Statistica Sinica Preprint No: SS-2022-0108	
Title	Subsampling Based Community Detection for Large Networks
Manuscript ID	SS-2022-0108
URL	http://www.stat.sinica.edu.tw/statistica/
DOI	10.5705/ss.202022.0108
Complete List of Authors	Sayan Chakrabarty, Srijan Sengupta and Yuguo Chen
Corresponding Authors	Yuguo Chen
E-mails	yuguo@illinois.edu
Notice: Accepted version subject to English editing.	

SUBSAMPLING BASED COMMUNITY DETECTION FOR LARGE NETWORKS

Sayan Chakrabarty, Srijan Sengupta and Yuguo Chen

Abstract: Large networks are becoming pervasive in scientific applications. Statistical analysis of such large networks is prohibitive due to exorbitant runtime and high memory requirements. We propose a subsampling based divide-and-conquer algorithm, `SONNET`, for community detection in large networks. The algorithm splits the original network into multiple subnetworks with a common overlap, and carries out detection algorithm for each subnetwork. The results from individual subnetworks are aggregated using a label matching method to get the final community labels. This method saves both memory and computation costs significantly as one needs to store and process only the smaller subnetworks. This method is also parallelizable which makes it even faster.

Key words and phrases: Community detection, computational efficiency, degree corrected blockmodel, spectral clustering, stochastic blockmodel, subsampling.

1. Introduction

Network data appears in a wide variety of scientific and technological disciplines, such as social media (Sarkar and Rózemberczki, 2021), epidemiology

(Leitch et al., 2019), neuroscience (Roncal et al., 2013), and transportation (Gastner and Newman, 2006). A number of statistical models have been developed for analyzing such network data, starting with the homogeneous random graph model proposed by Erdös and Rényi (1959). In recent years, there has been substantial interest in blockmodels, such as the stochastic blockmodel (SBM), that allow nodes to be partitioned into different communities or blocks (Holland et al., 1983; Goldenberg et al., 2010). A number of generalizations of the SBM have been developed, such as the mixed membership blockmodel (Airoldi et al., 2008), the degree corrected blockmodel (DCBM) (Karrer and Newman, 2011), the popularity adjusted blockmodel (PABM) (Sengupta and Chen, 2018), etc.

One of the main inferential tasks on a network with an underlying community structure is to discover the community membership of each node. A number of community detection algorithms have been studied in the literature. This includes spectral clustering and its variants that leverage the eigen structure of the network (Rohe et al., 2011; Lei and Rinaldo, 2015; Sarkar and Bickel, 2015; Sengupta and Chen, 2015; Cao and Chen, 2011), likelihood based methods that maximize the model likelihood (Amini et al., 2013; Zhao et al., 2012; Nowicki and Snijders, 2001), as well as optimization based methods (Chen et al., 2012; Le et al., 2014). Most of these community

detection algorithms have a high computation complexity. For example, the simplest version of spectral clustering involves eigen decomposition of the graph Laplacian, which can have computation complexity of $O(n^3)$ for a network with n nodes (Pan and Chen, 1999). Likelihood modularity based methods require iterative optimization of the likelihood functions, which is computationally even more expensive (Mukherjee et al., 2021).

The high computational expense of community detection is also due to computational memory requirement. During the analysis of the network, e.g., when performing eigen decomposition, the computer program needs a large amount of memory to store the intermediate variables in the RAM. Table 1 illustrates the memory used in storage (ROM) and eigen decomposition (RAM) of a network adjacency matrix stored as a sparse matrix in R. The following computations were done in a cluster with 1 computing node and 160 gigabyte of RAM. Although the storage space for a 20000-node network is 91.61 megabyte, the computation memory is nearly 10 gigabyte, which is more than the memory available in most personal computers. Extrapolating to a 100000-node network, a computing system with at least $10 \times 5^2 = 250$ gigabyte of memory is needed for the eigen decomposition alone. The computation time also becomes progressively infeasible.

Number of nodes	Storage memory ROM (MB)	Computation memory RAM (MB)	Computation time (hour)
10000	26.75	2480	4.58
20000	91.61	9806	22.28
30000	188.92	21948	55.22

Table 1: Computation costs in eigen decomposition of large networks.

Many real world networks are even larger than those in Table 1, with the number of nodes in hundreds of thousands or even millions. Such large networks are increasingly prevalent in many important application areas, such as neuroscience and cybersecurity (Guo et al., 2020; Roncal et al., 2013). Due to high runtime and memory cost, it is infeasible to carry out statistically principled community detection on such networks.

One way to overcome these issues is to use a subsampling based method that divides the network into smaller subnetworks and performs the community detection algorithm on each of the subnetworks. Each subnetwork requires significantly smaller memory and time for the computation, resulting in overall gain in both storage and computation costs. There are only a few divide and conquer based methods for community detection in the statistics literature. Mukherjee et al. (2021) introduced two algorithms that split a large network into smaller subnetworks, perform community detection in the subnetworks, and combine the results in two different ways. These algorithms are shown to improve the runtime by a big margin without much

compromise in the accuracy of community recovery. There have been many developments in the computer science literature, where fast computation methods and/or parallelization were used for the spectral decomposition methods (Yang and Xu, 2015; Karypis and Kumar, 1998; Yan et al., 2009; Chen et al., 2011). Any divide and conquer algorithm on networks loses information on the interaction between the subnetworks. Another challenge for a divide and conquer algorithm for community detection is of combining the outputs from the subnetworks into a final set of meaningful labels for the entire network. Since the community labels are correct up to any permutation, combining the results becomes challenging.

This paper introduces a versatile algorithm called Subsampling on Networks or `SONNET` that takes subsamples from the network and applies the relevant community detection algorithm on each subsample. It then stitches the community labels from each subsample to get the community labels for the entire network. `SONNET` can be paired with any community detection algorithm on any network. Our key methodological innovation is to overcome the label permutation challenge by first selecting a number of nodes from the main network to create an overlapping part. Then the remaining nodes are partitioned into several mutually exclusive and exhaustive groups. `SONNET` attaches nodes in the overlapping part to each of the

groups to form a set of induced subnetworks with some common nodes. It then applies the community detection algorithm on each of the subnetworks to obtain the community labels for the nodes in each subnetwork. Finally, the labels from each subnetwork are matched using the labels of the overlapping nodes to obtain the final set of labels for all the nodes. A preliminary version of `SONNET` was discussed by Kumar (2017) in a Master's thesis. We modify and generalize the algorithm so that it can be paired up with any suitable community detection algorithms to work on a bigger class of networks. We derive a detailed generalized theory for `SONNET` and use SBM and DCBM to illustrate how the general theory can be used to obtain error bounds and complexities, and how `SONNET` can be applied on such networks. We also compare `SONNET` with contemporary divide and conquer methods both theoretically and numerically. Open source code for implementing the algorithm and reproducing the results is available at `https://github.com/sayan-ch/SONNET-Community-Detection`.

The paper is structured as follows. Section 2 presents and explains `SONNET`. Section 3 contains a general upper bound for the error rate, general computation complexity, results on data usage by `SONNET`, and theories for SBM and DCBM. Section 4 describes the applications of `SONNET` on simulated and real networks. Section 5 presents a discussion on `SONNET`.

