



A k-MCST Based Algorithm for Discovering Core-Periphery Structures in Graphs

Susheela Polepalli^(✉) and Raj Bhatnagar^(✉)

University of Cincinnati, Cincinnati, OH 45221, USA
polepasa@mail.uc.edu, bhatnark@ucmail.uc.edu

Abstract. Core-periphery structures are examples of meso-scale characteristics of graphs. Most existing algorithms for core-periphery (CP) structures work by first finding the dense cores of a network and then discovering the peripheral nodes around them. Our algorithm presented here seeks to query a graph to return the CP structures centered around any selected query node. Our algorithm significantly reduces the computational complexity of repeatedly querying the CP structures from a network. Our algorithm repeatedly extracts minimum cost spanning trees (MCSTs), first from the original network, and then successively from the residual networks. From the union of these MCSTs, our algorithm efficiently answers the queries for CP structures around nodes. We validate our algorithm on example networks taken from two domains.

Keywords: Graph mining · Core-periphery structures · Community detection

1 Introduction

The underlying data for a number of domains can be viewed in the form of graph structures. This representation denotes entities as nodes, and strengths of relationships among entities as edge labels. Most of the existing community detection algorithms are based on one of the following main ideas: minimum cut method, hierarchical clustering, Girvan–Newman algorithm [12], and modularity maximization method. In all these algorithms a common theme is to identify densely connected subgraphs of the complete graph.

Communities having core-periphery (CP) structures are observed in many domains. They occur in the form of a densely connected core set of nodes, surrounded by a layer of relatively loosely connected peripheral nodes, as introduced by Borgatti and Everett in [2]. A number of approaches have attempted to find CP structures using graph-based ideas of overlapping communities [4], connected k-plexes [1], and random walk signatures [9]. There are quite a few applications in social and scientific networks that require us to repeatedly query the networks for CP community structures around specific individual nodes of the network. We may want to ask for the social group around a single individual in a social network or for the closest proteins to a specific protein in a biological network.

These query nodes may or may not be parts of dense clusters of the complete network. To address this need we reduce the size of the graph, by dropping weaker edges, so that repeated query tasks are made computationally efficient.

To construct the CP structure from the reduced graph, we initialize the core set by including in it all the immediate 1-hop neighbors of the query node. We compute the edge density of this initial core and set it as the initial threshold for edge density. Neighbors of the nodes that are already in the core are then successively added to the core such that these node additions do not reduce the edge density. We then adds layers of peripheral nodes that follow a similar closure property on edge density, and the density thresholds decrease successively from one peripheral layer to the next. Our algorithm uses the clustering coefficient of each node, if needed, to identify nodes that have high connectivity around them.

2 Related Work

The work by Borgatti and Everett in [2] identified the CP structures as dense blocks of 1's, connected to sparser blocks, in graphs' adjacency matrices. Exploiting the properties of overlapping communities, J. Yang and J. Leskovec used the idea of overlapping tiles [4] to identify CP structures. The algorithm *ClusterONE-CP* [3] also works with overlapping communities discovered by the clustering algorithm *ClusterONE* and analyzes the core and overlapping sections of communities to discover the CP structures. Silva et al. [8] have defined core coefficients for nodes using closeness centrality and community modularity metrics. Most of these algorithms seek to first identify densely connected cores and then find nodes that are peripheral to these cores. One problem with algorithms based on centrality and affinity metrics for nodes and edges is that each core community may have a different average value for these centrality metrics. Some of the algorithms will not be able to work with such variations of average densities among cores across the whole graph. In our algorithm we achieve independence from affinity variations by considering a few most significant of the minimum cost spanning trees (MCSTs) extracted from the target graph. Each MCST tends to select the highest affinity edges for each neighborhood. The concept of Merged MCSTs was introduced in [7] where it was shown that the K-merged MCST neighborhood graphs perform better than the k-nearest neighbor graphs for capturing the notion of distance (or affinity) between pairs of nodes in a graph. Two rounds of MCSTs were also used in *2MSTclus* algorithm [5] to extract separate but touching clusters. Use of k-MCSTs by our algorithm has, therefore, precedence and also justification for their use in effectively capturing inter-node similarities in graph clustering algorithms.

