

# A label-switching algorithm for fast core-periphery identification

Eric Yanchenko and Srijan Sengupta

**Abstract**—Core-periphery (CP) structure is frequently observed in networks where the nodes form two distinct groups: a small, densely interconnected core and a sparse periphery. Borgatti and Everett (2000) proposed one of the most popular methods to identify and quantify CP structure by comparing the observed network with an “ideal” CP structure. While this metric has been widely used, an improved algorithm is still needed. In this work, we detail a greedy, label-switching algorithm to identify CP structure that is both fast and accurate. By leveraging a mathematical reformulation of the CP metric, our proposed heuristic offers an order-of-magnitude improvement on the number of operations compared to a naive implementation. We prove that the algorithm converges to a local minimum while consistently yielding solutions within 90% of the global optimum on small toy networks. On synthetic networks, our algorithm exhibits superior classification accuracies and run-times compared to a popular competing method, and the analysis of real-world networks shows that the proposed method can be nearly 400 times faster than the competition.

**Index Terms**—Core-periphery, Graphs, Large networks, Meso-scale structures



## 1 INTRODUCTION

NETWORKS are a simple model for complex relationships with entities or objects represented as *nodes* and their interactions or relationships encoded as *edges*. One observed feature in many networks is *core-periphery* (CP) structure, where nodes form two distinct groups: a densely-connected core and a sparsely-connected periphery [1], [2]. The defining feature separating CP structure from the more well-known assortative mixing is that peripheral nodes are more likely to connect with core nodes than each other. CP structure has been observed across many domains including: airport networks [3], [4], power grid networks [5], economic networks [6], [7], and more. We refer the interested reader to [2] for a more thorough review.

One of the most popular approaches to detect and quantify CP structure is the Borgatti and Everett (BE) metric which finds the correlation between the observed network and an “ideal” CP structure [8]. There are multiple implementations available, including the UCInet software [9] and `cpnet` Python package. But as networks continue to increase in size, it is important for applied researchers to have efficient algorithms to quickly and accurately detect this CP structure.

In this work, we detail a greedy, label-switching algorithm to identify CP structure in networks. The algorithm builds off that of [10], but we expound on computational improvements both theoretically and empirically. Specifically, we exploit a mathematical reformulation of the BE metric which yields an order-of-magnitude speed-up compared to a naive implementation. Moreover, our algorithm is guaranteed to converge to a local minimum, and we

find that it typically yields results within 90% of the global optimum on small networks. Simulation studies and real-data analysis find the algorithm to be significantly faster and more accurate than a competing method. Indeed, our algorithm is typically one to two orders of magnitude faster, and on one network, is nearly 400 times faster.

In related work, there have been several notable developments in fast algorithms for statistical network analysis, particularly for community detection, model fitting, and two-sample testing. Ref. [11] proposed a fast pseudo-likelihood method for community detection, which was subsequently refined by [12]. Ref. [13] introduced a distributed algorithm for large-scale block models with a grouped community structure. Divide-and-conquer strategies have also gained attention as scalable alternatives to global community detection in large networks [14], [15], [16]. More recently, subsampling-based approaches have been developed for computationally efficient network analysis. Ref. [17] proposed a predictive subsampling method for fast community detection, while [18] introduced a subsampling-based framework for scalable estimation and two-sample testing. In parallel, [19] developed a subsampling-based approach for network cross-validation and model selection. Lastly, [10] and [20] develop divide-and-conquer algorithms for fast CP identification in large networks. Here, we leverage the base algorithm of these works and further detail the computational improvements.

The remainder of this paper is organized as follows. In Section 2, we describe the BE metric while in Section 3 we detail a computationally efficient algorithm and examine some of its properties. Experiments on simulated and real-world networks are the subjects of Sections 4 and 5, respectively, and we share concluding thoughts in Section 6.

- E. Yanchenko is an assistant professor in the Global Connectivity Program at Akita International University, Akita, Japan  
E-mail: eyanchenko@aiu.ac.jp
- S. Sengupta is an associate professor in the Department of Statistics at North Carolina State University, Raleigh, NC

Manuscript received April 19, 2005; revised August 26, 2015.

## 2 CORE-PERIPHERY METRIC

Consider a network with  $n$  nodes and let  $\mathbf{A}$  be the corresponding  $n \times n$  adjacency matrix where  $A_{ij} = 1$  if nodes  $i$  and  $j$  have an edge, and 0 otherwise, and  $A_{ij} \mid P_{ij} \stackrel{\text{ind.}}{\sim} \text{Bernoulli}(P_{ij})$  for  $0 \leq i < j \leq n$ . We write  $\mathbf{A} \sim \mathbf{P}$  as shorthand for this model. For simplicity, we only consider unweighted, undirected networks with no self-loops. We also define  $\mathbf{c} \in \{0, 1\}^n$  as the CP assignment vector where  $c_i = 1$  if node  $i$  is assigned to the core, and 0 if assigned to the periphery where  $k = \sum_{i=1}^n c_i$  is the size of the core. For a given  $\mathbf{c}$ , let  $\Delta(\mathbf{c}) \equiv \Delta$  be an  $n \times n$  matrix such that  $\Delta_{ij} = c_i + c_j - c_i c_j$ , i.e.,  $\Delta_{ij} = 1$  if node  $i$  or  $j$  is assigned to the core, and 0 otherwise. Then  $\Delta$  represents an “ideal” CP structure where each core node is connected to all other nodes, but periphery nodes have no connections among themselves.

