by the authors of the paper [12], we have implemented a Jupyter notebook that allows for the easy execution of the algorithm on different datasets. This notebook was also made to address the challenge of interpreting the results produced by the original program, as they are not straightforward to analyze and it produces many `.txt` files.
Our implementation includes additional processing to clarify where nodes appear within the detected structures by scraping information from 3 of the generated files and searches for:

- the minimum negative log-likelihood and its index;
- the best nodes' grouping configuration based on the previously mentioned index;
- number of groups generated.

Additionally, it can automatically plot a graph if the scraped `max_number_groups` is equal to 2.

The notebook is available at the following link:
https://github.com/dvdbaggio/hcp.

5 DATA

As datasets, we use the following networks:

- **Museum**: it's the directed network that represents the connections between rooms, which was created as part of Assignment 2 in the SNACS course [5].
  - Nodes: 51;
  - Edges: 70.
- **Dolphins**: it's an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand. [14]
  - Nodes: 62;
  - Edges: 159.
- **Karate**: it's an undirected social network of friendships between 34 members of a karate club at a US university in the 1970s. [16]
  - Nodes: 34;
  - Edges: 78.
- **Les Miserables**: it's an undirected social network of characters in Victor Hugo's novel "Les Miserables". [9]
  - Nodes: 77;
  - Edges: 254.
- **Dining**: Dataset represents dining table partners in a dormitory at a New York State Training School. [11]
  - Nodes: 26;
  - Edges: 42.
- **Literature 1976**: This network contains the critical attention among a set of Dutch literary authors and critics in 1976. It contains an arc between two people if the first has passed judgment on the work of second in an interview or review. [4]
  - Nodes: 34;
  - Edges: 79.
- **Star Wars: Episode 3**: The social network dataset of Star Wars characters extracted from Episode 3, where the links between characters are defined by the times the characters speak within the same scene. [7]
  - Nodes: 26;
  - Edges: 66.
- **Galesburg**: Datasets contains a network of friendship and discussion ties between 16 physicians who adopted the new drug in Galesburg (Illinois) in the 1950s.[8]
  - Nodes: 16;
  - Edges: 27.
- **Flying teams**: Dataset contains a sociometric test to 48 cadet pilots at an US Army Air Forces flying school from 1943 created by Leslie D. Zeleny. Cadets were trained to fly a two-seated aircraft, taking turns in flying and aerial observing. Cadets were assigned to instruction groups ranging in size from 5 to 7 at random, so they had little or no control over who their flying partners would be. [6]
  - Nodes: 48;
  - Edges: 282.
- **Football**: Network example describes the 22 soccer teams which participated in the World Championship in Paris, 1998. Players of the national team often have contracts in other countries. This constitutes a players market where national teams export players to other countries. Members of the 22 teams had contracts in altogether 35 countries. [10]
  - Nodes: 115;
  - Edges: 613.
- **GoT**: it's the directed network of coappearances of characters in the Game of Thrones series, by George R. R. Martin. [1]
  - Nodes: 107;
  - Edges: 352.

We intentionally focused on small to medium-sized networks to ensure the results were easily interpretable. This choice was also necessitated by the need to manually create network graphs, as no existing algorithm accurately plots hierarchical core-periphery structures. While we successfully developed an algorithm for plotting, its functionality is limited to networks with exactly two core-periphery groups, and extending it to handle multiple groups and subgroups remains a challenge.

6 EXPERIMENTS

In this section we present the results of applying the hierarchical core-periphery detection algorithm to the datasets described in section 5.