2. Methodology

We first introduce some notations. We use the terminology simple network to indicate undirected, unweighted graph without any self-loops or multiple edges between nodes. We use the letter S to denote the set of n nodes in the entire network G and K to denote the number of communities. For the scope of this paper, we assume K is fixed and known. We use s and o to denote the number of subnetworks and the size of the overlapping part used in `SONNET`, respectively. The set of o nodes in the overlapping part is denoted by S_0 and the s partitions of the remaining nodes in the non-overlapping part $S \backslash S_0$ by $S'_1, \ldots, S'_s$. The set of nodes in the ith subnetwork is $S_i = S_0 \cup S'_i$, $i = 1, \ldots, s$. Given a subset of nodes $S_d \subset S$, the subnetwork spanned by S_d is denoted by G_{S_d}. We use the terms subnetwork and subgraph interchangeably. A is the $n \times n$ adjacency matrix corresponding to the network G. C represents the $n \times K$ community membership matrix. The (i, j)th element C_{ij} is 1 if the ith node belongs to the jth community, and 0 otherwise. $\mathbb{C}_{n \times K}$ represents the space of all community membership matrices of order $n \times K$. For a network with K communities, the proportion of nodes in the ith community is denoted by π_i, $i = 1, \ldots, K$, and π_{min} and π_{max} are the proportions of nodes in the smallest and the largest communities, respectively. The l_0 norm of a matrix C is defined as the number

of non-zero entries in C, i.e., $\|C\|_0 = \sum_{i=1}^{n}\sum_{j=1}^{K} \mathbf{1}\{C_{ij} \neq 0\}$, where $\mathbf{1}\{\cdot\}$ is the indicator function. A permutation matrix $E_{K\times K}$ is a matrix which has a single 1 in each row and column, and 0 elsewhere. The space of all permutation matrices of size $K \times K$ is defined as $\mathbf{E}_K$. For any matrix D and two sets of indices I, J, the notation $D_{I,J}$ denotes the submatrix of D that has the rows corresponding to the indices in I and the columns corresponding to the indices in J. Similarly, D_{I*} denotes the submatrix of D with the rows corresponding to the indices in I and all the columns, and D_{*J} denotes the submatrix of D with the columns corresponding to the indices in J and all the rows. For a matrix $M_{p\times q}$, the matrix $2,1$ norm is defined as $\|M\|_{2,1} = \sum_{i=1}^{p}\|M_{i*}\|$, where $\|\cdot\|$ is the Euclidean norm. We use the abbreviation 'w.p.' for 'with probability'.

The algorithm `SONNET` consists of four main parts: division, detection, stitching, and repetition. At the division step, the entire network is split into multiple subnetworks, with a set of overlapping nodes which is common to all the subnetworks. Then at the detection step, we apply a suitable community detection algorithm on each of the subnetwork. Since the subnetworks are small compared to the entire network, this step is faster than community detection on the full network. Also, this is the step which can be parallelized by running community detection for each subnetwork on dif-

ferent processors of a multi-core computer. Next, we 'stitch' the multiple sets of labels that we get as the outputs from the multiple subnetworks to obtain the labels for the entire network. The labels of the overlapping part from different subnetworks are matched to perform the stitching. Since we use information from multiple sets of labels for the overlapping part and only one set of labels for each non-overlapping part, the community detection might not always be accurate for the non-overlapping parts. Thus we split the entire non-overlapping part again into multiple subgraphs and perform community detection on each subgraph. We stitch the community labels from the new subgraphs with the labels from the previous step. We repeat this step a number of times to get accurate community assignments for the nodes in the non-overlapping part.

At the division step, we fix the overlap size o and the number of subnetworks s. Then, we select o nodes from the entire network, and partition the remaining $n - o$ nodes randomly into s different groups of size $m = (n - o)/s$. We form s subnetworks by inducing the subnetworks corresponding to the $o + m$ nodes. Each subnetwork contains all the nodes from the overlapping part and m nodes from the corresponding partition of the non-overlapping part. The parameters o and s must be selected in such a way to ensure that nodes from all communities are represented in each of

the subnetworks. The selection criterion depends on the random selection and partitioning mechanism as well. When simple random sampling without replacement (SRSWOR) is used to select the overlapping part, o must be greater than the order of K^3 in a balanced network to guarantee the inclusion of all the communities in it with a high probability (See Lemma S3 in the Supplementary Material). In the presence of degree heterogeneity in the network, a random selection of the overlapping part might not be optimal. Since overlapping part contains the information to stitch the outputs from each subnetwork, it must be well connected with all the partitions of the non-overlapping part. In that case, one may use sampling proportional to the degree or some random walk based methods to ensure that higher degree nodes from all the communities are well represented in the overlapping part.

In the next step, we select a community detection algorithm and apply it on each of the s subnetworks to obtain community labels for the nodes in each subnetwork. The selection of the community detection algorithm plays a crucial role in determining the accuracy of the overall community detection using `SONNET`. The community detection algorithm itself has to be reasonably accurate on the subnetworks for `SONNET` to be accurate.

In the stitching step, we match the community labels of the overlap-

ping nodes across different subnetworks, which is challenging. For a small number of communities K, a brute force method (Algorithm S1 in the Supplementary Material) is used where we search over all possible permutations of the community labels to find the best match between two sets of labels. The drawback of the brute force method is that it is computationly expensive for a large value of K as it takes $K!$ searches. For large K, we use a greedy search algorithm to find the best match between two sets of labels instead. We present two label matching algorithms, `MatchBF` and `MatchGreedy`, for the brute force search and the greedy search, respectively. Both algorithms take two community membership matrices C_1 and C_2 (both $o \times K$) as input, and return a permutation matrix $E \in \mathbf{E}_K$. While `MatchBF` returns $E_0 = \underset{E \in \mathbf{E}_K}{\arg\min} \|C_1 E - C_2\|_0$, `MatchGreedy` is shown to return E_0 under certain assumptions (Mukherjee et al., 2021). The algorithms are described in Appendix S1. We fix a subnetwork at random (for example, we choose the first subnetwork in our application) and match the labels of the overlapping part of the other subnetworks with the fixed subnetwork. We use the resulting permutations that are used for the overlapping part to change the labels of the non-overlapping part as well for each subnetwork. Then, we combine the labels for the non-overlapping part and take a vote of count for the overlapping part. If there is a tie when taking a vote for

Algorithm 1 SONNET

Input A network G with n nodes in S and adjacency matrix A, number of communities K, number of subgraphs s, size of the overlapping part o, number of repetitions r, a community detection algorithm $\mathcal{A}$, and a label matching algorithm $\mathcal{M}$.

Output A membership matrix $\hat{C}_{n\times K}$.

procedure SONNET$(S, K, s, o, r, \mathcal{A}, \mathcal{M})$

1. Select o nodes at random from S. Name the corresponding set of nodes S_0.

2. Split $S \setminus S_0$ randomly into s exhaustive parts: $S'_1, \ldots, S'_s$.

3. Form $S_i = S_0 \cup S'_i$, $i = 1, \ldots, s$.

4. Apply clustering algorithm $\mathcal{A}$ on the subnetwork G_{S_i} and label the output as $\hat{C}^{(i)}_{(m+o)\times K}$, $i = 1, \ldots, s$, where $m = (n-o)/s$.

5. Compute $E_i = \mathcal{M}\left(\hat{C}^{(i)}_{S_0*},\ \hat{C}^{(1)}_{S_0*}\right)$, $i = 2, \ldots, s$.

6. Update $\hat{C}^{(i)} = \hat{C}^{(i)}E_i$, $i = 2, , \ldots, s$. Make $\hat{C}^{(i)}$ an $n\times K$ matrix by inserting rows with all 0's for the nodes that are not present in $\hat{C}^{(i)}$, $i = 1, \ldots, s$. Compute $\hat{C}^{(temp)}$ such that $\hat{C}^{(temp)}_{S\setminus S_0*} = \sum_{i=1}^{s} \hat{C}^{(i)}_{S\setminus S_0*}$ and $\hat{C}^{(temp)}_{S_0*}$ is a matrix of 0's.

7.

for (j from 1 to r) **do**

7.1. Split $S \setminus S_0$ randomly into s partitions: $S'_{1j}, \ldots, S'_{sj}$.

7.2. Apply clustering algorithm $\mathcal{A}$ on the subnetwork $G_{S'_{ij}}$ and label the output as $\hat{C}^{(ij)}$ for $i = 1, \ldots, s$.

7.3. Compute $E_{ij} = \mathcal{M}\left(\hat{C}^{(ij)},\ \hat{C}^{(temp)}_{S'_{ij}*}\right)$, $i = 1, \ldots, s$.

7.4. Update $\hat{C}^{(ij)} = \hat{C}^{(ij)}E_{ij}$, $i = 1, \ldots, s$. Make $\hat{C}^{(ij)}$ an $n \times K$ matrix by inserting rows with all 0's for the nodes that are not present in S'_{ij}. Compute $\hat{C}_j = \sum_{i=1}^{s} \hat{C}^{(ij)}$.

8. Set $\hat{C}_{S\setminus S_0} = \frac{1}{r+1}\left(\hat{C}^{(temp)} + \sum_{j=1}^{r} \hat{C}_j\right)$. Replace the highest value in each row corresponding to $S \setminus S_0$ by 1 and the rest by 0. Break the ties, if any, randomly.