To quantify the CP structure, [8] define

$$T(\mathbf{A}, \mathbf{c}) = \text{Cor}(\mathbf{A}, \Delta), \quad (1)$$

i.e., the Pearson correlation between the upper-triangles of  $\mathbf{A}$  and  $\Delta$ . We can re-write (1) as

$$\begin{aligned} T(\mathbf{A}, \mathbf{c}) &= \frac{\sum_{i < j} (A_{ij} - \bar{A})(\Delta_{ij} - \bar{\Delta})}{\frac{1}{2}n(n-1)\{\bar{A}(1 - \bar{A})\bar{\Delta}(1 - \bar{\Delta})\}^{1/2}} \\ &= \frac{\sum_{i < j} A_{ij}\Delta_{ij} - \frac{1}{2}n(n-1)\bar{A}\bar{\Delta}}{\frac{1}{2}n(n-1)\{\bar{A}(1 - \bar{A})\bar{\Delta}(1 - \bar{\Delta})\}^{1/2}} \end{aligned} \quad (2)$$

where  $\bar{X}$  is the entry-wise mean of the upper-triangle of matrix  $X$ . The form in (2) is more amenable to statistical analysis and will be leveraged in the remainder of this work. Lastly, the labels  $\mathbf{c}$  are typically unknown so we must find the optimal labels, i.e.,

$$\hat{\mathbf{c}} = \arg \max_{\mathbf{c}} \{T(\mathbf{A}, \mathbf{c})\}. \quad (3)$$

The space of all possible solutions,  $\{0, 1\}^n$ , has  $\mathcal{O}(2^n)$  elements, meaning that an exhaustive search to find the globally optimal solution is not feasible for even moderately sized networks. Thus, an optimization algorithm must be employed to find an approximation of the global optimum. The remainder of this paper is devoted to such an algorithm that is both computationally efficient and empirically accurate.

## 3 ALGORITHM

### 3.1 Label-switching algorithm

We first detail an efficient algorithm to approximate  $\hat{\mathbf{c}}$  in (3), building off that of [10]. After randomly initializing CP labels, one at a time and for each node  $i$ , we swap the label of the node and compute (2) with these new labels. The new label is kept if the test statistic is larger than before, otherwise the original label remains. This process repeats until the labels are unchanged for an entire cycle through all  $n$  nodes. The steps are outlined in Algorithm 1.

---

### Algorithm 1 Label-switching for core-periphery detection

---

**Result:** Core-periphery labels  $\mathbf{c} \in \{0, 1\}^n$

**Input:** adjacency matrix  $\mathbf{A} \in \{0, 1\}^{n \times n}$

Initialize labels  $\mathbf{c}$

$run = 1$

**while**  $run > 0$  **do**

    Set  $run = 0$

    Randomly order nodes

**for**  $i$  in  $1 : n$  **do**

$\mathbf{c}' = \mathbf{c}, c'_i = 1 - c_i$

**if**  $T(\mathbf{A}, \mathbf{c}') > T(\mathbf{A}, \mathbf{c})$  **then**

$\mathbf{c} \leftarrow \mathbf{c}'$

$run = 1$

**end**

**end**

**end**

---

### 3.2 Connection with Kernighan-Lin algorithm

There are some clear connections between Algorithm 1 and the well-known Kernighan-Lin heuristic for graph partitioning [21]. In that context, we assume the network is partitioned into two sets of nodes, and the algorithm swaps node pairs which minimize some cost function (the number of inter-group edges minus intra-group edges). Once these nodes have been swapped, their group is fixed for the remaining pass of the algorithm. Similarly, in Algorithm 1, we swap the group (core or periphery) of a node and keep the swap if it increases our objective function,  $T(\mathbf{A}, \mathbf{c})$ . After this, the node's assignment is fixed for the remainder of that pass (for loop). One key difference is that the size of the groups is not fixed in Algorithm 1 but is in the Kernighan-Lin algorithm.

### 3.3 Computational improvement

The main computational bottleneck in Algorithm 1 comes from computing  $T(\mathbf{A}, \mathbf{c}')$ . Ostensibly, this requires  $\mathcal{O}(n^2)$  operations since we must compute  $M(\mathbf{c}) = \sum_{i < j} A_{ij}\Delta_{ij}(\mathbf{c})$  and  $\bar{A} = \sum_{i < j} A_{ij} / \{\frac{1}{2}n(n-1)\}$ . We can leverage the reformulation of the BE metric from (2), however, to improve the computational efficiency of Algorithm 1 from that of the original algorithm in [10]. In particular,  $T(\mathbf{A}, \mathbf{c}')$  can be efficiently updated since  $\mathbf{c}'$  differs from the current CP labels by only a single entry. While the following approach has been used in previous works (e.g., [20]), the details of this computational improvement have yet to be formally described and thoroughly investigated.