We ran the algorithm on each network setting the parameter `max_num_groups` to 5, which allows for the detection of up to 5 core-periphery groups in each network. Please note that this means that the algorithm may detect fewer groups if the network does not contain enough structure to support the detection of 5 groups at the configuration with the smallest negative log-likelihood value. Due to the limited space available, and since many graphs contain an important number of nodes, it's not always possible to properly read nodes' labels. You can find the complete list of nodes divisions in the repository at
`Outputs/'dataset name'/xyz_groups_with_labels.txt`

6.1 Museum dataset

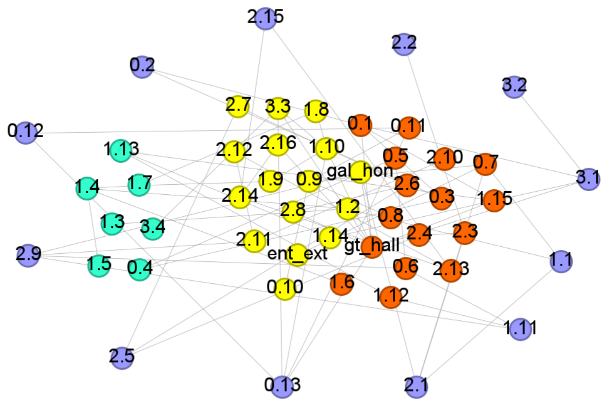

Figure 2: CP museum dataset with 3 groups.

In this graph we can observe that the algorithm detected 3 core-periphery groups in the museum network:

- Group 0: all nodes;
- Group 1: lilac/cyan;
- Group 2: lilac/orange/yellow.

The core-periphery structure is clearly visible, with the core nodes forming a dense cluster in the center of the network, while the peripheral nodes are located on the outskirts. The connections between the core and periphery are well defined, with the core nodes acting as bridges between the different groups.

6.2 Dolphins dataset

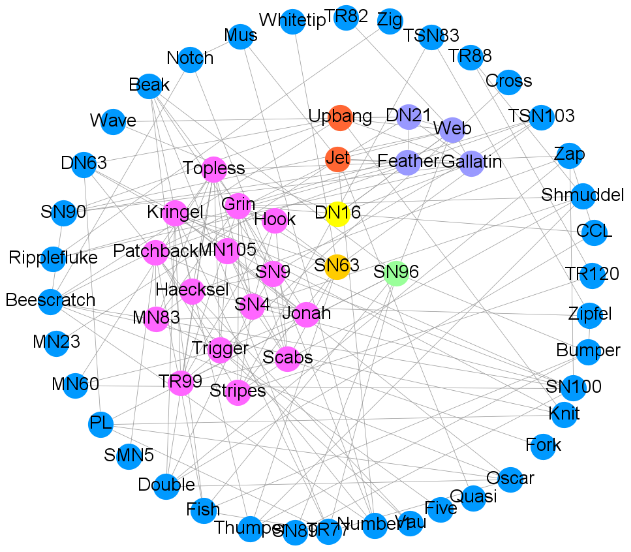

Figure 3: CP dolphins dataset with 4 groups.

In this graph we can observe that the algorithm detected 4 core-periphery groups in the dolphin network:

- Group 0: all nodes;
- Group 1: light-blue/lilac/orange/yellow;
- Group 2: light-blue/pink/ocher/yellow;
- Group 3: light-blue/green/yellow/ocher/orange.

This graph representation is well-structured as well, but is slightly more complex to interpret. The added complexity arises from the presence of an additional group and the fact that many nodes belong to multiple groups.

6.3 Karate dataset

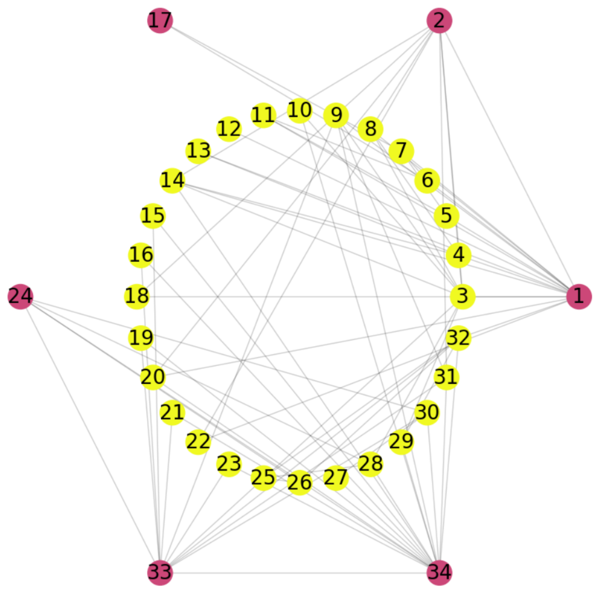

Figure 4: CP karate dataset with 2 groups.

