Community Detection on Ego Networks

Nicolas Kim & Alessandro Rinaldo
Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213
nicolask@stat.cmu.edu, arinaldo@cmu.edu

December 25, 2017

Abstract

An ego network is a subgraph induced by the neighbors of a given node. In this article, we describe the sampling bias induced by sampling such a subgraph, and show that spectral clustering may fail to find a useful clustering as a result of this bias. A simple approach based on degree clustering is shown to outperform spectral clustering when there are many communities. These results assume the modeled graph follows the Stochastic Block Model, which is the standard modeling assumption for the community detection problem.

1 Introduction

Facebook claims to have 1.79 billion active monthly active users. For Twitter, this figure is over 300 million. These massive social networks present an opportunity to understand social connectivity on an unprecedented scale. Of particular interest to social networking companies is the potential to add compelling features for users and for advertisers. Clustering users is one way of extracting relevant information from the network; however, these networks evolve rapidly, which necessitates constant updating of these clusters in order to keep user and ad metrics as relevant as possible.

Clustering for the stochastic block model (SBM) has been thoroughly studied. \cite{1,22}; the canonical approach is to perform spectral clustering on some form of the graph Laplacian. Theoretical results by \cite{12,17} establish statistical consistency for this algorithm, but it is $O(|V|^3)$ and thus not suited for clustering networks with billions of nodes. Approximations to the community detection task have been proposed, notably in \cite{21} with statistical guarantees based on a perturbation analysis, but this approximation is still unable to cluster a network with billions of nodes. Spectral clustering has been shown to be parallelizable by \cite{7}, and very recent results can cluster a billion-node graph on the order of
several hours \[20\] using a label-propagation method, but cluster quality has not been shown to be statistically guaranteed in either case.

The difficulty of simultaneously clustering the whole network has motivated a new class of methods. Local methods for network clustering are a recent breed of algorithms which are inspired by the idea that one rarely needs to cluster an entire billion-node network all at once. Rather, the small neighborhood of a single node of interest should contain a lot of the information relevant to that particular node. The development of algorithms that can reliably and instantaneously determine the community structure around specific individuals in massive networks has been driven by \[19, 4, 3, 15, 13, 18, 9\]. Such results have generally come from the theoretical computer science community, and as such, cluster quality comes in the form of guarantees on the volume of the identified partition, or the conductance of the cut. These are traditionally graph-theoretic properties for which there are currently few translations to properties of common statistical graph models, such as the SBM \[11\].

Along the same lines as these approaches, we consider local neighborhoods of graphs. In particular, we look at ego networks chosen from graphs that can be modeled by SBMs, and try to determine the set of nodes in the ego network which are likely to belong to the same community as the ego node. We describe the sampling bias induced by taking this kind of edge-induced subgraph, and how this bias can present difficulties for standard spectral-based approaches to clustering. We compare spectral clustering with a degree-clustering algorithm dubbed Mutual-Friends and prove statistical consistency for it. Some of the results are analogous to those in \[6\], where the friend counts (rather than mutual friend counts) are shown to separate the network communities in the case where the whole network is simultaneously clustered; however, in this article, we are particularly interested in the regime where one community is significantly larger than the others, and the clustering task is slightly different since we are not interested in identifying all of the communities.

## 2 The Mutual-Friends Algorithm

The Mutual-Friends algorithm (Algorithm \[1\], partitions any given node’s list of connections. This task corresponds to classifying which nodes in a given ego network belong to the same community as the ego node. The more general task of finding communities within ego networks is tackled by \[14\], but they utilize node covariates and do not provide statistical guarantees. Our method only requires the degrees of the nodes in the ego network, using an approach adapted from \[6\], which estimates the community memberships of a whole network drawn from a SBM using only the degrees of the nodes. Our algorithm exploits the fact that in an SBM, it is usually the case that \(\pi_{ii} \gg \pi_{ij} \forall j \neq i\); this makes the partition boundary considerably more learnable.