At initialization, there are no shortcuts and the objective function  $T(\mathbf{A}, \mathbf{c})$  must be computed as in (2), which takes  $\mathcal{O}(n^2)$  operations. Now, let  $(\mathbf{c}')^{(i)}$  be the proposed CP labels where  $(c')_i^{(i)} = 1 - c_i$  and  $(c')_j^{(i)} = c_j$  for  $j \neq i$ , i.e., the labels differ only at node  $i$ . It is clear that we only need to update  $\bar{A}$  and  $M(\mathbf{c})$  to compute  $T(\mathbf{A}, (\mathbf{c}')^{(i)})$ . We know that  $\bar{A}(\mathbf{c}) = \{\frac{1}{2}k(k-1) + k(n-k)\} / \{\frac{1}{2}n(n-1)\}$  where  $k = \sum_{j=1}^n c_j$ . If  $k'$  is the size of the core for labels  $(\mathbf{c}')^{(i)}$ , then  $k' = k + 1$  if  $c_i = 0$ , and  $k' = k - 1$  if  $c_i = 1$ . Thus  $\bar{A}((\mathbf{c}')^{(i)}) = \{\frac{1}{2}k'(k'-1) + k'(n-k')\} / \{\frac{1}{2}n(n-1)\}$  can be

computed in  $\mathcal{O}(1)$ . The key step is computing  $M((c')^{(i)})$  by using  $M(c)$ . Notice that

$$\begin{aligned}
 M((c')^{(i)}) &= \sum_{j=1}^n \sum_{k=1}^{j-1} A_{jk} \Delta_{jk}((c')^{(i)}) \\
 &= \sum_{j \neq i}^n \sum_{k=1}^{j-1} A_{jk} \Delta_{jk}((c')^{(i)}) + \sum_{k=1}^{i-1} A_{ik} \Delta_{ik}((c')^{(i)}) \\
 &= \sum_{j \neq i}^n \sum_{k=1}^{j-1} A_{jk} \Delta_{jk}(c) + \sum_{k=1}^{i-1} A_{ik} \Delta_{ik}((c')^{(i)}) \\
 &= \left( M(c) - \sum_{k=1}^{i-1} A_{ik} \Delta_{ik}(c) \right) + \sum_{k=1}^{i-1} A_{ik} \Delta_{ik}((c')^{(i)}) \\
 &= M(c) + \sum_{k=1}^{i-1} A_{ik} \{ \Delta_{ik}((c')^{(i)}) - \Delta_{ik}(c) \}. \quad (4)
 \end{aligned}$$

Since we have already computed  $M(c)$ , the only term we need to compute in (4) is the final sum, which requires  $\mathcal{O}(n)$  operations. Thus, while the initial evaluation of  $T(\mathbf{A}, c)$  is  $\mathcal{O}(n^2)$ , we can update its value in  $\mathcal{O}(n)$ . In Section 6, we discuss how this could be improved even further using an edge list instead of adjacency matrix.

### 3.4 Time complexity

We precisely characterize the complexity of one pass of the modified algorithm. Since there are  $n$  nodes in the network, initializing the CP labels is  $\mathcal{O}(n)$ , as is randomly ordering the nodes. Computing  $T(\mathbf{A}, c)$  is  $\mathcal{O}(n^2)$ , but  $T(\mathbf{A}, c')$  can be updated with only  $\mathcal{O}(n)$  operations. This must be computed for each node such that one pass of the for loop takes  $\mathcal{O}(n^2)$  operations. Note that if we re-computed  $T(\mathbf{A}, c')$  from scratch at every iteration, then the pass would require  $\mathcal{O}(n^3)$  operations. Thus, our improvement yields an order-of-magnitude computational speed-up.

### 3.5 Convergence guarantees

In this sub-section, we discuss some of the theoretical properties of the proposed algorithm. Algorithm 1 is a greedy algorithm in the sense that it makes the locally optimal decision. Ideally, we hope that this algorithm adequately approximates the global optimum in (3).

We first prove that Algorithm 1 is guaranteed to converge to a *local* optimum value. To do so, we first must define what “local” means. Let  $N(c)$  be a neighborhood of label  $c$  defined as

$$N(c) = \left\{ c' : \sum_{i=1}^n \mathbb{I}(c'_i \neq c_i) = 1 \right\}.$$

where  $\mathbb{I}(\cdot)$  is the indicator function. In words, this is the set of all CP labels that differ from  $c$  by only one node assignment. In the following proposition, we show that Algorithm 1 converges to a local optimum by this definition.

**PROPOSITION 1.** *Let  $\hat{c}_1$  be the labels returned by Algorithm 1. Then for all  $c \in N(\hat{c}_1)$ ,  $T(\mathbf{A}, \hat{c}_1) \geq T(\mathbf{A}, c)$ , i.e., Algorithm 1 converges to a local optimum.*

We give a brief proof in Appendix A. Proposition 1 ensures that the labels returned by Algorithm 1 yield a local maximum of  $T(\mathbf{A}, c)$ . We would like to have some guarantee, however, of how close this result is to the *global* maximum. In many proofs that an algorithm achieves the global maximum, the *sub-modularity* of the objective function is required, e.g., [22]. Let  $2^{\{1, \dots, n\}} := \Omega$  denote the power set of  $\{1, \dots, n\}$ , i.e., all possible subsets of  $\{1, \dots, n\}$ . Then a function  $f : \Omega \rightarrow \mathbb{R}$  is said to be sub-modular if for all  $S, T \in \Omega$ ,

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T).$$

In the context of the BE metric, there is a one-to-one correspondence between some set of nodes  $S$  and CP label  $c$  if  $v \in S \iff c_v = 1$  and  $v \notin S \iff c_v = 0$ .