This one is the first time that the algorithm detected the classic 2 core-periphery groups. The core-periphery structure is clearly identifiable; however, we are not confident that the algorithm accurately separated the groups. Many peripheral nodes have numerous edges connecting them to core nodes (notably nodes 1, 33, and 34), while several core nodes exhibit only a few connections with other core nodes. This discrepancy suggests that the algorithm failed to assign nodes to the appropriate groups. Because of this, we used the same dataset with the community detection algorithm developed by Vincent D. Blondel et al. [2] and we got the following result:

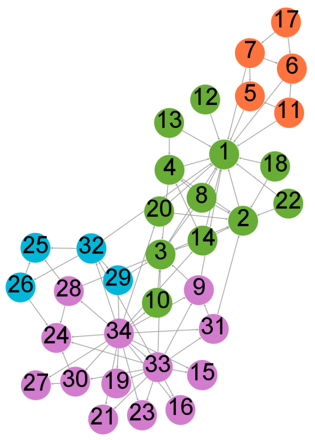

Figure 5: Community detection karate dataset.

As we can see, the community detection algorithm identified 4 groups and the nodes seems to be more accurately assigned to the groups, as each community has strong connections between its nodes and weaker ones with the other communities.

6.4 Les Miserables dataset

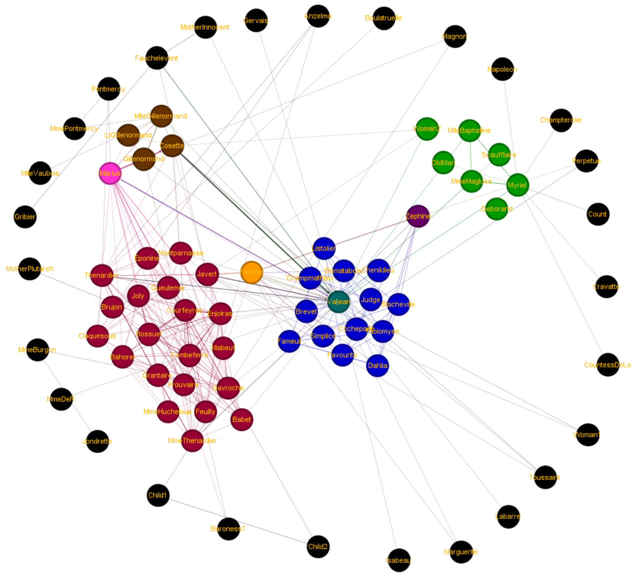

Figure 6: Les Miserables dataset with 5 groups.

In this graph we can observe that the algorithm detected 4 core-periphery groups in the Les Miserables network:

- Group 0: all nodes;
- Group 1: red/pink/orange/sea-blue;
- Group 2: green/purple/sea-blue;
- Group 3: pink/brown/sea blue;
- Group 4: orange/purple/blue/sea blue.
- Group 5: sea blue.

Figure 6 illustrates the core-periphery structure of the Les Miserables dataset, characterized by five distinct groups. The primary groups are well-defined and visually distinct, with the periphery clearly depicted and represented by nodes colored in black. Notably, there are instances of overlap (represented by nodes in pink, orange and purple) between certain groups reflecting the interconnected nature of the dataset. Interestingly, the core structure is represented by a single node. This is caused by the dense connectivity with all other groups. Overall, the proposed structure is well-constructed, with the only limitation being that the core node is isolated.