---

\(^1\)The ego network is the subgraph induced by including only the nodes that are connected to a chosen, fixed, ego node. If the whole network is \(G = (V, E)\), we denote the ego node by \(e \in V\). The neighbors of \(e\) are also referred to as alters.
Algorithm 1: Mutual-Friends

**Data:** Ego node \( e \); mutual friend counts \((D^*_i)_i\) for all neighbors \( i \in \text{neighbors}(e) \); and a one-dimensional clustering algorithm \texttt{cluster} that outputs integer classes

**Result:** Ideally, the maximal set of nodes \( V_e \) such that \( Z_i = Z_e \) and \( A_{ie} = 1 \) for all \( i \in V_e \)

```
begin
\hat{Z} \leftarrow \text{cluster}((D^*_i)_i), \text{ num_clusters = 2});
m_0 \leftarrow \text{mean}((D^*_i)_i; \hat{Z}_i=0));
m_1 \leftarrow \text{mean}((D^*_i)_i; \hat{Z}_i=1));
if m_0 > m_1 then
\text{return} \{v \in \text{neighbors}(e) : \hat{Z}_v = 0\};
else
\text{return} \{v \in \text{neighbors}(e) : \hat{Z}_v = 1\};
end
end
```

Figure 1 illustrates how the observed mutual friend count distribution is obtained from an example whole network, which is generated according to the following procedure:

1. The whole network is sampled from an SBM with parameters \( \pi_{11} = \pi_{22} = 0.5 \), \( \pi_{12} = 0.1 \), and \( \alpha = (0.5, 0.5) \). This corresponds to two balanced and equivalent communities, denoted by orange and green.
2. An ego node is chosen, which happens to be green, and its alters are highlighted in red and blue, corresponding to the orange and green communities, respectively.
3. The ego network is the subgraph induced by the red and blue nodes.
4. The mutual friend counts correspond to the degrees of the nodes in the ego network.

### 2.1 Terminology

We consider the case of an undirected social network with \( n \) nodes, \( G = (V, E) \). This network is drawn from an SBM with \( Q \) true communities, with the symmetric block tie probability matrix \([\pi_{ij}]_{1 \leq i, j \leq Q} \). We let \((Z_1, \ldots, Z_n) \in [Q]^n\) denote the vector of community assignments, and \( Z_i \overset{\text{iid}}{\sim} \text{Multinomial}(\vec{\alpha}; 1) \), where \( \vec{\alpha} = (\alpha_1, \ldots, \alpha_Q) \in [0, 1]^Q \) such that \( \sum_q \alpha_q = 1 \) is the vector containing the probability any node belongs to each community. In other words, \( P(Z_i = q) = \alpha_q \) for all
Figure 1: Even though the community structure in the whole network is totally symmetric, the ego network is unbalanced since nodes from the same community as the ego are more likely to be in the ego network.

We assume that $\pi_{ii} > \pi_{ij}$ for all $j \neq i$; this condition is commonly referred to as assortativity.

We adopt a modification of the notation used by [6] for the ego network case. By way of comparison, recall that [6] denote the degree of a given node $i$ by $D_i := \sum_{j \neq i} A_{ij}$. The normalized degree is $T_i := D_i / (n - 1)$. Then, the largest deviation of any node’s normalized degree from its expected value is

$$d_n := \max_{q \in [Q]} \sup_{i \in [n]: Z_i = q} |T_i - \bar{\pi}_q|.$$

Note that these degrees are exchangeable (but not independent). They can be thought of as being sampled from a mixture distribution, with the mean of each component of the mixture corresponding to the appropriate $\bar{\pi}_q$.

The preceding definitions are used by [6] to prove the consistency of an algorithm that partitions nodes based on their empirical degrees, in the global case where the whole network is studied. We need to redefine certain terms so they pertain to the local case. First, let $D_i^* := \sum_{j \in [n] \setminus \{e, i\}} A_{ie} A_{ij} A_{je}$, so that

$$T_i^* := \frac{D_i^*}{\sum_{i \in [n] \setminus \{e, i\}} A_{ie}}.$$

Then,

$$d_n^* := \max_{s \in [Q]} \left[ \sup_{i \in [n] \setminus \{e\}: Z_i = s, A_{ie} = 1} |T_i^* - \bar{\pi}_{r,s}| \right],$$

where $\bar{\pi}_{r,s} := \mathbb{P}(A_{ij} = 1 | A_{ie} = 1, A_{je} = 1, Z_e = r, Z_i = s)$. $d_n^*$ is visualized in an example in Figure 2.
Lemma 1. \( \hat{\pi}_{rs} \) can be expressed as

\[
\hat{\pi}_{rs} = \sum_{t \in [Q]} \pi_{st} \frac{\pi_{rt} \alpha_t}{\sum_{u \in [Q]} \pi_{ru} \alpha_u}.
\]

This follows from [10], as the ego network is an SBM, just with an adjusted \( \alpha \) vector. That adjustment accounts for the difference between \( \hat{\pi}_{rs} \) and \( \bar{\pi}_q \).