Unfortunately, the BE metric is not sub-modular. First, if  $S \cap T = \emptyset$  or  $S \cup T = \{1, \dots, n\}$ , then  $T(\mathbf{A}, c)$  is not well-defined because  $\bar{\Delta} = 0$  or  $\bar{\Delta} = 1$ , respectively, causing the denominator to be zero. Furthermore, consider the following example where  $n = 4$ , and

$$\mathbf{A} = \begin{pmatrix} - & 1 & 0 & 1 \\ & - & 1 & 0 \\ & & - & 1 \\ & & & - \end{pmatrix}.$$

Let  $S = \{2, 3\}$ ,  $T = \{3, 4\}$  and such that  $S \cup T = \{2, 3, 4\}$  and  $S \cap T = \{3\}$ . Define  $c_S, c_T, c_U$  and  $c_\cap$  as the CP label analogue of  $S, T, S \cup T$  and  $S \cap T$ , respectively. Then

$$T(\mathbf{A}, c_S) + T(\mathbf{A}, c_T) = -0.63 < T(\mathbf{A}, c_U) + T(\mathbf{A}, c_\cap) = 0$$

such that sub-modularity is violated. Therefore, it is not straightforward to theoretically show when (or even if) Algorithm 1 achieves a global optimum.

For small  $n$ , however, we can empirically compare Algorithm 1 to the global optimum by using a brute force approach. Specifically, we let  $n = 20$  and generate Erdos-Renyi (ER) networks with  $P_{ij} = p$  for all  $i, j$  [23]. We use ER networks because they do not have an intrinsic CP structure, making the detection task more difficult. Since  $n = 20$ , there are  $2^{20} \approx 10^6$  possible CP labels and we evaluate each one to find  $\hat{c} = \arg \max_c \{T(\mathbf{A}, c)\}$ , i.e., the global optimum. We vary  $p = 0.05, 0.10, \dots, 0.95$  and find  $T(\mathbf{A}, \hat{c}_1)/T(\mathbf{A}, \hat{c})$ , i.e., the ratio of the optimum value returned by Algorithm 1 to the true global optimum. In Figure 1, we report the average proportion over 100 Monte Carlo (MC) replicates. In general, the labels returned by Algorithm 1 are within about 90% of the global optimum. As the network density increases, however, the performance slightly decreases. This result is not too concerning as most real-world networks are relatively sparse, and in these regimes (small  $p$ ), the proposed algorithm performs best. Thus, these results show that the proposed method yields CP labels close to the global optimum.

## 4 SIMULATION STUDY

In this section, we compare Algorithm 1 with an existing algorithm to identify CP structure on synthetic networks. Specifically, `cpnet` is a popular Python package for various network tasks related to CP structure and can be used in R

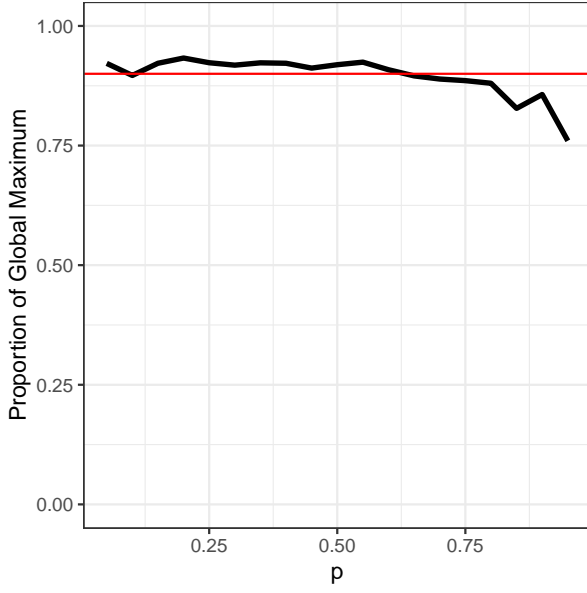


Fig. 1: Ratio of the maximum value of the Borgatti and Everett metric returned by Algorithm 1 with the true global optimum for Erdos-Renyi networks with different values of  $p$ . The red horizontal line denotes 90%.

with the `reticulate` package [24]. This package also uses a Kernighan-Lin-type algorithm to optimize the BE metric.

We generate networks with a known CP structure to compare the labels returned by the algorithms with the ground-truth. In the following sub-sections, we describe the data-generating models, but for now, let  $c^*$  be the true labels and  $\hat{c}$  be the estimated labels returned by one of the algorithms. Then the classification accuracy can be defined as

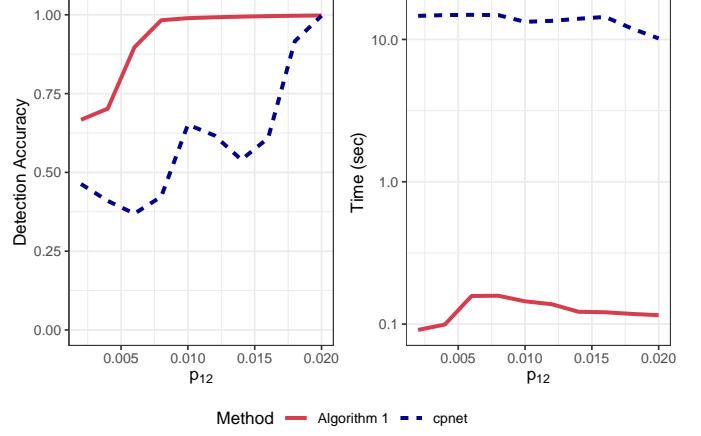
$$\frac{1}{n} \sum_{i=1}^n \mathbb{I}(\hat{c}_i = c_i^*).$$

We report the classification accuracy and run-time for each algorithm. All experiments are carried out on a 2024 Mac Mini with 16 GB of memory, and the results are averaged over 100 MC samples. The R code to implement Algorithm 1, as well as replicate the following simulation study, is available on the author's GitHub: <https://github.com/eyanchenko/fastCP>.