6.5 Dining dataset

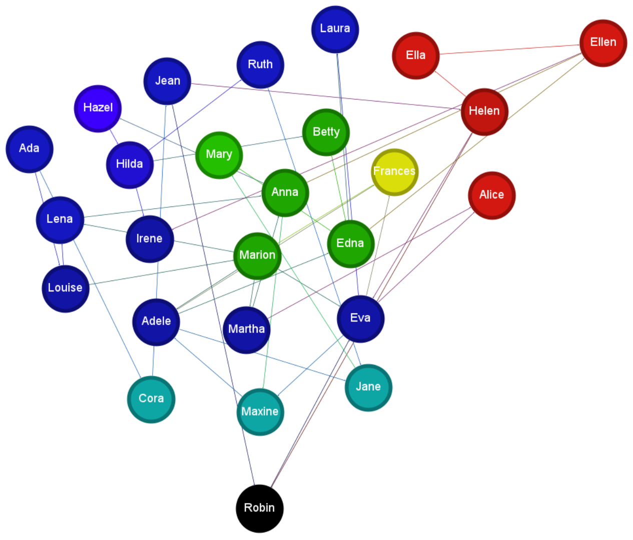

Figure 7: Dining dataset with 4 groups.

In this graph we can observe that the algorithm detected 4 core-periphery groups in the dining network:

- Group 0: all nodes;
- Group 1: sea-blue;
- Group 2: blue;
- Group 3: green/yellow/red.

Figure 7 illustrates the core-periphery structure observed in a dining dataset. The majority of nodes are organized into distinct groups, with a single black node representing the periphery. The graph shows a hierarchical and overlapping structure, where the core is composed of green, yellow, and red nodes. The green nodes belong to a broader blue group, indicating their participation in an overlapping region. Similarly, the blue nodes are nested within a more general sea-blue group, further highlighting the multi-level and overlapping nature of the structure. Overall, the structure is not optimal since given a small number of nodes it assigned excessive amount of nodes to different groups, which makes it too complicated and less informative.

6.6 Literature dataset

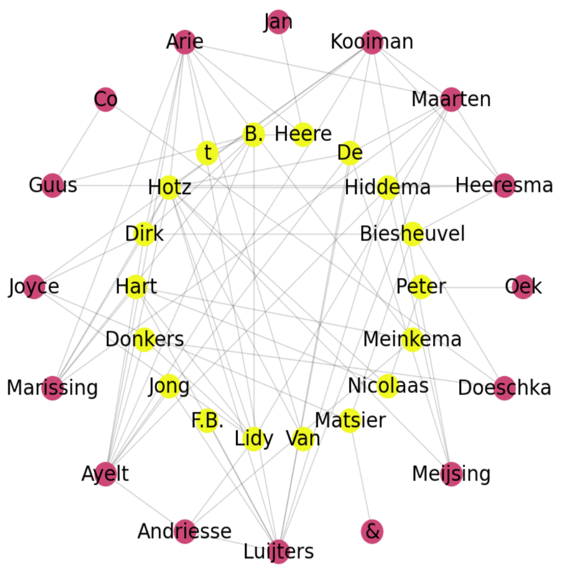

Figure 8: Literature dataset with 2 groups.

This time the algorithm generated a more plausible node division, since peripheral nodes have weaker connections and they mainly rely on edges with core nodes, that are tightly connected to each other.

6.7 Star Wars: Episode 3 dataset

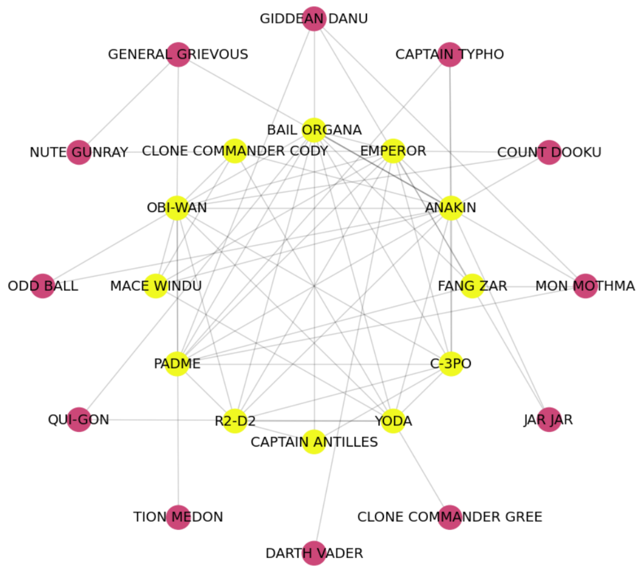

Figure 9: Star Wars: Episode 3 dataset with 2 groups.