9. Set $\hat{C}_{S_0} = \frac{1}{s}\sum_{i=1}^{s} \hat{C}^{(i)}_{S_0*}$. Replace the highest value in each row by 1 and the rest by 0. Break the ties, if any, randomly. Make it an $n \times K$ matrix by inserting rows with all 0's for the nodes that are not in the overlapping part.

10. Return $\hat{C} = \hat{C}_{S_0} + \hat{C}_{S\setminus S_0}$.

the nodes in the overlapping part, we use one of the tied labels randomly.

Finally in the repetition step, we split the entire non-overlapping part randomly into s subnetworks. Then we perform community detection on each of these parts and match the labels with those from the output from the stitching step. We repeat this procedure r times and take a vote of count for each of the nodes in the non-overlapping parts out of the r sets of labels from this step. This step, although not obligatory, can be used to increase the accuracy of `SONNET`. A large value of r can also increase the computation time significantly, defeating the purpose of `SONNET`. In practice, we select a combination of s and o such that we expect a reasonable precision in a quick time, and then increase the value of r based on the computation budget.

`SONNET` is presented in Algorithm 1. The following new notations are used in Algorithm 1. The output membership matrices from applying community detection algorithm $\mathcal{A}$ on each subnetwork spanned by S_i is defined as $\hat{C}^{(i)}_{(m+o)\times K}$, $i = 1, \ldots, s$.

3. Theoretical Results

3.1 General Error Bound for `SONNET`

`SONNET` is a generalized divide and conquer algorithm that can be used with any community detection algorithm that is reasonably accurate. In this

section, we develop the general theory for a simplified version of `SONNET`, called `SimpleSONNET` (Algorithm 2). In `SimpleSONNET`, we assign the labels from the first subgraph to the nodes in the overlapping part instead of performing a majority voting. Label matchings are done using brute force method (Algorithm S1) in `SimpleSONNET`. We omit the repetition step in this version of the algorithm. The theory is not specific to any community detection algorithm or any network model. It provides an upper bound for the error rate of `SimpleSONNET` given that the community detection algorithm has low error rates on the subnetworks. We also apply the general bound on two special cases: spectral clustering on SBM and spherical K-median spectral clustering on DCBM in Sections 3.4 and 3.5.

Given two $n \times K$ community membership matrices C_1 and C_2, the number of mismatches is defined as $\mathcal{M}(C_1, C_2) = \arg\min_{E \in \mathbf{E}_K} \|C_1 E - C_2\|_0$, and the proportion of mismatches or error rate is $\delta(C_1, C_2) = \frac{1}{n}\mathcal{M}(C_1, C_2)$. Note that the output from `SimpleSONNET` algorithm with s subgraphs and an overlapping size of o can be written as

$$\hat{C} = \hat{C}^{(1)}_{S_{0*}} + \hat{C}^{(1)}_{S'_1*} + \hat{C}^{(2)}_{S'_2*} E_2 + \cdots + \hat{C}^{(s)}_{S'_s*} E_s = \hat{C}^{(1)}_{S_{1*}} + \hat{C}^{(2)}_{S'_2*} E_2 + \cdots + \hat{C}^{(s)}_{S'_s*} E_s, \quad (3.1)$$

where $\hat{C}^{(j)}_{S'_j*}$ is the community membership matrix obtained by applying the community detection algorithm $\mathcal{A}$ to subgraph G_{S_j} and padded with $\mathbf{0}$ rows

for the nodes that are not present in S'_j, and $E_j = \arg\min_{E \in \mathbf{E}_K} \|\hat{C}^{(j)}_{S_0*} E - \hat{C}^{(1)}_{S_0*}\|$.

Algorithm 2 `SimpleSONNET`

Input A network G with n nodes in S and adjacency matrix A, number of communities K, number of subgraphs s, size of the overlapping part o, and a community detection algorithm $\mathcal{A}$.

Output A membership matrix $\hat{C}_{n\times K}$.

procedure SIMPLESONNET$(S, K, s, o, \mathcal{A})$

1. Select o nodes at random from S. Name the corresponding set of nodes S_0.

2. Split $S \setminus S_0$ randomly into s exhaustive parts: $S'_1, \ldots, S'_s$.

3. Form $S_i = S_0 \cup S'_i$, $i = 1, \ldots, s$.

4. Apply clustering algorithm $\mathcal{A}$ on the subnetwork G_{S_i} and label the output as $\hat{C}^{(i)}_{(m+o)\times K}$, $i = 1, \ldots, s$, where $m = \frac{n-o}{s}$.

5. Compute $E_i = \arg\min_{E \in \mathbf{E}_K} ||\hat{C}^{(i)}_{S_0*} E - \hat{C}^{(1)}_{S_0*}||_0$, $i = 2, , \ldots, s$.

6. Update $\hat{C}^{(i)} = \hat{C}^{(i)} E_i$, $i = 2, , \ldots, s$. Make $\hat{C}^{(i)}$ an $n \times K$ matrix by inserting $\mathbf{0}$ rows for the nodes that are not present in $\hat{C}^{(i)}$, $i = 1, \ldots, s$.

7. Return $\hat{C} = \hat{C}^{(1)}_{S_0*} + \hat{C}^{(1)}_{S'_1*} + \hat{C}^{(2)}_{S'_2*} + \cdots + \hat{C}^{(s)}_{S'_s*}$.

We are interested in obtaining an upper bound for the quantity $\delta(\hat{C}, C)$, where C is the true $n \times K$ membership matrix for the network G. Note that, for any $E \in \mathbf{E}_K$,

$$\begin{aligned}
&\|\hat{C}E - C\|_0 \\
=&\|\hat{C}^{(1)}_{S_0*}E - C_{S_0*}\|_0 + \|\hat{C}^{(1)}_{S'_1*}E - C_{S'_1*}\|_0 + \|\hat{C}^{(2)}_{S'_2*}E_2E - C_{S'_2*}\|_0 + \cdots + \|\hat{C}^{(s)}_{S'_s*}E_sE - C_{S'_s*}\|_0 \\
=&\|\hat{C}^{(1)}_{S_1*}E - C_{S_1*}\|_0 + \|\hat{C}^{(2)}_{S_2*}E_2E - C_{S_2*}\|_0 + \cdots + \|\hat{C}^{(s)}_{S'_s*}E_sE - C_{S'_s*}\|_0. \qquad (3.2)
\end{aligned}$$

Taking the minimum over $E \in \mathbf{E}_K$ on the left hand side of (3.2), in general, results into a larger value than taking the minimum on the individual terms

on the right hand side, as the minimum is taken over a larger space on the right hand side where each partition $\hat{C}^{(j)}_{S_j*}$ or $\hat{C}^{(j)}_{S'_j*}$ can have different best permutation matrices. In the following results, we show that under certain assumptions on the network and the community detection algorithm, the permutation matrix that minimizes the term on the left can be used to find expressions for the permutation matrices that minimize the terms on the right. We use this insight to find an upper bound for the error rate.

Lemma S2 in the Supplementary Material states that multiplying each term with a permutation matrix inside $\|.\|_0$-norm does not change its value.

If some communities are not well represented in the overlapping part in `SimpleSONNET`, the stitching step produces erroneous results. Consider the situation where a particular community is completely absent from the overlapping part. Then all the nodes in that community will be assigned community labels at random. To ensure that all the communities are well represented in the overlapping part, its size o should not be too small. Lemma S3 in the Supplementary Material states a condition on o such that the size of the smallest community in the overlapping part is larger than a certain value. We denote the sizes of the smallest and the largest communities in the overlapping part by $\hat{o}_{min}$ and $\hat{o}_{max}$ respectively. Then, for a balanced network where $\pi_{max}=\pi_{min}=\frac{1}{K}$, if $o/K(K+1)^2 \to \infty$, then

$\hat{o}_{min} \geq o/(K+1)$ and $\hat{o}_{max} \leq o/(K-1)$ hold with probability $\geq 1-\omega_o$, where $\omega_o = K\exp\{-o/(4K(K+1)^2)\}$.