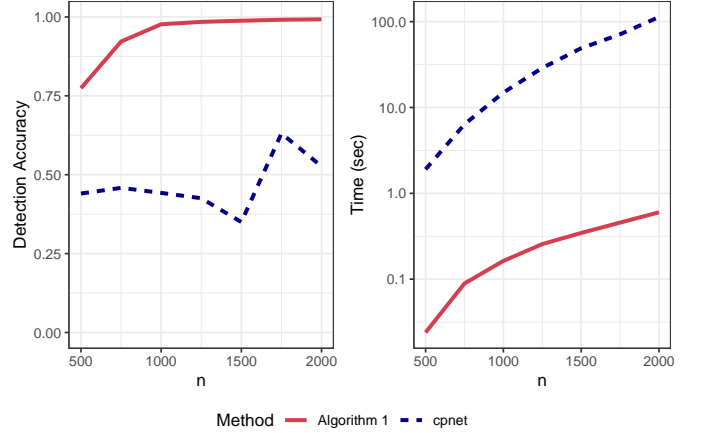
#### 4.1 Stochastic Block Model CP structure

First, we generate networks according to a 2-block stochastic block model (SBM) [25]. We let  $P_{ij} = p_{c_i^*, c_j^*}$  for  $i, j \in \{1, 2\}$  with  $p_{11} > p_{12} = p_{21} > p_{22}$  to ensure the networks have a CP structure where we use a slight abuse of notation as now  $c_i^* = 2$  means that node  $i$  is in the periphery. Unless otherwise noted, we fix  $p_{11} = 2p_{12}$ , i.e., the core-core edge probability is twice the core-periphery edge probability, and  $k = 0.1n$ , i.e., 10% of the nodes are in the core.

In the first set of simulations, we fix  $n = 1000$ ,  $p_{22} = n^{-1} = 0.001$  and vary  $p_{12} = 0.002, 0.004, \dots, 0.02$ . As  $p_{12}$  increases, so too does the strength of the CP structure in the network. The results are in Figure 2a. Algorithm 1 has a much larger detection accuracy and is approximately



(a) Accuracy and computing time for varying  $p_{12}$ .



(b) Accuracy and computing time for varying  $n$ .

Fig. 2: Core-periphery identification results on SBM networks.

two orders of magnitude faster than that of `cpnet` for all values of  $p_{12}$ . Next, we fix  $p_{12} = 0.005$  and  $p_{22} = 0.001$  and vary  $n = 500, 750, \dots, 2000$  with results in Figure 2b. Again, we see that Algorithm 1 clearly outperforms `cpnet` with superior accuracy and run-time. It also appears that Algorithm 1 is more stable in as in both settings, the classification accuracy is monotonically increasing with  $p_{12}$  and  $n$ , unlike that of `cpnet`.

#### 4.2 Degree-corrected Block Model CP structure

In the second set of simulations, we generate networks from a degree-corrected block model (DCBM) [26]. Namely, we introduce weight parameters  $\theta = (\theta_1, \dots, \theta_n)^\top$  to allow for degree heterogeneity among nodes, and generate networks with  $P_{ij} = \theta_i \theta_j p_{c_i^*, c_j^*}$ . If  $p_{11} > p_{12} > p_{22}$ , then we expect the resulting networks to exhibit a CP structure. We sample  $\theta_i \stackrel{\text{iid.}}{\sim} \text{Uniform}(0.6, 0.8)$  and fix  $p_{22} = 0.05$ . First, we fix  $n = 1000$  and vary  $p_{12} = 0.05, 0.06, \dots, 0.15$ , with results in Figure 3a. Second, we fix  $p_{12} = 0.10$  and vary  $n = 500, 750, \dots, 2000$ . As in the SBM examples, Algorithm 1 yields greater detection accuracy and is significantly faster across both settings. That being said, the relative accuracy of `cpnet` is slightly improved in the DCBM example.