3 Theoretical Analysis of Mutual-Friends

The motivation for the main theorem comes from the following observation: clearly, if each mutual friend count is close enough to its corresponding \( \hat{\pi} \), the gap between the ego’s community and any other community should be easily found by using a one-dimensional clustering algorithm. [6] provides guarantees for this task when the (\( Q - 1 \))-largest gaps are used to cluster the \( Q \) communities in the whole (not ego) network.

Theorem 1. As long as \( 0 < t = O(\sqrt{\log n/n}) \),

\[
P(d^*_n > t) \xrightarrow{n \to \infty} 0.
\]

The theorem provides a guarantee for the concentration of the mutual friend counts. In order to use this theorem effectively, an additional assumption must be made on the values of \( \hat{\pi}_{zs} \). Of course, \( \hat{\pi}_{zs} \) should not equal \( \hat{\pi}_{zs} \) for any community \( s \neq Z_e \); otherwise, nodes in community \( s \) will be assumed to be in the same community as the ego. Furthermore, if \( \hat{\pi}_{Z_e Z_e} < \hat{\pi}_{Z_e s} \) for any community \( s \neq Z_e \), \( s \) will be taken to be the ego’s community rather than \( Z_e \).
3.1 Conditions on $\pi$ and $\alpha$

It turns out that the concerns in the preceding paragraph are easily avoided. To see this, assume for simplicity of notation that the block matrix $\pi$ is of the form $\pi_{ss} = w$ and $\pi_{rs} = b$ for all $w \neq s$, i.e. all of the within-block probabilities are $w$, and all of the between-block probabilities are $b$ (similar results can be found without this assumption, but for brevity we omit them). This means that

$$\hat{\pi}_{Ze_s} = 1 \bar{\pi}_{Ze_s} \left[ \left( \sum_{t \in \{Q\}\setminus\{Z_e\}} \pi_{Ze_t} \pi_{st} \alpha_t \right) + \pi_{Ze_s} \left( \alpha_{Ze_e} + \pi_{ss} \alpha_s \right) \right]$$

and

$$\hat{\pi}_{Z_eZ_e} = 1 \bar{\pi}_{Z_eZ_e} \left[ \left( \sum_{t \in \{Q\}\setminus\{Z_e\}} \pi^2_{Ze_t} \alpha_t \right) + \pi^2_{Z_eZ_e} \alpha_{Z_e} \right].$$

Then, the necessary and sufficient conditions for the ego’s community to have the highest expected mutual friend count are

$$\left\{ \begin{array}{l} \hat{\pi}_{ZeZ_e} - \hat{\pi}_{Z_e1} > 0 \\ \vdots \\ \hat{\pi}_{ZeZ_e} - \hat{\pi}_{ZeQ} > 0 \end{array} \right\} \iff \left\{ \begin{array}{l} (w - b)(w \alpha_{Ze_e} - b \alpha_1) > 0 \\ \vdots \\ (w - b)(w \alpha_{Ze_e} - b \alpha_Q) > 0 \end{array} \right\} \iff \left\{ \begin{array}{l} \alpha_{Ze_e} > \frac{b}{w} \alpha_1 \\ \vdots \\ \alpha_{Ze_e} > \frac{b}{w} \alpha_Q. \end{array} \right\}$$

Assuming an assortative SBM (i.e. $w > b$), the ego community can be quite small in the overall network. Often, $w$ is taken such that $w \gg b$, so that the ego community will generally be identifiable from the mutual friend counts. Note that even if $\alpha'_{Ze_e} = \alpha'_{s}$ for some $s \neq Z_e$, the mutual friend counts are still expected to be higher for the ego’s community, since $\pi_{ZeZ_e} > \pi_{Z_e}s$. This property endows the Mutual-Friends algorithm with a kind of robustness, as long as the whole network can be adequately modeled with an SBM.

Once the mutual friend counts have been obtained, any reasonable one-dimensional clustering algorithm should identify the ego’s community, up to the above conditions. As mentioned earlier, in [6], a $(Q - 1)$-largest gaps algorithm is analyzed. In this local clustering case, finding the single-largest gap should, in many cases, identify the ego community. In practice, $k$-means with $k = 2$ is more robust to random deviations from the cluster means compared to using the largest gap. The simulation results describe this in more depth.