Here's another example of two groups structure, however the results are more satisfactory, as the core nodes exhibit stronger interconnections, while most peripheral nodes demonstrate weaker connections. This is the best representation of the two groups we have achieved so far.

6.8 Galesburg dataset

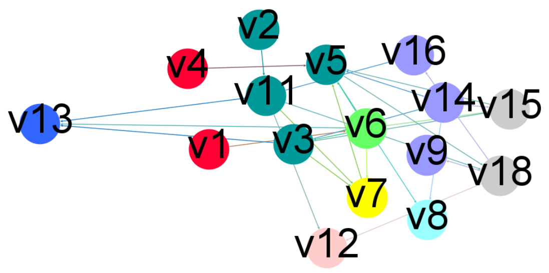

Figure 10: Galesburg dataset with 5 groups.

In this graph we can observe that the algorithm detected 5 core-periphery groups in the dining network:

- Group 0: all nodes;
- Group 1: red/green/teal/gray;
- Group 2: green/lilac/gray/light-blue;
- Group 3: red/green/yellow/pink;
- Group 4: yellow/light-blue.

The algorithm assigned only a single node to the periphery, which is suboptimal. Apart from this issue, the other subgroups were correctly identified, however, for such a simple dataset of only 16 nodes, the algorithm produced too many subgroups. In contrast, the community detection algorithm [2] successfully identified 3 distinct groups, which sounds more reasonable.

6.9 Flying teams dataset

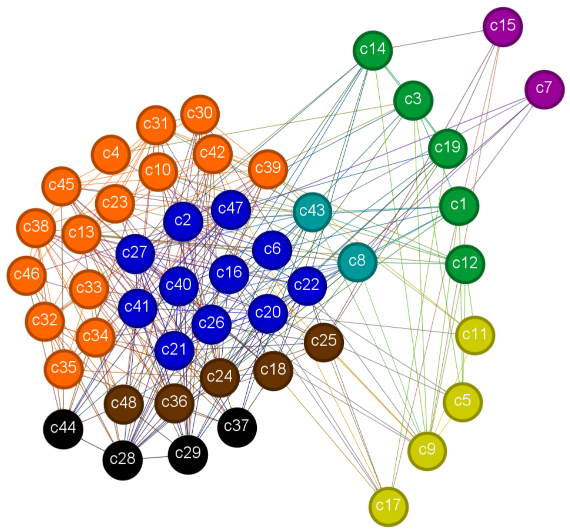

Figure 11: Flying teams dataset with 5 groups.

In this graph we can observe that the algorithm detected 4 core groups in the flying teams network:

- Group 0: all nodes;
- Group 1: blue/green/purple/sea-blue/orange;
- Group 2: blue/green/yellow/orange/black/purple;
- Group 3: green/yellow/blue/sea-blue/brown/orange;
- Group 4: blue/brown/orange/black/sea-blue.

The algorithm performed suboptimally on this dataset, producing an overly complex structure. It assigned an excessive number of nested hierarchical and overlapping features, which is disproportionate to the dataset's small size. Furthermore, it failed to assign any nodes to the periphery.This omission suggests a potential misinterpretation of the dataset's underlying structure.

6.10 Football dataset

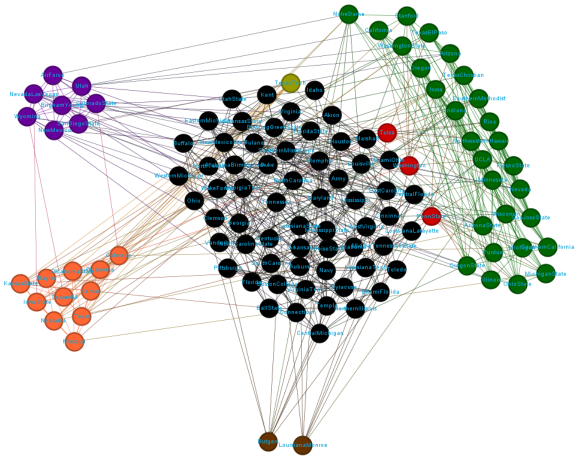

Figure 12: Football dataset with 5 groups.