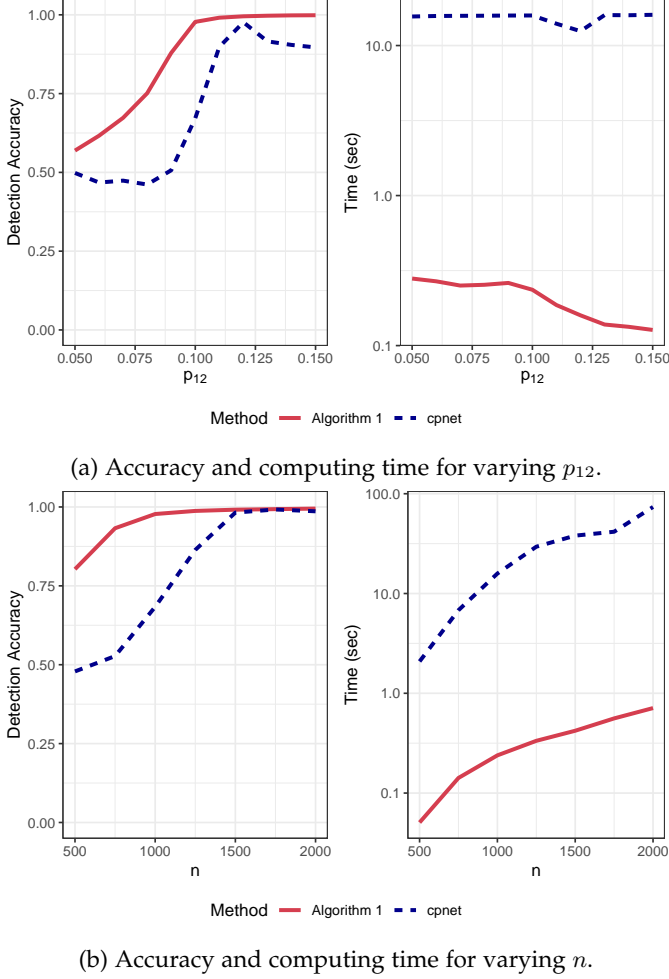


Fig. 3: Core-periphery identification results on DCBM networks.

## 5 REAL-DATA ANALYSIS

Lastly, we apply the proposed algorithm to thirteen real-world networks. We consider data sets of a variety of domains, sizes and densities to ensure the broad applicability of the method, including: *UK Faculty* (number of nodes  $n = 81$ , average edge probability  $\bar{A} = 0.18$ ) [27], *Email* ( $n = 167$ ,  $\bar{A} = 0.23$ ) [28], *British MP* ( $n = 381$ ,  $\bar{A} = 0.08$ ) [29], *Twitter Congress* ( $n = 475$ ,  $\bar{A} = 0.09$ ) [30], *Political Blogs* ( $n = 1224$ ,  $\bar{A} = 0.02$ ) [31], *DBLP* ( $n = 2203$ ,  $\bar{A} = 0.47$ ) [32], [33], *Hospital* ( $n = 75$ ,  $\bar{A} = 0.41$ ) [34], *School* ( $n = 242$ ,  $\bar{A} = 0.29$ ) [35], *Bluetooth* ( $n = 672$ ,  $\bar{A} = 0.10$ ) [36], *Biological 1, 2, 3* ( $n = 2220, 3289, 4040$ ,  $\bar{A} = 0.02, 0.02, 0.01$ , respectively) [37], and *Facebook* ( $n = 4039$ ,  $\bar{A} = 0.10$ ) [38]. All data was retrieved from [39], [40], [41]. For each network, we remove all edge weights, directions, time stamps, self-loops, etc. We also only consider two groups of researchers in the DBLP network as in [42]. We run Algorithm 1 and that of cpnet to identify the optimal CP labels, and record  $T(\mathbf{A}, \hat{\mathbf{c}})$ , the value of the BE metric evaluated at the optimal labels returned by each algorithm, the number of core nodes in these optimal labels and the computing time. Results are reported in Table 1. In this section, we run the cpnet algorithm directly in Python to eliminate any time that might be added from using the `reticulate` package. In general, however, we

Network	Algorithm	$T(\mathbf{A}, \hat{\mathbf{c}})$	$k$	Time
UK Faculty	Alg. 1	0.26	16	0.001
	cpnet	0.25	20	0.364
Email	Alg. 1	0.42	22	0.005
	cpnet	0.42	14	0.481
British MP	Alg. 1	0.25	61	0.021
	cpnet	0.04	1	0.970
Twitter Congress	Alg. 1	0.19	67	0.044
	cpnet	0.18	66	0.092
Political Blogs	Alg. 1	0.21	91	0.247
	cpnet	0.14	323	28.8
DBLP	Alg. 1	0.26	529	0.556
	cpnet	0.20	856	217
Hospital	Alg. 1	0.44	22	0.001
	cpnet	0.44	24	0.017
School	Alg. 1	0.22	52	0.011
	cpnet	0.22	53	0.075
Bluetooth	Alg. 1	0.17	127	0.087
	cpnet	0.17	128	3.17
Biological 1	Alg. 1	0.13	411	0.710
	cpnet	0.03	14	171
Biological 2	Alg. 1	0.13	273	4.11
	cpnet	0.00	3287	546
Biological 3	Alg. 1	0.10	375	6.38
	cpnet	0.07	1472	1007
Facebook	Alg. 1	0.10	299	6.36
	cpnet	0.01	2	1036

TABLE 1: Results of real-data analysis using Algorithm 1 and cpnet.  $T(\mathbf{A}, \hat{\mathbf{c}})$ : value of objective function at optimal labels returned by the algorithm;  $k$ : number of nodes assigned to the core in the optimal labels; Time: computing time (seconds).

found virtually no difference in the run-time or performance when using the function in R vs. Python.

Overall, the results demonstrate strong performance of the proposed method. In about half of the networks, the optimal value of the objective function is similar between the two algorithms, but the proposed algorithm is consistently one to two orders of magnitude faster. On the remaining networks (*British MP*, *Political Blogs*, *DBLP*, *Biological 1 - 3*, *Facebook*), Algorithm 1 is not only faster than that of cpnet but also yields noticeably superior labels in terms of the value of  $T(\mathbf{A}, \hat{\mathbf{c}})$ . Moreover, on *DBLP*, Algorithm 1 is almost 400 times faster than the competing method. Additionally, our method appears to converge to a local optimum for all networks, but the cpnet algorithm seems not to in *British MP*, *Biological 2* and *Facebook*. Even after re-running the algorithm on these networks, the result did not improve. In summary, the proposed algorithm yields better approximations for the global maximum at a fraction of the computation time.