3 Our Approach

The first phase of our algorithm constructs the k -MCST graph from the original graph. This is done by first determining the MCST of the graph and then dropping from the graph all those edges that are included in the MCST. We again find the MCST from the residual graph and repeat this process until we have

obtained k MCSTs. If the graph is not connected and has multiple components, we can obtain and work with the Minimum Cost Spanning Forests (MCSFs). We then merge the first k MCSTs to form the graph within which we will look for the CP structures for query nodes. The resulting graph has the properties [7] that it is k -connected, and contains only the highest affinity edges. This ensures that each node is connected to at least k other nodes, and these k edges have the strongest affinity in the neighborhood of this node.

3.1 Definitions

Structural Density ρ : We introduce a metric called structural density for a cluster of nodes, defined as: $\rho = \frac{\text{Number of edges in the cluster}}{\text{Number of possible edges in the cluster}}$. When a set of nodes is included in a cluster, all edges from the k -MCST that connect these nodes among themselves are also included in the cluster.

Core-Periphery Structure: A core-periphery network for a query node N , derived from a k -MCST neighborhood graph, consists of a set of nodes constituting the core C , surrounded by its sets of nodes for peripheral layers ($P_1 \cup P_2 \cup \dots \cup P_N$). Let us say a core C is a set of nodes, and p is a node that is not in C but has a direct edge to at least one node in C . Then considering all those edges that can be included in C from the k -MCST, a core set of nodes C has the following property: For each nodes p connected to C , the density $\rho(C) \geq \rho(C \cup \{p\})$. That is, C is that closed set of nodes to which no other immediate neighboring node can be added without reducing its structural density.

We add a caveat here about a limiting situation for the above definition of a core set. It is possible that the affinity values of the query node to its nearest neighbors are the smallest in a very large neighborhood around it. This will result in very large cores for such nodes, and in extreme cases the entire graph may be included as core for a node. In real graphs such nodes are rare. To account for such query nodes we may place a limit on the growth of a core set.

We consider a number of peripheral layers of nodes that surround the core. Each successive layer's affinity to the core nodes is weaker than that of the preceding layer. A node p belongs to the j^{th} peripheral layer P_j when it has a direct edge to at least one of the nodes in core set C and the following are satisfied: for $j == 1$: (1^{st} peripheral layer) : $\rho(C) > \rho(C \cup \{p\}) \geq P_j\text{-threshold}$ and for $j > 1$: $P_{j-1}\text{-threshold} > \rho(C \cup \{p\}) \geq P_j\text{-threshold}$.

3.2 Algorithm

Our methodology has the following two main phases. *Phase 1:* Construction of k -MCST graph from the original graph. We also compute the clustering coefficient of each node in the k -MCST graph to aid in choosing the nodes in the denser parts of the graph, if needed. *Phase 2:* Construction of core-periphery structures for query nodes guided by the structural density metric

Phase 1: We construct the k -MCST graph from the original graph as follows [7]. Let $G = (V, E)$ denote the complete graph where the label for each edge represents the inverse of the affinity between its two nodes. $MCST_1$ denotes the

set of edges of this graph's MCST, that is, $MCST_1 = MCST(V, E)$. The second MCST is computed from the graph that results after removing all edges of $MCST_1$ from the original graph. That is, $MCST_2 = MCST(V, (E - \text{Edges-of-}MCST_1))$. Similarly, $MCST_i$ denotes the set of edges of the MCST of G with edges of $\sum_{j=1}^{i-1} MCST_j$ removed from the original graph. We define the k -MCST neighborhood graph as: $k\text{-MCST} = (V, \sum_{i=1}^k MCST_i)$. Typically, a large number of weaker edges are discarded in this process. For the next phase of CP construction we no longer consider the edge labels (affinity values) and work only with the edge densities of the subgraphs.