In this graph we can observe that the algorithm detected 4 core groups in the flying teams network:

- Group 0: all nodes;
- Group 1: orange;
- Group 2: purple;
- Group 3: green;
- Group 4: black/red/yellow.

The structure is clearly displayed, showing distinct groups of nodes with only a few overlaps between them. This makes it easy to see how the nodes are organized. However, the main drawback is that the algorithm placed only two nodes (shown in brown) in the periphery. This might suggest that the algorithm didn't fully capture the less connected or boundary nodes of the network. Still, the overall structure is well-organized.

6.11 GoT dataset

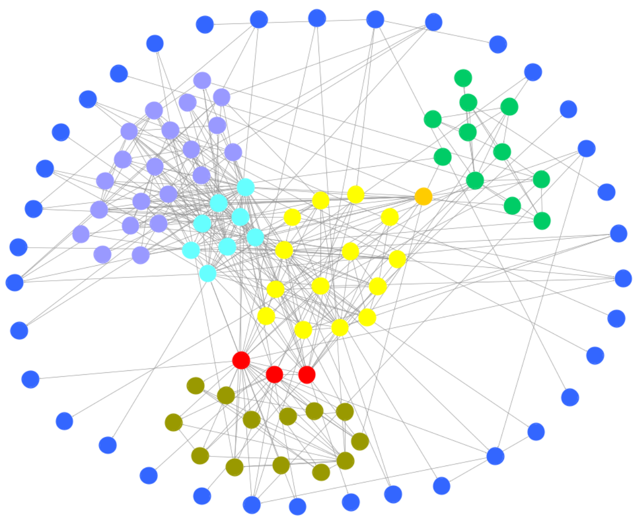

Figure 13: GoT dataset with 5 groups.

In this graph we can observe that the algorithm detected 5 core groups in the flying teams network:

- Group 0: all nodes;
- Group 1: mustard/red;
- Group 2: lilac/light-blue;
- Group 3: orange/green;
- Group 4: red/light-blue/orange.

Although this dataset is one of the largest tested in our experiment, we are satisfied with the division of nodes and the resulting structure. It is evident that each subgroup exhibits stronger internal connections among its nodes, successfully achieving a correct hcp division.

7 CONCERNS AND LIMITATIONS

The algorithm demonstrates a tendency to overcomplicate the structure, particularly for small datasets. On datasets with simple or limited structures, the algorithm often extracts unnecessarily complex core-periphery representations, assigning only a single node to the periphery group. This overcomplication raises significant concerns, as small datasets would ideally yield simpler, more interpretable structures, such as 2-3 distinct groups at most. A simpler approach would enhance clarity and ensure the resulting core-periphery structures are both informative and meaningful. To address this issue, manually reducing the maximum number of desired groups could be considered. However, this solution may not be optimal, as it introduces manual intervention, which undermines the algorithm's automated nature and generalizability. A more robust approach would involve incorporating adaptive mechanisms that dynamically adjust the number of groups based on the dataset's size and structural complexity.
Moreover, the algorithm's effectiveness diminishes when tasked with finding balanced and interpretable outcomes. For instance,

when constrained to a two-group structure, the algorithm performs more effectively, but its random nature persists. As a result, it does not consistently represent the true core-periphery dynamics within the data. This inconsistency suggests that the group assignments require further scrutiny and adjustment to ensure meaningful allocations.

One of the key limitations lies in the algorithm's apparent inability to converge to the most likely representation of the graph structure. Instead, the algorithm frequently produces representations that, while technically feasible, may not reflect the true underlying structure of the dataset. This issue could be caused by the probabilistic nature of the algorithm, that is designed to explore a range of configurations stochastically. The algorithm might be falling into local minima, meaning it might get stuck on solutions that aren't the best possible. It doesn't seem to focus enough on minimizing errors or improving the quality of the results. Instead, it relies heavily on random changes and likelihood calculations, which don't always help it move toward better solutions. This makes it harder for the algorithm to break free from less optimal outcomes and find the most likely and meaningful structure.