The following theorem states that the permutation matrix, which best matches a subgraph with the truth, also best matches any of its subgraphs, provided the community detection algorithm satisfies certain conditions. It also provides an expression for the permutation matrix that stitches each subgraph with the first subgraph using only the overlapping part.

Theorem 1. *Suppose for each partition $S_q = S_0 \cup S_q'$, $q = 1, \ldots, s$, there exists an upper bound $\epsilon > 0$ for the proportion of misclustered nodes in each subgraph and a probability α (free from q), such that the following conditions hold:*

$$\boldsymbol{P}\left(\|\hat{C}^{(q)}_{S_{q*}} E^*_q - C_{S_{q*}}\|_0 < \epsilon(o+m)\right) \geq 1-\alpha, \tag{3.3}$$

$$\text{where } E^*_q \coloneqq \underset{E\in\mathbf{E}_K}{\arg\min}\|\hat{C}^{(q)}_{S_{q*}} E - C_{S_{q*}}\|_0,\ q = 1, \ldots, s, \text{ and}$$

$$\epsilon(o+m) \leq \frac{o\pi_{min}}{(1+\pi_{min})}. \tag{3.4}$$

Then we have

$$\boldsymbol{P}\Big(\underset{E\in\mathbf{E}_K}{\arg\min}\|\hat{C}^{(q)}_{S_{0*}} E - C_{S_{0*}}\|_0 = E^*_q\Big) \geq 1-\omega_o-2\alpha,\ q = 1, \ldots, s, \text{ and} \tag{3.5}$$

$$\boldsymbol{P}\Big(E^*_q E_1^{*-1} = \underset{E\in\mathbf{E}_K}{\arg\min}\|\hat{C}^{(q)}_{S_{0*}} E - \hat{C}^{(1)}_{S_{0*}}\|_0\Big) \geq 1-\omega_o-4\alpha,\ q = 2, \ldots, s, \tag{3.6}$$

where ω_o is defined similarly as in Lemma S3.

Proofs of the above theorem and other theoretical results are given in the Supplementary Material. Condition (3.3) ensures that there is a uniform upper bound to the output of algorithm $\mathcal{A}$ when applied on the individual subgraphs in `SimpleSONNET`. Condition (3.4) states that the error bound should be reasonably small. Also, the right hand side of Condition (3.4) tends to be smaller for unbalanced cases, which makes it necessary to find a tighter upper bound ϵ for the number of misclustered nodes in each subgraph. To summarize, for the algorithm `SimpleSONNET` to effectively recover the community labels for a large network using a community detection algorithm, the communities in the whole network must not be too unbalanced, and the underlying community detection algorithm must be able to recover the community labels in each subgraph with certain accuracy.

Combining the above results, we obtain the following main theorem.

Theorem 2. *Under the assumptions in Theorem 1, the overall proportion of misclustered nodes for the output of* `SimpleSONNET` *on G with s subgraphs and an overlapping size o satisfies*

$$\delta(\hat{C}, C) = \frac{1}{n} \min_{E \in \mathbf{E}_K} \|\hat{C}E - C\|_0 \leq \frac{s(o+m)}{n}\epsilon$$
$$w.p. \quad \geq 1 - (s-1)\omega_o - (5s-4)\alpha, \tag{3.7}$$

where ω_o and α are defined in the same way as Theorem 1.

Theorem 2 gives an upper bound for the error in community detection using `SimpleSONNET` in terms of the upper bound ϵ for the error rate in community detection of the individual subgraphs. In Sections 3.4 and 3.5, we derive upper bounds on the error rate when `SimpleSONNET` is applied with spectral clustering on SBMs, and with spherical K-median spectral clustering on DCBMs.

3.2 Computational Costs of `SONNET`

In this section, we find a general expression for the complexity of `SONNET`. Assume that `SONNET` is applied with community detection algorithm $\mathcal{A}$ on a network with n nodes and K communities. Let the computation complexity of $\mathcal{A}$ on such a network be $\boldsymbol{T}(n, K)$. Then the computation complexity of `SONNET` due to the detection step is $O(s\boldsymbol{T}(o + m, K))$ as algorithm $\mathcal{A}$ is applied on s subgraphs of size $o + m$ each.

The complexity due to the stitching step for the overlapping part is dominated by the label matching of the overlapping part in each of $G_{S_2}, \ldots, G_{S_s}$ with that of G_{S_1}. Label matching of two sets of labels of size o, ranging from 1 to K, takes $O(oK!)$ time. Thus, the computation complexity from this step is $O(soK!)$. The computation complexity due to the repetition step is split into two parts. In the repetition step, community detection al-

gorithm $\mathcal{A}$ is applied on s new subgraphs of size m and is repeated r times. The complexity due to this part is $O(rs\boldsymbol{T}(m, K))$. In the second part, for each repetition, s label matchings are performed on sets of labels of size m, ranging from 1 to K. Thus, the complexity due to this part is $O(rsmK!)$.

Therefore, the total computation complexity of `SONNET` with community detection algorithm $\mathcal{A}$ and brute force label matching is $O(s\boldsymbol{T}(o+m, K))+O(oK!)+O(rs\boldsymbol{T}(m, K))+O(rsmK!)$. In general, the time function $\boldsymbol{T}$ of the community detection algorithm can be assumed to be non-decreasing with the size of the network. Thus, $\boldsymbol{T}(m, K) \leq \boldsymbol{T}(o+m, K)$. Also, for most of the parameter combinations, $rsm = r(n-o) \geq o$. Then we simplify the above expression to $O(rs\boldsymbol{T}(o+m, K))+O(rsmK!)$.

For small K, the complexities due to the stitching steps for both the overlap and the repetitions are small compared to the subgraph detection parts. For large K, label matching is very slow as the complexity is $O(K!)$, and some greedy label matching algorithm is recommended. If Algorithm S2 (`MatchGreedy`) is used as the label matching algorithm with `SONNET`, the computation complexity reduces to $O(rs\boldsymbol{T}(o+m, K))+O(rsmK^2)$.

The complexity of algorithm $\mathcal{A}$ depends on many factors including the base method used (eigen decomposition, or likelihood/modularity optimization, or convex optimization, etc.) and the sparsity of the network, e.g.,

spectral clustering of a dense matrix has the complexity of $\boldsymbol{T}(n, K) = n^3$, whereas the same can be performed with a complexity of $\boldsymbol{T}(n, K) = n^\theta$ for some $1 < \theta < 3$ depending on the sparsity of the network and the algorithm used for eigen decomposition. The ratio of complexities of `SONNET` implemented in parallel using *ncore* processors to algorithm $\mathcal{A}$ with complexity n^θ on the whole network is approximately $\lceil \frac{rs}{ncore} \rceil \left(\frac{o+m}{n}\right)^\theta$. For $\theta > 1$, `SONNET` will usually see a computational gain compared to the whole network.

For a network of size n, both the storage and computation memory are proportional to n^2. For `SONNET`, one only needs to store s subnetworks of size $(m + o)$ for computation. Since only one subnetwork is stored in the memory for each computation, the maximum memory required to execute the computation is proportional to $(m + o)^2$. For example, if `SONNET` with $s = 100$, $o = 5000$ is applied on a network of size 50000, only $\frac{(5450)^2}{50000^2} \times 100\% = 1.18\%$ of the memory that is required for community detection on the whole network is needed. In case community detection on the entire network is infeasible due to memory limitations, `SONNET` can still be applied.

3.3 Optimizing Trade-off Between Data Usage and Computation Complexity of `SONNET`

We use the term data usage to refer to the part of the network adjacency matrix that is used in `SONNET`.

Theorem 3. *The expected data usage proportion of* `SONNET` *up to the* ρ*th repetition step with* s *subgraphs and overlapping size* o *is*

$$1 - \left(\frac{n-o}{n}\right)^2 \left(\frac{s-1}{s}\right)^{\rho+1}. \tag{3.8}$$

From (3.8), the expected proportion of used pairs up to the ρth repetition step of `SONNET` is a strictly increasing function of ρ. This result suggests that one should choose as large a value of r as the computation resources permit to avoid data wastage. We exploit the trade-off between expected data usage proportion and the computation complexity to select the parameters for `SONNET`. We maximize the data usage proportion over s and o with an upper bound q of the ratio of complexities of `SimpleSONNET` to that of algorithm $\mathcal{A}$ on the whole network. Once the optimized values of s and o are obtained, one may choose the number of repetitions based on available computing resources.