Phase 2: Construction of CP Structures. We take the query node N and grow its core subgraph around it. The main idea of constructing a core around a seed node is to identify that closed set of nodes and edges, whose structural density can not be either increased or maintained the same by adding any new node. The main steps for growing the core are:

1. Take the query node N and add to it all its immediate neighbors, and also all the edges from k -MCST that connect them. This is the starting core set in the core growth process. We then compute the structural density ρ_{temp} for this core, and set it as the initial density threshold.
2. We now consider all those immediate 1-hop neighbor nodes of the nodes in the core, that are not yet included in the core, and call this set L . We test each member of L , one by one, to see if its addition to the core set increases or decreases the value ρ_{temp} . All those nodes from L that, individually, do not decrease ρ_{temp} , when potentially added to the current core, are marked for addition to the core. After all nodes in L have been individually tested, the marked nodes are added to the core. A new structural density is computed for this new set of nodes. If this new density value is larger than ρ_{temp} then ρ_{temp} is set to this new higher value; else, the old value of ρ_{temp} is retained.
3. Step-2 above is performed repeatedly until there is no node in L that gets marked for addition to the core. In this case the core growth process is stopped.

For any set of nodes L the process of adding nodes to the core is independent of the order in which individual nodes in L are tested for their effect on structural density.

In effect, a core set of nodes includes (i) the query node, (ii) all its 1-hop neighbors, and (iii) all those nodes that are m hops away from the query node, and when added to the core, maintain or exceed the edge density of the core formed from nodes that are at least $m-1$ hops away. This results in cores that bring in densely connected neighboring nodes irrespective of the number of hops they may be away from the query node.

Defining Periphery Thresholds: We define the density thresholds for the peripheral layers in terms of the density of the core. The first peripheral layer threshold for a given core C is: $Periphery_1\text{threshold} = \rho_{core}(1 - r)$. Here r is a parameter by which the density threshold is reduced for each successive peripheral layer from that of the previous layer. That is, if r is set to 0.1 then the density threshold for the first peripheral layer is 0.9 times the

density of the core. The density thresholds for other outer peripheral layers are obtained by successively reducing the threshold by the parameter $(1 - r)$. $Periphery_j threshold = \rho_{core}(1 - j \times r)$ where j is the peripheral layer number.

Identifying Peripheral Nodes: We essentially repeat the same process that is used for core growth, but now use the lower density threshold of peripheral layer $(l-1)$, to add nodes to the first peripheral layer (1). Step-2 and step-3 of the core growth process above are repeated but with the lower threshold. Once the iterations of these steps are completed for a layer, we lower the threshold to that for the next outer peripheral layer and repeat the process. From our results we have seen that it is possible for a node three hops away from the query node to be included in the core, but a node that is two hops away to be in the peripheral layer. Details of this algorithm can be found in [14].

3.3 Complexity Analysis

In the first phase, the neighborhood graph is constructed by merging k MCSTs. For generating k MCSTs, the time complexity is of the order of $\mathcal{O}(k(E \log n))$, where E is the number of edges and n is the number of nodes in the graph. For identifying nodes that are in the denser regions of the graph we compute the clustering coefficient of each node in the k -MCST. The complexity of computing the clustering coefficients, using some good data structures to represent the graphs is, either $\mathcal{O}(m^2/n)$ or $\mathcal{O}(m^{1.48})$ [10] where m is the number of edges in the graph and n is the number of nodes in the graph. Having significantly reduced the number of edges in the k -MCST compared to the original graph, we have made the task of computing clustering coefficients much less compute intensive. In the second phase, as each node is connected to at least k nearest nodes, the time complexity for growing a core and build peripheries for a given query node is $\mathcal{O}(k^{number-of-layers})$. Since we expect to query the k -MCST multiple times, complexity of this repetitive task will be significantly reduced. The number of nearest neighbors of a node in the k -MCST is, on average, equal to $2 * k$, but will be much higher in the complete, unreduced graph. So, the complexity of our algorithm is significantly reduced for the operations of computing clustering coefficients and for repeatedly computing the CP structures from a k -MCST (Fig. 1).