Overall, these results indicate that the algorithm, in its current form, may not be well-suited for a wide range of datasets. Significant tuning and refinement are necessary to improve its ability to produce balanced, interpretable structures. Introducing more robust mechanisms to minimize loss and avoid local minima could potentially enhance the algorithm's ability to converge on the most likely and informative representations of core-periphery structures.

8 CONCLUSION

The research conducted by Polanco and Newman [12] provides a valuable contribution to the field of network science by introducing a method for detecting hierarchical core-periphery structures in networks. This approach offers a more nuanced understanding of network organization, allowing for the identification of multiple nested core layers within a single network.

As demonstrated in the experiments section, while the algorithm effectively identifies core-periphery structures across a wide range of networks, it is not universally optimal for all datasets, as it may occasionally fail to assign nodes to their respective groups.

Future improvements in this area could include developing an algorithm to effectively visualize subgroup structures, as this feature is currently absent. Enhancing network graph representations is crucial for comprehensive analysis and interpretation of the results, offering a clearer understanding of the hierarchical core-periphery organization. Another area to improve could be the development of adaptive mechanisms that dynamically adjust the number of groups based on dataset size and complexity, reducing the tendency to overcomplicate results. Additionally, introducing more robust optimization techniques to avoid local minima and systematically minimize errors could enhance the algorithm's convergence toward more accurate solutions.

REFERENCES

[1] Andrew Beveridge and Jie Shan. 2016. Network of Thrones. *Math Horizons* 23, 4 (2016), 18–22. https://doi.org/10.4169/mathhorizons.23.4.18 arXiv:https://doi.org/10.4169/mathhorizons.23.4.18

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (Oct. 2008), P10008. https://doi.org/10.1088/1742-5468/2008/10/p10008

[3] Stephen Borgatti and Martin Everett. 1999. Models of Core/Periphery Structures. *Social Networks* 21 (11 1999), 375–395. https://doi.org/10.1016/S0378-8733(99)00019-2

[4] W. de Nooy. 1999. A literary playground. Literary criticism and balance theory. Poetics. , 26:385-404 pages. https://sites.google.com/site/ucinetsoftware/datasets/dutchliterarycriticism1976.

[5] dr. Frank Takes and dr. Vincent Traag. 2024. Social Network Analysis for Computer Scientists. https://liacs.leidenuniv.nl/~takesfw/SNACS/ Accessed: 15-12-2024.

[6] J. L . Moreno et.al. 1960. The Sociometry Reader. , 534-547 pages. https://sites.google.com/site/ucinetsoftware/datasets/flyingteams.

[7] Evelina Gabasova. 2016. Star Wars social network. https://doi.org/10.5281/zenodo.1411479

[8] R.S. Knoke D., Burt. 1983. Prominence, in R.S. Burt and M.J. Minor (Eds.), Applied Network Analysis. A Methodological Introduction. , 195-222 pages. https://sites.google.com/site/ucinetsoftware/datasets/galesburgdrugstudy.

[9] Donald E. Knuth. 1993. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA.

[10] Lothar Kremple. 1999. Football data set. http://www.casos.cs.cmu.edu/tools/datasets/external/index.php

[11] J. L. Moreno. 1960. The Sociometry Reader. , 35 pages. https://sites.google.com/site/ucinetsoftware/datasets/dining-tablepartnersinadormitoryatanewyorkstatetrainingschool.

[12] Austin Polanco and M. E. J. Newman. 2023. Hierarchical core-periphery structure in networks. *Physical Review E* 108, 2 (Aug. 2023). https://doi.org/10.1103/physreve.108.024311

[13] M. Puck Rombach, Mason A. Porter, James H. Fowler, and Peter J. Mucha. 2013. Core-Periphery Structure in Networks. arXiv:1202.2684 [cs.SI] https://arxiv.org/abs/1202.2684

[14] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. https://networkrepository.com

[15] Stephen B. Seidman. 1983. Network structure and minimum degree. *Social Networks* 5, 3 (1983), 269–287. https://doi.org/10.1016/0378-8733(83)90028-X

[16] Wayne Zachary. 1976. An Information Flow Model for Conflict and Fission in Small Groups1. *Journal of anthropological research* 33 (11 1976).