Assume that the complexity of $\mathcal{A}$ is given by n^θ for some $\theta > 0$, and the number of communities K does not scale with n. Then the ratio of complexities of `SONNET` with $\mathcal{A}$ to that of $\mathcal{A}$ on the whole network is given by $s((o+m)/n)^\theta$. Let q be the runtime budget for `SONNET`, expressed as a fraction of the runtime needed for the whole network. For example, $q = 0.001$ means we want `SONNET` to take no more than 0.1% of the runtime for the whole network. Then, we carry out the following constrained optimization

to obtain s and o:

$$\max_{s,o} \left[1 - \left(\frac{n-o}{n} \right)^2 \left(\frac{s-1}{s} \right) \right] \tag{3.9}$$

$$\text{with the constraint } s \left(\frac{o+m}{n} \right)^\theta \leq q. \tag{3.10}$$

3.4 `SimpleSONNET` with Spectral Clustering on SBM

We use the general bound structure to obtain an upper bound for the expected error rate when `SimpleSONNET` is applied with spectral clustering (Algorithm S3 in the Supplementary Material) on SBMs (the Supplementary Material contains the details of SBM). First, we derive a uniform upper bound ϵ for the expected error rates when spectral clustering is applied on each of the subgraph. Then we obtain a final bound using Theorem 2.

Lei and Rinaldo (2015) derived an upper bound for the proportion of misclustered nodes for the output from spectral clustering on an SBM that satisfies certain conditions. We restate the bound with our notations in Theorem S1 in the Supplementary Material. However, Theorem S1 is not directly applicable to a random subgraph $G_{\mathbb{S}_d}$ of size d as the terms $\pi_{min}^{(\mathbb{S}_d)}$ and $\pi_{max}^{(\mathbb{S}_d)}$ in the error bound are random. Thus, we use a lower bound for $\pi_{min}^{(\mathbb{S}_d)}$ and an upper bound for $\pi_{max}^{(\mathbb{S}_d)}$ from Lemma S3 to replace those in the bound in Theorem S1 and obtain the following results.

Theorem 4. *Suppose $\mathbb{S}_d$ is a random subset of d nodes from the n nodes in*

S. Assume that the network G satisfies Conditions 1, 2, and 3 of Theorem S1. Let $\hat{C}^{(\mathbb{S}_d)}$ *be the output of* $(1+\delta)$*-approximate* K*-means spectral clustering (Algorithm S3) applied on* $G_{\mathbb{S}_d}$*. Then there exists an absolute constant* c *such that if*

$$(2+\delta)\frac{K(1+\pi_{min})^2}{\pi_{min}^2\lambda^2 d\alpha_n} < c, \tag{3.11}$$

then with probability at least $1-\frac{1}{d}-3\omega_d$*,*

$$\delta(\hat{C}^{(\mathbb{S}_d)}, C_{\mathbb{S}_d *}) \leq c^{-1}(2+\delta)\frac{K\pi_{max}(1+\pi_{min})^2}{\pi_{min}^2(1-\pi_{max})\lambda^2 d\alpha_n}, \tag{3.12}$$

where $\omega_d = K\exp\left(-\frac{d\pi_{min}^3}{4(1+\pi_{min})^2}\right)$*, and* α_n *and* λ *are defined in Theorem S1.*

We use Theorem S1 along with Theorem 4 to derive an upper bound for the expected error rate when `SimpleSONNET` is applied with $(1+\delta)$-approximate K-means spectral clustering on SBM.

Theorem 5. *Let* $\hat{C}$ *be the output of* ***`SimpleSONNET`****, applied with* $(1+\delta)$*-approximate* K*-means spectral clustering on a network* G *of size* n *that is generated by SBM. Assume* G *satisfies Conditions* $1, 2,$ *and* 3 *in Theorem S1. Suppose* s *subgraphs were used with an overlapping size of* o*. Then there exists an absolute constant* c *such that if*

$$(2+\delta)\frac{K(1+\pi_{min})^2}{\pi_{min}^2\lambda^2(o+m)\alpha_n} < c, \text{ and} \tag{3.13}$$

$$o \geq 2c^{-1}(2+\delta)\frac{K\pi_{max}(1+\pi_{min})^3}{\pi_{min}^3(1-\pi_{max})\lambda^2\alpha_n}, \tag{3.14}$$

then with probability $\geq 1-(s-1)\omega_o - 3(5s-4)\omega_{o+m} - \frac{5s-4}{o+m}$,

$$\delta(\hat{C}, C) \leq c^{-1}(2+\delta)\frac{sK\pi_{max}(1+\pi_{min})^2}{\pi_{min}^2(1-\pi_{max})\lambda^2 n\alpha_n}. \tag{3.15}$$

Condition 3.13 in this theorem is technical and ensures that the network model parameters are such that we can apply the bound given in Theorem S1. Condition 3.14 comes from Theorem 1 that ensures that spectral clustering is reasonably accurate for each of the subgraphs.

3.5 `SimpleSONNET` with Spherical Spectral Clustering on DCBM

We use the general bound structure to obtain an upper bound for the expected error rate when `SimpleSONNET` is applied with spherical K-median spectral clustering on DCBMs. Details of Algorithm S4 and the DCBM are in the Supplementary Material. First, we derive a uniform upper bound ϵ for the expected error rates when spectral clustering is applied on each of the subgraph. Then we obtain a final bound using Theorem 2.

We introduce some new notations for DCBM. Let $\psi \in \mathbb{R}^n$ be the node-level degree parameter. Let ϕ_k be the $n \times 1$ vector that agrees with ψ on G_k and zero otherwise. Define $\tilde{\phi}_k = \frac{\phi_k}{\|\phi_k\|}$, and $\tilde{\psi} = \sum_{k=1}^{K} \tilde{\phi}_k$. We also define effective community size $\tilde{n}_k = \|\phi_k\|^2$. Let $\tilde{n}_{min} = \min\{\tilde{n}_1, \ldots, \tilde{n}_K\}$.

Theorem S2 in the Supplementary Material restates a result by Lei and Rinaldo (2015) on an upper bound of the proportion of misclustered nodes

of spherical $(1+\delta)$-approximate K-median spectral clustering applied on a DCBM using our notation. However, Theorem S2 is not directly applicable to a random subgraph $G_{\mathbb{S}_d}$ of G as the terms $\tilde{n}_{min}^{(\mathbb{S}_d)}$ and $\tilde{\psi}_i$'s in the error bound are random. Thus, we find a lower bound for $\tilde{n}_{min}^{(\mathbb{S}_d)}$ in Lemma S6 and an upper bound for $\sum_{i\in\mathbb{S}_d}\tilde{\psi}_i^{-2}$ in Lemma S7 in the Supplementary Material, and replace that in the bound in Theorem S2 to obtain an upper bound for the error rate of spherical K-median spectral clustering on a random subgraph $G_{\mathbb{S}_d}$. The following theorem uses the two lemmas to obtain an error bound when spherical K-median spectral clustering is applied on a subgraph spanned by a random subset $\mathbb{S}_d$ of size d of all the nodes in S.

Theorem 6. *Suppose $G_{\mathbb{S}_d}$ is a random subgraph of size d from a network G of size n that is generated by DCBM. Assume that G satisfies Conditions 1, 2, and 3 of Theorem S2. Let $\hat{C}^{(\mathbb{S}_d)}$ be the output of spherical spectral clustering using $(1+\delta)$-approximate K-median clustering applied on $G_{\mathbb{S}_d}$. Then there exists an absolute constant c such that if*

$$(2.5+\delta)\frac{\sqrt{K(\frac{d}{n}S_{\tilde{\psi}}+\eta^*)}(1+\pi_{min})}{\lambda\left(\frac{d}{n}\tilde{n}_{min}-\gamma^*\right)\pi_{min}\sqrt{d\alpha_n}}<c, \tag{3.16}$$

then, with probability $\geq 1-\frac{1}{d}-\omega_d-\frac{2(K+1)}{n}$,

$$\delta(\hat{C}^{(\mathbb{S}_d)},C_{\mathbb{S}_d*})\leq c^{-1}(2.5+\delta)\frac{\sqrt{K(\frac{d}{n}S_{\tilde{\psi}}+\eta^*)}}{\lambda\left(\frac{d}{n}\tilde{n}_{min}-\gamma^*\right)\sqrt{d\alpha_n}}, \tag{3.17}$$

where $S_{\tilde{\psi}}$, γ^*, *and* η^* *are defined in Lemmas S6 and S7,* α_n *and* λ *in Theorem S2, and* $\omega_d = K \exp\left(-\frac{d\pi_{min}^3}{4(1+\pi_{min})^2}\right)$.