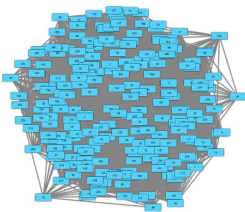


Fig. 1. Original network: 5899 edges

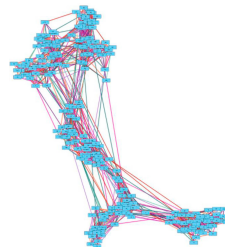


Fig. 2. 7-MCST network: 1645 edges

4 Experiments and Results on Real World Data Sets

4.1 Primary School Data Set

This data set is taken from a study of contact networks from a primary school conducted by SocioPatterns [11]. It describes interactions among students and teachers in a primary school for one day. Nodes represent students and teachers. Edges represent cumulative interaction time spent between two people, measured in seconds. Each node is also labeled by the class and gender of the person. The network has 236 nodes and 5899 labeled edges.

k-MCST Generation: The edge weights here describe the amount of time spent between two people. Since distance or cost for an edge is inverse of the time spent, for our MCST algorithm we replace each edge label with the inverse of the time spent between pairs of people. From this network, we constructed a merged k -MCST graph for various values of k , and the one for $k = 7$ is shown in Fig. 2. The number of edges in this 7-MCST network is only 27.8% of those in the original network. This significantly reduces the complexity of computing the clustering coefficients of nodes and also the CP structures.

CP Network for Node with Highest Clustering Coefficient: Among all the students in the 7-MCST network, student #52 has the highest clustering coefficient (0.92). The core-periphery structure for this student, obtained by our algorithm, is shown in Fig. 6. The periphery thresholds are defined using the reduction factor of $r = 5\%$.

The core for student #52 contains 10 students, (Core(52) is shown in yellow nodes). There are a total of 31 other nodes that have at least one edge incident on at least one of the core nodes. These 31 nodes are tested for placement in the next peripheral layer. The core exhibits a high structural density (0.97) and is surrounded by peripheral layers in decreasing order of structural density.

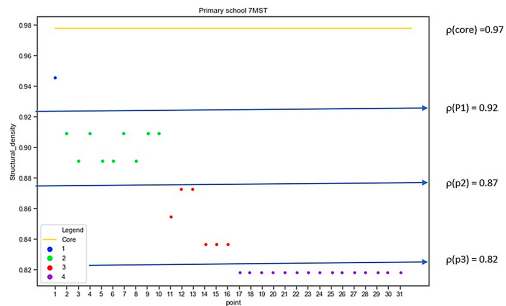


Fig. 3. Density values when each of 31 neighbors is added to the core

Figure 3 shows the structural density of the (core+node) pairs when each of these 31 nodes is tested for placement in the next peripheral layer. It is clear from this plot that the nodes in the outer peripheral layers have decreasing connectivity to the core.

CP Network for a Node with Low Clustering Coefficient: We constructed the core-periphery network for student #125 who has a very low clustering coefficient (0.19) in the 7-MCST network. As this student is very sparsely connected to his immediate neighbors, the structural density of his core social network is very low. The algorithm then finds a large number of other sparsely

connected students who meet this low threshold and become members of his core and periphery sets. That is, his core community consists of a large number of students, each of whom is only lowly connected to others in the core community. This core community has its peripheral layers at decreasing levels of edge densities, and they are also highly populated due to low density thresholds (Figs. 4 and 5).