## 6 CONCLUSION

In this work, we describe a greedy, label-switching algorithm to quickly identify core-periphery (CP) structure in networks. Our algorithm improves on existing methods by noting the mathematical relationship between the proposed

and current CP labels, allowing for efficient updates of the objective function. Theoretically, we prove the algorithm converges to a local maximum while also finding that it generally comes within 90% of the global optimum on toy networks. Our simulation study and real-data analysis reveal that our algorithm is generally one to two orders of magnitude faster than the competing method, while yielding more accurate CP labels and larger values of the objective function.

There are several interesting avenues for future work. Precise theoretical results surrounding the conditions needed to obtain a global optimum would be worthwhile. Additionally, we focused on algorithmic speed-ups stemming from mathematical properties of the objective function. Considering more efficient network storage and data structures, however, could also improve the efficiency of the algorithm. For example, we currently compute  $\sum_{k=1}^i A_{ik} \Delta_{ik}$  in  $\mathcal{O}(n)$  operations. But on sparse networks where  $E(A_{ij}) = \rho_n P_{ij}$  for some sparsity parameter  $\rho_n \rightarrow 0$ , if we use an edge list, then the sum could be computed in  $\mathcal{O}(n\rho_n)$  operations, leading to potentially large gains.

## APPENDIX A

### PROOF OF PROPOSITION 1.

Let  $\hat{c}_1$  be the labels returned by Algorithm 1. Then for all  $c \in N(\hat{c}_1)$ ,  $T(\mathbf{A}, \hat{c}_1) \geq T(\mathbf{A}, c)$ , i.e., Algorithm 1 converges to a local optimum.

*Proof.* Recall the steps of Algorithm 1. Then the result is immediate. Let  $\hat{c}_1$  be the labels that Algorithm 1 converged to. Assume, for contradiction, that there is some  $c' \in N(\hat{c}_1)$  such that  $T(\mathbf{A}, c') > T(\mathbf{A}, \hat{c}_1)$ . But this is a contradiction because if  $T(\mathbf{A}, c') > T(\mathbf{A}, \hat{c}_1)$ , then Algorithm 1 would have swapped the differing labels such that  $\hat{c}_1 = c'$ . Therefore, it must be that  $T(\mathbf{A}, \hat{c}_1) \geq T(\mathbf{A}, c')$  for any  $c' \in N(\hat{c}_1)$ .  $\square$