We use Theorem 2 along with Theorem 6 to derive an upper bound for the expected error rate when `SimpleSONNET` is applied with spherical $(1+\delta)$-approximate K-median spectral clustering.

Theorem 7. *Let* $\hat{C}$ *be the output of* `SimpleSONNET`*, applied with spherical* $(1+\delta)$*-approximate* K*-median spectral clustering on a network* G *of size* n *that is generated by DCBM. Assume* G *satisfies Conditions 1, 2, 3 in Theorem S2. Suppose* s *subgraphs were used with an overlapping size of* o*. Then there exists an absolute constant* c *such that if*

$$(2.5+\delta)\frac{\sqrt{K(\frac{o+m}{n}S_{\tilde{\psi}}+\eta^*)}(1+\pi_{min})}{\lambda\left(\frac{o+m}{n}\tilde{n}_{min}-\gamma^*\right)\pi_{min}\sqrt{(o+m)\alpha_n}} < c, \text{ and} \tag{3.18}$$

$$\frac{o}{\sqrt{o+m}} \geq c^{-1}(2.5+\delta)\frac{(1+\pi_{min})\sqrt{K(\frac{o+m}{n}S_{\tilde{\psi}}+\eta^*)}}{\lambda\pi_{min}\left(\frac{o+m}{n}\tilde{n}_{min}-\gamma^*\right)\sqrt{\alpha_n}}, \tag{3.19}$$

then, with probability $\geq 1-(s-1)\omega_o-(5s-4)(\frac{1}{o+m}+\omega_{o+m}+\frac{2(K+1)}{n})$,

$$\delta(\hat{C},C) \leq c^{-1}(2.5+\delta)\frac{s\sqrt{(o+m)K(\frac{o+m}{n}S_{\tilde{\psi}}+\eta^*)}}{\lambda\left(\frac{o+m}{n}\tilde{n}_{min}-\gamma^*\right)n\sqrt{\alpha_n}}. \tag{3.20}$$

4. Application

We apply `SONNET` with spectral and spherical K-median spectral clustering on networks simulated from SBM and DCBM, as well as real-world networks. We compare the performance of `SONNET` with the performance of

community detection on the whole network and GALE. We did not report the results for the other divide and conquer algorithm, PACE, proposed in Mukherjee et al. (2021) as GALE performed better than PACE for the examples we considered. The implementation details of SONNET are given in Section S2.7 in the Supplementary Material.

4.1 SONNET with spectral clustering on SBM

We consider two simulation setups for SBM. In each setup, we independently generate 100 simple networks with n nodes and each node is randomly assigned to one of the K communities. Edges are generated randomly with probability $p^{(intra)}$ for two nodes in the same community, and with probability $p^{(inter)}$ for two nodes in different communities. For each network, we run SONNET with spectral clustering on the Laplacian matrix. We match the best permutation of the outputs of SONNET with the true membership of the nodes to compute the error rate. In the first setup, we take $n = 10000$, $K = 5$, $p^{(intra)} = 0.2$, and $p^{(inter)} = 0.05$. In the second setup, we consider $n = 20000$, $K = 20$, $p^{(intra)} = 0.2$, and $p^{(inter)} = 0.1$. In both cases, we choose the optimized parameters $\tilde{s}$ and $\tilde{o}$ obtained by maximizing (3.9) with the constraint (3.10) for different values of the time constraint q. For each combination of $\tilde{s}$ and $\tilde{o}$, we use $r = 0, 2, 5$ for 10000-

node network and $r = 0, 10, 20$ for 20000-node network. We also applied GALE with spectral clustering in each setup. We tried several combinations of parameters p (number of subgraphs) and T (size of each subgraph) of GALE and reported the cases with the lowest error rate and with the shortest computation time. All the results are in Table 2. Furthermore, we present a detailed study on error rates and runtimes of SONNET for different values of s, o, and r for $n = 10000$ in Section S2.6 in the Supplementary Material.

Table 2 shows that, using the parameter selection method in Section 3.3, SONNET can achieve error rates close to (and in some cases even lower than) spectral clustering on the entire network for both $n = 10000$ and $n = 20000$. For $n = 10000$, SONNET with spectral clustering achieves 0% error rate in 4.1 seconds (highlighted row in Table 2), compared to 1967.5 seconds for spectral clustering on the entire network and 3.23% error rate for GALE in 26.3 seconds. For $n = 20000$, SONNET reaches 0.4% error rate in only 488 seconds (highlighted row in Table 2), compared to 5.6% error rate of spectral clustering on the whole network in 15379 seconds and 7.33% error rate of GALE in 2218 seconds. Overall, SONNET is both faster and more accurate than spectral clustering on the whole network and GALE on SBMs.

SBM Specifications	Method	Parameter Selection q	$\tilde{s}$	$\tilde{o}$	r	Error Rate %	Comp. Time in sec.	Exp. Data Use %
$n = 10000$	SC	-	-	-	-	0 (0)	1967.5 (9.8)	100
$K = 5$	SONNET	0.0001	41	160	0	0.98 (0.01)	4.9 (0)	5.5
$p^{(intra)} = 0.2$	+				2	0.29 (0.01)	11.2 (0)	10.1
$p^{(inter)} = 0.05$	SC				5	0.01 (0.00)	19.5 (0.1)	16.5
	$\left(O(n^3)\right)$	0.0005	17	276	0	**0 (0)**	**4.1 (0)**	11.0
					2	0 (0)	9.4 (0)	21.2
					5	0 (0)	16.4 (0)	34.3
		0.001	12	400	0	0 (0)	4.7 (0)	15.5
					2	0 (0)	10.0 (0)	29.0
					5	0 (0)	17.7 (0)	45.3
		0.005	5	905	0	0 (0)	14.3 (0.1)	33.8
					2	0 (0)	23.2 (0.1)	57.6
					5	0 (0)	38.0 (0.1)	78.3
		0.01	3	1309	0	0 (0)	29.8 (0.1)	49.6
					2	0 (0)	47.4 (0.3)	77.6
					5	0 (0)	74.1 (0.3)	93.4
	GALE	p		T		-	-	-
	+	50		300		14.01 (3.00)	15.2 (0.8)	-
	SC	50		1000		3.23 (1.04)	26.3 (1.1)	-
$n = 20000$	SC	-	-	-	-	5.57 (0.31)	15378.8 (107.2)	100
$K = 20$	SONNET	0.005	5	2000	0	14.97 (0.08)	**112.2 (0.4)**	35.2
$p^{(intra)} = 0.2$	+				10	9.04 (0.05)	414.7 (0.9)	93.0
$p^{(inter)} = 0.1$	SC				20	3.84 (0.03)	745.0 (1.3)	99.3
	$\left(O(n^3)\right)$	0.025	5	6605	0	**0.40 (0.04)**	487.8 (0.8)	64.1
					10	14.66 (0.04)	626.6 (1.1)	96.1
					20	12.24 (0.05)	754.8 (1.1)	99.6
		0.05	5	9625	0	1.75 (0.14)	932.1 (1.0)	78.55
					10	15.87 (0.07)	1003.2 (1.1)	97.7
					20	15.12 (0.10)	1091.4 (1.4)	99.8
		0.075	35	6280	0	4.86 (0.03)	1123.2 (3.4)	54.3
					10	18.33 (0.03)	1206.5 (2.8)	65.8
					20	15.48 (0.03)	1274.7 (2.9)	74.4
	GALE	p		T		-	-	-
	+	50		1000		10.15 (4.10)	442 (61)	-
	SC	50		5000		7.33 (0)	2218 (1364)	-

Table 2: SONNET with spectral clustering (SC) on networks simulated from SBM: Average error rates and computation times from 100 simulations are reported with standard deviations in the parentheses.