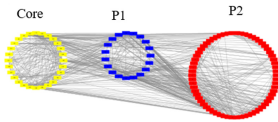


Fig. 4. CP structure of student #125

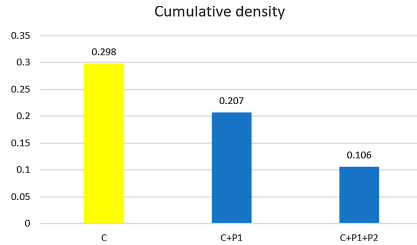


Fig. 5. Cumulative density of CP structure of student #125

This shows that a community for a query node with high clustering coefficient will include fewer but highly connected students, and when the clustering coefficient of the query node is low, we are likely to get a loosely connected and highly populated community.

Comparison with Ground Truth: Figure 6 here shows the CP structure for student #52, with red border drawn around nodes representing students who are from his own class (4A).

All ten students in his core are from his own class, his teacher is in layer P1, and some students from his class occur in peripheral layer P2 and p3. The numbers in Table 1 show the distribution

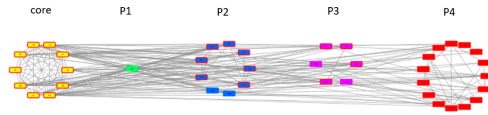


Fig. 6. Red bordered students are in the same class as query node #52 (Color figure online)

of classes the students come from in each layer of the CP structure. This is very much in line with the expectation that students interact more with their own classmates and have fewer interactions with students from other classes. The teachers were also included in the CP structure and the CP structure explains the interaction of the students with their class teacher. The teacher is in the first peripheral layer and this says that the student #52 interacts more with his friends and less with his teacher.

Similarly, as we analyze the class distribution of students in the CP structure of student #125 shown in Fig. 7 and Table 2, it is seen that the student’s core includes students mostly his own class (1A), and many other students from 2nd, 3rd and 4th grades appear in his peripheral layers.

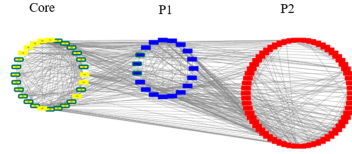


Fig. 7. Green-bordered students are in the same class as student #125 (Color figure online)

Though the strength of connectivity with each student is low, the structural density of his core and peripheries is also lower, but he has his own CP structure in the sense of communities for other students.

Table 1. Distribution of nodes in CP structure of student #52

Node class	Core	P1	P2	P3	P4
4A	10	0	7	4	0
4B	0	0	2	1	4
5A	0	0	0	0	2
5B	0	0	0	1	5
3B	0	0	0	0	2
2B	0	0	0	0	2
Teachers	0	1	0	0	0

Table 2. Distribution of nodes in CP structure of student #125

Node class	Core	P1	P2
1A	24	2	0
1B	2	9	12
2A	2	3	13
2B	4	7	13
3A	0	1	7
4A	0	0	2
4B	1	0	13
Teachers	1	1	2

Our algorithm is thus able to construct meaningful CP structure for any node, from denser or sparser parts of the graph. This is in contrast to traditional algorithms that find their cores in only the dense parts of the graphs.

4.2 Airport Network

This data set describes passenger traffic through a network of airports in year 2010. The data is taken from the Bureau of Transportation Statistics (BTS).

The data set includes 1574 airports as nodes and 28236 edges which are labeled with the number of passengers traveling between pairs of airports. In this data set, each edge represents the number of passengers who traveled between two airports. The affinity between two airports is determined by the number of passengers traveling between them.

A 5-MCST network retains 5396 edges, extracted from the original data set containing 28236 edges. Using our algorithm we then constructed the core-periphery network for the CVG airport as the query node. CVG airport is an international airport located in Hebron, Kentucky. The cities in the core set when CVG airport is used as the query are shown in Table 3. The set of cities in the core for the same query airport CVG has fewer members when we use the 3-MCST network. This is shown in Table 4. The effect of k on the CP structures has been described in the next subsection. The core and periphery cities are plotted on the map in Fig. 8 and its properties are illustrated in Table 5. The yellow, pink and blue nodes represent the core, periphery-1 and periphery-2 respectively.