## REFERENCES

- [1] P. Csermely, A. London, L.-Y. Wu, and B. Uzzi, "Structure and dynamics of core/periphery networks," *Journal of Complex Networks*, vol. 1, no. 2, pp. 93–123, 10 2013.
- [2] E. Yanchenko and S. Sengupta, "Core-periphery structure in networks: a statistical exposition," *Statistic Surveys*, vol. 17, pp. 42–74, 2023.
- [3] O. Lordan and J. M. Sallan, "Analyzing the multilevel structure of the european airport network," *Chinese Journal of Aeronautics*, vol. 30, no. 2, pp. 554–560, 2017.
- [4] —, "Core and critical cities of global region airport networks," *Physica A: Statistical mechanics and its applications*, vol. 513, pp. 724–733, 2019.
- [5] B. Yang, X. Huang, X. Hu, W. Cheng, Z. Pei, and X. Li, "Optimizing robustness of core-periphery structure in complex networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 12, pp. 3572–3576, 2021.
- [6] P. Krugman, *The self organizing economy*. John Wiley & Sons, 1996.
- [7] J. M. Magone, B. Laffan, and C. Schweiger, *Core-periphery relations in the European Union: Power and conflict in a dualist political economy*. Routledge, 2016.
- [8] S. P. Borgatti and M. G. Everett, "Models of core/periphery structures," *Social Networks*, vol. 21, no. 4, pp. 375–395, 2000.
- [9] M. Borgatti, SP Everett, L. Freeman *et al.*, "Ucinet for windows: software for social network analysis," *Harvard, MA: Analytic Technologies*, 2002.
- [10] E. Yanchenko, "A divide-and-conquer algorithm for core-periphery identification in large networks," *Stat*, p. e475, 2022.
- [11] A. A. Amini, A. Chen, P. J. Bickel, and E. Levina, "Pseudo-likelihood methods for community detection in large sparse networks," *Annals of Statistics*, vol. 41, no. 4, 2013.
- [12] J. Wang, J. Zhang, B. Liu, J. Zhu, and J. Guo, "Fast network community detection with profile-pseudo likelihood methods," *Journal of the American Statistical Association*, vol. 118, no. 542, pp. 1359–1372, 2023. [Online]. Available: <https://doi.org/10.1080/01621459.2021.1996378>
- [13] S. Zhang, R. Song, W. Lu, and J. Zhu, "Distributed community detection in large networks," *Journal of Machine Learning Research*, vol. 24, no. 401, pp. 1–28, 2023. [Online]. Available: <http://jmlr.org/papers/v24/19-590.html>
- [14] S. S. Mukherjee, P. Sarkar, and P. J. Bickel, "Two provably consistent divide-and-conquer clustering algorithms for large networks," *Proceedings of the National Academy of Sciences*, vol. 118, no. 44, p. e2100482118, 2021.
- [15] S. Wu, Z. Li, and X. Zhu, "Distributed community detection for large scale networks using stochastic block model," *arXiv preprint arXiv:2009.11747*, 2020.
- [16] S. Chakrabarty, S. Sengupta, and Y. Chen, "Subsampling based community detection for large networks," *Statistica Sinica*, vol. 35, no. 3, pp. 1–42, 2025.
- [17] S. Bhadra, M. Pensky, and S. Sengupta, "Scalable community detection in massive networks via predictive assignment," *arXiv preprint arXiv:2503.16730*, 2025.
- [18] K. Chakraborty, S. Sengupta, and Y. Chen, "Scalable estimation and two-sample testing for large networks via subsampling," *Journal of Computational and Graphical Statistics*, pp. 1–13, 2025.
- [19] S. Chakrabarty, S. Sengupta, and Y. Chen, "Network cross-validation and model selection via subsampling," *arXiv preprint arXiv:2504.06903*, 2025.
- [20] E. Yanchenko, "Graph sub-sampling for divide-and-conquer algorithms in large networks," *Statistics and Computing*, vol. 35, no. 4, p. 87, 2025.
- [21] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.
- [22] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 137–146.
- [23] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.
- [24] K. Ushey, J. Allaire, and Y. Tang, *reticulate: Interface to 'Python'*, 2025, r package version 1.41.0.9001.
- [25] P. Holland, K. Laskey, and S. Leinhardt, "Stochastic blockmodels: first steps," *Social Networks*, vol. 5, pp. 109–137, 1983.
- [26] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, p. 016107, 2011.
- [27] T. Nepusz, A. Petróczy, L. Négyessy, and F. Bazsó, "Fuzzy communities and the concept of bridgeness in complex networks," *Physical Review E*, vol. 77, no. 1, p. 016107, 2008.
- [28] R. Michalski, S. Palus, and P. Kazienko, "Matching organizational structure and social network extracted from email communication," in *Business Information Systems: 14th International Conference, BIS 2011, Poznań, Poland, June 15-17, 2011. Proceedings 14*. Springer, 2011, pp. 197–206.
- [29] D. Greene and P. Cunningham, "Producing a unified graph representation from multiple social network views," in *Proceedings of the 5th annual ACM web science conference*, 2013, pp. 118–121.
- [30] C. G. Fink, K. Fullin, G. Gutierrez, N. Omodt, S. Zinnecker, G. Sprint, and S. McCulloch, "A centrality measure for quantifying spread on weighted, directed networks," *Physica A*, 2023.
- [31] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*, 2005, pp. 36–43.
- [32] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "Graph-based consensus maximization among multiple supervised and unsupervised models," *Advances in neural information processing systems*, vol. 22, 2009.
- [33] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 570–586.
- [34] P. Vanhems, A. Barrat, C. Cattuto, J.-F. Pinton, N. Khanafer, C. Regis, B.-A. Kim, B. Comte, and N. Voirin, "Estimating potential

- infection transmission routes in hospital wards using wearable proximity sensors," *PloS one*, vol. 8, p. e73970, 09 2013.
- [35] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina *et al.*, "High-resolution measurements of face-to-face contact patterns in a primary school," *PloS one*, vol. 6, no. 8, p. e23176, 2011.
- [36] P. Sapiezynski, A. Stopczynski, D. D. Lassen, and S. Lehmann, "Interaction data from the copenhagen networks study," *Scientific Data*, vol. 6, no. 1, p. 315, 2019.
- [37] A. Cho, J. Shin, S. Hwang, C. Kim, H. Shim, H. Kim, H. Kim, and I. Lee, "Wormnet v3: a network-assisted hypothesis-generating server for *caenorhabditis elegans*," *Nucleic acids research*, vol. 42, no. W1, pp. W76–W82, 2014.
- [38] J. Leskovec and J. Mcauley, "Learning to discover social circles in ego networks," *Advances in neural information processing systems*, vol. 25, 2012.
- [39] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," <http://snap.stanford.edu/data>, Jun. 2014.
- [40] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: <https://networkrepository.com>
- [41] G. Csárdi, T. Nepusz, V. Traag, S. Horvát, F. Zanini, D. Noom, and K. Müller, *igraph: Network Analysis and Visualization in R*, 2025, r package version 2.1.4.
- [42] S. Sengupta and Y. Chen, "A block model for node popularity in networks with community structure," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 80, no. 2, pp. 365–386, 2018.

**Eric Yanchenko** received a BS in Mathematics and Physics from The Ohio State University, Columbus, Ohio, USA, and a PhD in statistics from North Carolina State University, Raleigh, North Carolina, USA. He is currently an assistant professor at Akita International University in Akita, Japan. Previously, he worked as a Japan Society for Promotion of Science (JSPS) Short-term Fellow at Tokyo Institute of Technology, Tokyo, Japan. His research interests include hypothesis testing for meso-scale structures on graphs, eliciting prior distributions for scale parameters in hierarchical Bayesian models, and Bayesian optimization.

**Srijan Sengupta** completed a B.Stat and M.Stat from Indian Statistical Institute in Kolkata, India and a PhD in statistics from the University of Illinois-Urbana Champaign. He is currently an associate professor of statistics at North Carolina State University, Raleigh, North Carolina, USA. His research interests include developing inferential algorithms for networks with applications in epidemiology, social sciences, and environmental health.