4.2 SONNET with spherical K-median spectral clustering on DCBM

We consider two simulation setups for DCBM. In each setup, we independently generate 100 simple networks with n nodes and each node is randomly assigned to one of the K communities. Given a degree vector $\psi \in \mathbb{R}^n$, edges are generated randomly with probability $\psi_i \psi_j p^{(intra)}$ for two nodes i, j in the same community, and with probability $\psi_i \psi_j p^{(inter)}$ for two nodes i, j in different communities.

For each network, we run SONNET with spherical K-median spectral clustering on the Laplacian matrix. We match the best permutation of the outputs of SONNET with the true membership of the nodes to compute the error rate. We choose the parameters $\tilde{s}$ and $\tilde{o}$ as in the SBM case and try multiple values of r. In the first setup, we take $n = 10000$ and $K = 5$. In the second setup, we consider $n = 20000$ and $K = 20$. In both cases, we have $p^{(intra)} = 0.1$ and $p^{(inter)} = 0.03$, with ψ randomly selected between 1 and 100 for each node. The results are presented in Table 3. For $n = 10000$, SONNET achieves 3% error rate in only 4 seconds and 0.2% error rate in 76 seconds (highlighted rows in Table 3), compared to 0.09% error rate in 2068 seconds for spherical K-median spectral clustering on the whole network. GALE achieves 0.9% error rate in 784 seconds in the same setting.

DCBM Specifications	Method	Parameter Selection q	$\tilde{s}$	$\tilde{o}$	r	Error Rate %	Comp. Time in sec.	Exp. Data Use %
$n = 10000$	SSC	-	-	-	-	0.09 (0)	2068.2 (8.62)	100
$K = 5$	SONNET	0.0001	41	160	0	7.07 (0.04)	5.1 (0)	5.5
$p^{(intra)} = 0.1$	+				2	5.31 (0.02)	11.9 (0)	10.1
$p^{(inter)} = 0.03$	SSC				5	3.20 (0.02)	21.1 (0.0)	16.5
$\psi \sim U[1, 100]$	$(O(n^3))$	0.0005	17	276	0	3.04 (0.02)	**4.0 (0)**	11.0
					2	2.09 (0.01)	9.6 (0)	21.2
					5	1.22 (0.01)	17.1 (0)	34.3
		0.001	12	400	0	2.07 (0.01)	4.7 (0)	15.5
					2	1.40 (0.01)	10.4 (0)	29.0
					5	0.82 (0.01)	18.4 (0.1)	45.3
		0.005	5	905	0	0.73 (0.01)	14.7 (0.1)	33.8
					2	0.52 (0.01)	24.1 (0.1)	57.6
					5	0.32 (0.01)	38.8 (0.1)	78.3
		0.01	3	1309	0	0.39 (0.01)	30.1 (0.1)	49.6
					2	0.28 (0.01)	49.1 (0.2)	77.6
					5	**0.20 (0.01)**	75.8 (0.3)	93.4
	GALE	p		T		-	-	-
	+	50		300		5.00 (1.00)	38 (1.0)	-
	SC	50		1000		0.90 (0)	784 (237)	-
$n = 20000$	SSC	-	-	-	-	0.76 (0)	15796.2 (98.1)	100
$K = 20$	SONNET	0.005	5	2000	0	3.72 (0.01)	**109.2 (0.3)**	35.2
$p^{(intra)} = 0.1$	+				10	1.67 (0)	398.8 (0.9)	93.0
$p^{(inter)} = 0.03$	SSC				20	**1.33 (0)**	712.3 (1.3)	99.3
$\psi \sim U[1, 100]$	$(O(n^3))$	0.025	5	6605	0	1.92 (0.01)	473.2 (0.8)	64.1
					10	1.76 (0.01)	600.2 (0.9)	96.1
					20	1.59 (0)	745.4 (1.1)	99.6
		0.05	5	9625	0	1.43 (0)	936.4 (1.2)	78.55
					10	1.81 (0)	998.4 (1.1)	97.7
					20	1.70 (0.01)	1087.3 (1.4)	99.8
		0.075	35	6280	0	3.04 (0.01)	1129.5 (3.0)	54.3
					10	10.41 (0.02)	1195.5 (2.6)	65.8
					20	7.97 (0.02)	1254.1 (2.6)	74.4
	GALE	p		T		-	-	-
	+	50		1000		15.2 (11.0)	1302 (901)	-
	SC	50		5000		7.7 (2.0)	3483 (1964)	-

Table 3: SONNET with spherical K-median spectral clustering (SSC) on networks simulated from DCBM: Average error rates and computation times from 100 simulations are reported with standard deviations in the parentheses.

For $n = 20000$, the lowest error rate of 1.33% for `SONNET` is achieved with 20 repetitions in 712 seconds (highlighted rows in Table 3), compared to 0.76% error rate in 15796 seconds for spherical K-median spectral clustering on the whole network. `GALE` attains 7.7% error rate in 3483 seconds. `SONNET` reaches an error rate comparable to the whole network only in a fraction of the time, and is faster and more accurate than `GALE` in both simulation settings.

4.3 Real Data: DBLP Four-Area Network

The DBLP four-area network was curated by Gao et al. (2009) and Ji et al. (2010) and previously analyzed by Sengupta and Chen (2015). The data consists of $n = 4057$ data mining researchers from the research areas of database, data mining, information retrieval, and artificial intelligence. Two nodes are connected if the authors have presented at the same conference, and there are $K = 4$ ground-truth communities representing the research areas. We model the network using the SBM and apply spectral clustering on the whole network to set a benchmark of the error rate and the computation time. We then apply `SONNET` with spectral clustering with the optimized parameters as in the SBM simulation setting. Table 4 summarizes results from spectral clustering, `SONNET`, and `GALE`.

Table 4 shows that SONNET is slightly more accurate than spectral clustering on the whole network with a significantly smaller runtime. It achieves an error rate of 9.51% in 20.5 seconds (highlighted row in Table 4) compared to 9.86% for spectral clustering in 351.8 seconds. SONNET attains a slightly higher error rate of 9.98% in only 2.9 seconds. GALE achieves an error rate of 10.03% in 22.1 seconds for this real data example.

4.4 Real Data: Twitch Gamers Social Network

We also analyzed the Twitch Gamers Social Network (Sarkar and Rózemberczki, 2021) with $n = 32407$ Twitch users from $K = 20$ language communities (see detailed data description in Section S2.7 of the Supplementary Material). Two Twitch users are connected by an edge if they have a mutual follower. We apply spectral clustering with row normalization (SC+RN) on the entire network and SONNET with SC+RN to recover the language communities. Due to the large size of the network, we did not try repetitions for this example. We also did not run GALE due to computational constraints. From Table 4, spectral clustering with row-normalization on the entire network took about 29 hours with an error rate of 4.51%. SONNET achieves 7.54% error (highlighted in Table 4) in only 31.3 minutes. This example shows that SONNET is a practical choice for large networks with unbalanced communities.

Network Specifications	Method	Parameter Selection				Error Rate %	Comp. Time in sec.	Exp. Data Use %
		q	$\tilde{s}$	$\tilde{o}$	r			
DBLP	SC	-	-	-	-	9.86	351.8 (0.6)	100
Network	SONNET				0	10.34 (0.07)	5.6 (0)	6.7
$n = 4057$	+	0.0001	59	104	2	**9.51** (**0.03**)	20.5 (0.1)	9.8
$K = 4$	SC				0	10.03 (0.05)	3.0 (0)	11.1
		0.0005	17	113	2	9.78 (0.03)	7.0 (0)	21.2
					0	9.98 (0.05)	**2.9** (**0**)	15.8
		0.001	12	169	2	9.89 (0.04)	6.2 (0.1)	29.3
					0	10.08 (0.06)	4.0 (0.1)	33.8
		0.005	5	367	2	9.94 (0.07)	6.4 (0.1)	57.6
	GALE	p		T		-	-	-
	+	20		300		13.34 (0.07)	9.7 (0)	-
	SC	20		1000		10.03 (0.03)	22.1 (0.2)	-
Twitch Gamers	SC+RN	-	-	-	-	4.51	104653.1	100
Network	SONNET	0.001	8	6007	0	13.98 (0.07)	780.9 (0.8)	41.9
$n = 32407$	+	0.01	5	6002	0	13.21 (0.05)	856.8 (1.1)	46.9
$K = 20$	SC+RN	0.025	2	8425	0	**7.54** (0.02)	1880.9 (3.2)	72.6
		0.05	8	12879	0	9.02 (0.01)	3283.1 (5.7)	68.2

Table 4: SONNET with spectral clustering on the DBLP four-area network and the Twitch network: Average error rates and computation times from 100 simulations are reported with standard deviations in the parentheses.