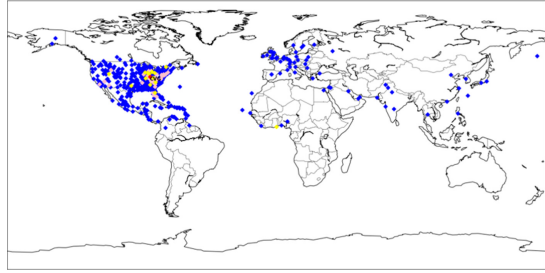


Fig. 8. Core and periphery cities of CVG airport (Color figure online)

Table 3. Core cities for CVG airport on 5-MCST network

Montreal, Canada	Chicago, IL
Atlanta, GA	Cincinnati, OH
Cleveland, OH	Charlotte, NC
Knoxville, TN	Asheville, NC
South Bend, IN	Accra, Ghana
Dallas/Fort Worth, TX	Evansville, IN
Fort Wayne, IN	Ashland, WV
Salt Lake City, UT	Detroit, MI
Orlando, Florida	

Table 4. Core cities of CVG airport on 3-MCST network

Montreal, Canada	Chicago, IL
Atlanta, GA	Cincinnati, OH
Cleveland, OH	Charlotte, NC
Knoxville, TN	Asheville, NC

Table 5. Properties of each layer in core-periphery network of CVG airport

Layer	#airports	$\rho(\text{Layer})$	Avg. # passengers/day (mean edge weight)	Avg. # passengers/day from core to each periphery layer
Core	16	0.35	633	–
P1	30	0.03	793	713
P2	408	0.007	167	267

4.3 Effect of Varying the Parameter k

The number of MCSTs included in the reduced graph, k , has significant impact on the nature of CP structure obtained. Tables 6, and 7 below show the variation in the characteristics of CP structures obtained for $k = 3, 5, 7,$ and 9 for both our data sets. An increase in the value of k results in more edges getting included in the k -MCST graph. This causes cores to have higher edge densities, and some nodes that were in peripheries for lower k 's increase their connectivity to the cores and gets pulled into the core sets. We see that the choice of k affects the sizes of the core sets and also the average affinity values within the nodes of the core. Therefore, a choice of k must be made for each CP structure to identify the cores of desired size and optimal average affinity value. A solution for this is to maintain k -MCSTs for multiple values of k , process the query on all of them, and choose the CP structure with optimal affinity values.

Table 6. Primary school data set: variation of CP structure of student 52

k	#edges(k-MCST)	$\rho(\text{core})$	Avg core edge wght	Core list
3	705	0.57	14.11	34, 38, 47, 51, 52, 119, 176, 196
5	1175	0.82	12.6	34, 38, 47, 51, 52, 119, 176, 196
7	1645	0.93	11.97	34, 37, 38, 47, 49, 51, 52, 119, 176, 196
9	2115	0.95	9.33	34, 37, 38, 46, 47, 49, 50, 51, 52, 53, 54, 93, 119, 176, 185, 196

4.4 Validation of Use of k -MCSTs for CP Structures

One assumption we have made is that the CP structures discovered from the reduced k -MCST graphs are as valid as those discovered from the complete graphs. To validate this assumption we apply our algorithm to the original graph and also to its reduced k -MCST graph and compare the resulting CP structures.

Table 7. Airport data set: variation in CP structure

k	#edges in kMCST	$\rho(\text{core})$	Avg # passengers/day in core cities	# core nodes
3	3822	0.36	529	8
5	5396	0.35	686	15
7	6675	0.35	1084	27
9	7755	0.35	893	38

We compute two different types of ratios from average affinity values corresponding to the edges in: (i) Core vs. (Core + P1), (ii) Core vs. (Core + P1

+ P2), and (iii) Core vs. (Core + P1 + P2 + P3), etc. The ratios show how much relatively weaker the peripheries are compared to the core’s own affinities. The precise descriptions of the two ratios used are given in the following paragraphs. These ratios are expected to decline as more and more peripheries are included in the denominator. The individual ratio values represent the relative distribution of the affinities across a core and its peripheries. The trend of these ratios, along increasing inclusion of peripheries, reflects the way the peripheral strengths decline as we move away from the core.

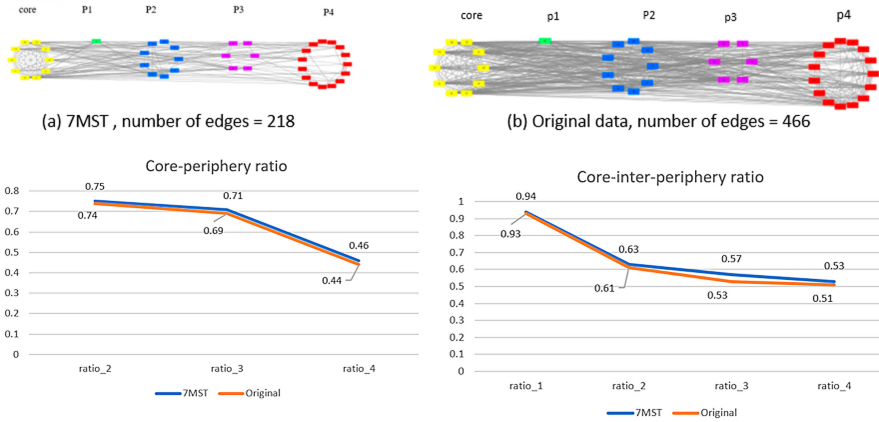


Fig. 9. Validation of core-periphery structure of student 52 (primary school)

We run our algorithm on the original complete graph and also on the reduced k -MCST graph to construct CP structures. If these ratios show very similar values and very similar declining trend, then we can infer that the average affinities of edges included in cores and peripheries are also very similar. Similar values and trends of the ratios will, therefore, show that the assumption of working with the k -MCSTs is justifiable. The names and precise descriptions of the two ratios that we have used to characterize the CP-structures are: (i) Core-periphery ratio, and (ii) Core-inter-periphery ratio. Given a core and its n periphery layers, we define core-periphery ratio up to a particular periphery level k as: $\text{Core-periphery_ratio}_k = \frac{W_{in}(Core)}{W_{in}(Core) + \sum_{k=1}^n W_{in}(Periphery_k)}$ where $1 \leq k \leq n$, and $W_{in}(set\ of\ edges)$ is the sum of the weights in *set of edges*. That is, we sum the edge weights for all edges included exclusively in the core, and then separately in individual peripheral layers. The core-periphery ratio is defined as the ratio of sum of core’s internal edge weights to the core and peripheries’ internal edge weights only. This measure depicts the cohesiveness of core and peripheries as new outer peripheral layers are added. As seen in Fig. 9 and Fig. 10 CP structures constructed from the k -MCST and the complete graph show very similar values and decreasing trends. For the school data set the individual ratios are almost identical. For the airport data, the trend is the same but the individual values differ

because in this case, during the core generation phase, some very weak edges got included in the core. We can thus see that the core-periphery networks generated using our approach are not significantly affected by dropping the weaker edges and retaining only the k -MCST graphs.

Now we consider the second metric for characterizing the CP structures. Core exhibits properties of cohesiveness by having high intra-core affinity values and low affinity values for the edges that connect the core nodes to peripheral nodes. To study the relative weights of intra-core and inter-core-periphery edges we introduce the measure called core-inter-periphery ratio. This is computed as:

$$\text{Core-inter-periphery_ratio}_k = \frac{W_{in}(Core)}{W_{in}(Core) + \sum_{k=1}^n \text{Interedge}(Core, Periphery_k)}$$

where $1 \leq k \leq n$, $W_{in}(set\ of\ edges)$ is the sum of the weights in the *set of edges*, and $\text{Interedge}(C, P_j)$ is the sum of edge-weights connecting Core to P_j . As periphery layers are added, we compute the successive core-inter-periphery ratio and examine its trend. As seen in Figs. 9 and 11, the CP structures obtained using k -MCST and the complete graph show very similar values and similar decreasing trends.