5. Discussion

In this paper, we developed SONNET, a highly versatile and scalable algorithm for community detection in large networks, which pairs with any

community detection algorithm. We theoretically studied the statistical error bounds and computational scalability of `SONNET` under a general setting, and further specialized our study to spectral clustering on SBMs and spherical K-median spectral clustering on DCBMs. We carried out detailed empirical experiments with simulated and real-world network datasets. These theoretical and empirical results show that `SONNET` achieves substantial computational savings with minimal loss of accuracy compared to community detection on the full network. Furthermore, `SONNET` is also found to be faster and more accurate than other divide and conquer methods.

Rate-optimal community detection methods under the SBM and the DCBM have been proposed by Gao et al. (2017) and Gao et al. (2018), respectively. Both papers point out that applying a clustering technique as an initialization step, followed by a refinement step, can lead to rate-optimal community detection as long as the initial clustering satisfies certain weak consistency conditions. We conjecture that `SONNET` could be used as a computationally efficient method for the initial step, and rate-optimality will be preserved after refinement under certain conditions. This will be an interesting direction for future research.

Supplementary Materials

The supplementary material contains the definitions and the statements of some of the relevant methods and results, as well as technical proofs.

References

Airoldi, E., Blei, D. M., Fienberg, S., and Xing, E. (2008), "Mixed Membership Stochastic Blockmodels," *Journal of Machine Learning Research*, 9, 1981–2014.

Amini, A. A., Chen, A., Bickel, P. J., and Levina, E. (2013), "Pseudo-likelihood Methods for Community Detection in Large Sparse Networks," *Annals of Statistics*, 41, 2097 – 2122.

Cao, Y. and Chen, D.-R. (2011), "Consistency of Regularized Spectral Clustering," *Applied and Computational Harmonic Analysis*, 30, 319–336.

Chen, W.-Y., Song, Y., Bai, H., Lin, C.-J., and Chang, E. Y. (2011), "Parallel Spectral Clustering in Distributed Systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, 33, 568–586.

Chen, Y., Sanghavi, S., and Xu, H. (2012), "Clustering Sparse Graphs," in *Proc. of 25th Int. Conf. on Neural Information Processing Systems - Vol 2*, p. 2204–2212.

Erdös, P. and Rényi, A. (1959), "On Random Graphs I," *Publ. Math. Debr.*, 6, 290–297.

Gao, C., Ma, Z., Zhang, A. Y., and Zhou, H. H. (2017), "Achieving Optimal Misclassification Proportion in Stochastic Block Models," *Journal of Machine Learning Research*, 18, 1980–2024.

— (2018), "Community Detection in Degree-Corrected Block Models," *The Annals of Statistics*,

46, 2153–2185.

Gao, J., Liang, F., Fan, W., Sun, Y., and Han, J. (2009), "Graph-based Consensus Maximization among Multiple Supervised and Unsupervised Models," in *Advances in Neural Information Processing Systems 22*, pp. 585–593.

Gastner, M. T. and Newman, M. E. J. (2006), "Shape and Efficiency in Spatial Distribution Networks," *Journal of Statistical Mechanics: Theory and Experiment*, 2006, 1015.

Goldenberg, A., Zheng, A. X., Fienberg, S. E., and Airoldi, E. M. (2010), "A Survey of Statistical Network Models," *Foundations and Trends in Machine Learning*, 2, 129–233.

Guo, Z., Cho, J.-H., Chen, R., Sengupta, S., Hong, M., and Mitra, T. (2020), "Online Social Deception and its Countermeasures: A Survey," *IEEE Access*, 9, 1770–1806.

Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983), "Stochastic Blockmodels: First Steps," *Social Networks*, 5, 109 – 137.

Ji, M., Sun, Y., Danilevsky, M., Han, J., and Gao, J. (2010), "Graph Regularized Transductive Classification on Heterogeneous Information Networks," in *Machine Learning and Knowledge Discovery in Databases*, Berlin, Heidelberg: Springer, pp. 570–586.

Karrer, B. and Newman, M. E. J. (2011), "Stochastic Blockmodels and Community Structure in Networks," *Physical Review E*, 83, 016107–01617.

Karypis, G. and Kumar, V. (1998), "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing*, 20, 359–392.

Kumar, B. V. P. (2017), "Efficient Community Detection for Large Scale Networks via Sub-

sampling," *Master's Thesis*, Virginia Tech.

Le, C., Levina, E., and Vershynin, R. (2014), "Optimization via Low-rank Approximation for Community Detection in Networks," *The Annals of Statistics*, 44, 373–400.

Lei, J. and Rinaldo, A. (2015), "Consistency of Spectral Clustering in Stochastic Block Models," *Annals of Statistics*, 43, 215–237.

Leitch, J., Alexander, K. A., and Sengupta, S. (2019), "Toward Epidemic Thresholds on Temporal Networks: a Review and Open Questions," *Applied Network Science*, 4, 105.

Mukherjee, S. S., Sarkar, P., and Bickel, P. J. (2021), "Two Provably Consistent Divide-and-Conquer Clustering Algorithms for Large Networks," *Proceedings of the National Academy of Sciences*, 118, e2100482118.

Nowicki, K. and Snijders, T. A. B. (2001), "Estimation and Prediction for Stochastic Blockstructures," *Journal of the American Statistical Association*, 96, 1077–1087.

Pan, V. Y. and Chen, Z. Q. (1999), "The Complexity of the Matrix Eigenproblem," in *Proc. of 31st Annual ACM Symposium on Theory of Computing*, p. 507–516.

Rohe, K., Chatterjee, S., and Yu, B. (2011), "Spectral Clustering and the High-dimensional Stochastic Blockmodel," *Annals of Statistics*, 39, 1878–1915.

Roncal, W. G., Koterba, Z. H., Mhembere, D., Kleissas, D. M., Vogelstein, J. T., Burns, R., Bowles, A. R., Donavos, D. K., Ryman, S., and Jung, R. E. (2013), "MIGRAINE: MRI Graph Reliability Analysis and Inference for Connectomics," in *2013 IEEE Global Conference on Signal and Information Processing*, IEEE, pp. 313–316.

Sarkar, P. and Bickel, P. J. (2015), "Role of normalization in spectral clustering for stochastic blockmodels," *The Annals of Statistics*, 43, 962 – 990.

Sarkar, R. and Rózemberczki, B. (2021), "Twitch Gamers: a Dataset for Evaluating Proximity Preserving and Structural Role-based Node Embeddings," in *Workshop on Graph Learning Benchmarks@ TheWebConf 2021*.

Sengupta, S. and Chen, Y. (2015), "Spectral Clustering in Heterogeneous Networks," *Statistica Sinica*, 25, 1081–1106.

— (2018), "A Block Model for Node Popularity in Networks with Community Structure," *Journal of the Royal Statistical Society, Series B*, 80, 365–386.

Yan, D., Huang, L., and Jordan, M. I. (2009), "Fast Approximate Spectral Clustering," in *Proc. of 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, p. 907–916.

Yang, W. and Xu, H. (2015), "A Divide and Conquer Framework for Distributed Graph Clustering," in *Proc. of 32nd Int. Conf. on Machine Learning - Vol 37*, p. 504–513.

Zhao, Y., Levina, E., and Zhu, J. (2012), "Consistency of Community Detection in Networks Under Degree-corrected Stochastic Block Models," *Annals of Statistics*, 40, 2266 – 2292.

Sayan Chakrabarty, Department of Statistics, University of Illinois at Urbana-Champaign
E-mail: sayanc3@illinois.edu

Srijan Sengupta, Department of Statistics, North Carolina State University
E-mail: ssengup2@ncsu.edu

Yuguo Chen, Department of Statistics, University of Illinois at Urbana-Champaign
E-mail: yuguo@illinois.edu