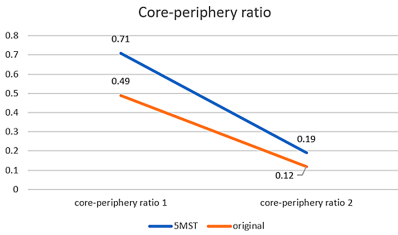


Fig. 10. Core-periphery ratio validation

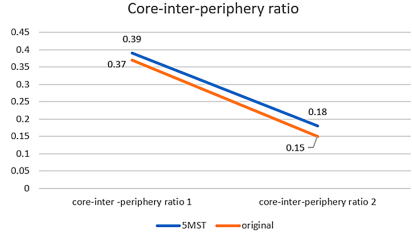


Fig. 11. Core-inter periphery ratio validation.

This shows that the CP structures obtained from the k -MCSTs have essentially the same characteristics as those that would have been obtained from the complete graph. Details of this algorithm can be found in [14].

5 Conclusion

In this paper we have presented a new algorithm for constructing core-periphery structures for individual query nodes from an edge-weighted undirected graph. Our algorithm is tailored for applications where we may want to repeatedly query a very large graph for CP structures surrounding individual nodes. Our algorithm reduces the computational complexity for each CP query by significantly reducing the size of the graph to be processed, by retaining only the strongest affinity edges. We have demonstrated the effectiveness of our algorithm by executing it on a social network and on an airport passenger traffic network. The ground truths from these domains validate the CP structures produced by our algorithm. Using two different validation measures, we have shown

that CP structures obtained from k -MCSTs and the original complete graphs shows very similar trends in terms of structural and edge weight densities among the cores and their peripheries.

References

1. Luo, F., et al.: Core and periphery structures in protein interaction networks. In: BMC Bioinformatics, vol. 10, no. 4. BioMed Central (2009)
2. Borgatti, S.P., Everett, M.G.: Models of core/periphery structures. Soc. Netw. **21**(4), 375–395 (2000)
3. Sardana, D., Raj, B.: Core periphery structures in weighted graphs using greedy growth. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI). IEEE (2016)
4. Yang, J., Leskovec, J.: Overlapping communities explain core-periphery organization of networks. Proc. IEEE **102**(12), 1892–1902 (2014)
5. Zhong, C., Miao, D., Wang, R.: A graph-theoretical clustering method based on two rounds of minimum spanning trees. Pattern Recogn. **43**(3), 752–766 (2010)
6. Zhong, C., Miao, D., Fránti, P.: Minimum spanning tree based split-and-merge: a hierarchical clustering method. Inf. Sci. **181**(16), 3397–3410 (2011)
7. Li, Y.: K-edge connected neighborhood graph for geodesic distance estimation and nonlinear data projection. In: Proceedings of the 17th International Conference on Pattern Recognition, 2004, ICPR 2004, vol. 1. IEEE (2004)
8. Silva, D., Rosa, M., Ma, H., Zeng, A.-P.: Centrality, network capacity, and modularity as parameters to analyze the core-periphery structure in metabolic networks. Proc. IEEE **96**(8), 1411–1420 (2008)
9. Della Rossa, F., Dercole, F., Piccardi, C.: Profiling core-periphery network structure by random walkers. Sci Rep. **3**(1), 1–8 (2013)
10. Schank, T., Wagner, D.: Approximating clustering coefficient and transitivity. J. Graph Algorithms Appl. **9**(2), 265–275 (2005)
11. Stehlé, J., et al.: High-resolution measurements of face-to-face contact patterns in a primary school. PloS One **6**(8), e23176 (2011)
12. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E **69**(6), 066133 (2004)
13. Ailem, M., Role, F., Nadif, M.: Graph modularity maximization as an effective method for co-clustering text data. Knowl.-Based Syst. **109**, 160–173 (2016)
14. Polepalli, S.: Discovery of core-periphery structures in networks using k -MSTs. Diss. University of Cincinnati (2019)