3.2 Perfect Recovery of the Within-community Set

The necessary and sufficient conditions for $\hat{\pi}_{ZeZ_e} > \hat{\pi}_{Z_e}s$ to hold are described above, but these conditions do not guarantee that the Mutual-Friends algorithm will perform well if the clustering is done by taking the largest gap in the mutual friend counts. In fact, even if the mutual friend counts are clustered perfectly (with zero variance) around their cluster means, there is still a way of setting the
parameters $\pi$ and $\alpha$ such that the largest-gap algorithm will fail to return the correct ego community. The following corollary provides a necessary condition for Theorem 1 to bound the probability of the largest-gap algorithm making a mistake.

**Corollary 1.** If

$$\hat{\pi}_{Z_e} - \max_{q \neq Z_e} \hat{\pi}_{Z_e q} > \max_{q \neq Z_e} \hat{\pi}_{Z_e q} - \min_{q \neq Z_e} \hat{\pi}_{Z_e q},$$

then as $n \to \infty$, the probability of the largest-gap algorithm making any mistake goes to zero.

## 4 Simulation Study

In this section, we present two simulations which demonstrate the potential difficulty in performing community detection when $Q$ is large. The first simulation describes a comparison between **Mutual-Friends** and spectral clustering with $K = 2$, which illustrates a scenario where spectral clustering can fail if $Q$ is large, for ego networks drawn from a global SBM model.

The second simulation compares the performance of degree clustering to clustering graph spectra when there are unbalanced community sizes. In this case, simulations are performed with $n$ fixed and $Q$ increasing, and in addition, we include spectral clustering with $K$ set to be the correct number of communities (the number of communities which are observed in the ego network). This simulation demonstrates that using spectral clustering with the theoretically-correct number of communities can lead to worse performance than using $K = 2$, although there are currently no theoretical guarantees to support this observation.

### 4.1 Spectral Clustering with $K = 2$

The theoretical results show that the **Mutual-Friends** algorithm works well even for smaller networks. To verify the practical utility of the algorithm, we compare it to a localized variant of spectral clustering on the graph Laplacian, which is the most common algorithm used for graph clustering. Spectral clustering takes the number of communities as a parameter, $K$; however, in this case, we cannot use $K = Q$, the number of communities in the SBM, since there may be fewer than $Q$ communities in the ego network. Hence, we use $K = 2$ in these simulations. As a benchmark, we compare these to spectral clustering where $K$ is set to be the true number of communities in the ego network, even though this would be a difficult parameter to estimate in real networks, since the true number of communities is large and many communities may have only one node.

For these simulations, we sample graphs from different SBMs. In all simulations, $\pi_{qq} = 0.1$ for all $q$ and $\pi_{qr} = 0$ for all $q \neq r$. This corresponds to the fully symmetric block matrix with $\psi = 0.1$ and $b = 0.01$. We vary $Q$, to depict the relationship between the two methods when there are few communities in the network vs. when there are many communities. As $Q$ varies, we set $\alpha = (1/Q, \ldots, 1/Q)$.
Figure 3: Blue lines depict the accuracy of Mutual-Friends. Green corresponds to spectral clustering on the graph Laplacian with $K = 2$. When the number of clusters is small (e.g. $Q = 2, 3$), Mutual-Friends performs worse than spectral clustering until the network is large enough, after which it slightly outperforms spectral clustering. Once $Q = 10$, Mutual-Friends and spectral clustering perform similarly. When the number of communities increases to $Q = 20$, Mutual-Friends outperforms spectral clustering. Figure 4 contains the FPR and FNR plots.

This corresponds to balanced community sizes. Figure 3 summarizes these results. The simulated whole-network sizes are $n = 500, 1000, 2000, \ldots, 20000$, with every 1000-node increment between 1000 and 20000. For each combination of network size $n$ and number of communities $Q$, 50 samples are drawn from the associated SBM. The 50 accuracy values for each method are then averaged and connected to create the plots. The corresponding plots for the false positive rate (FPR) and false negative rate (FNR) are given in Figure 4.

The simulations show that Mutual-Friends performs similarly to spectral clustering for small numbers of communities once $n$ is large. As the number of communities increases, Mutual-Friends also outperforms spectral clustering for small $n$.

One important reason to consider using a degree-clustering approach is speed. Our implementation of the Mutual-Friends algorithm in Python is typically at least an order of magnitude faster than the SciPy implementation of spectral clustering [10] when $K = 2$. Spectral clustering is slower with larger $K$, so if the observed data fits the SBM model, it may be worth using Mutual-Friends instead. If a largest-gap approach to clustering is used, the degree sorting stage may be done in time linear to the size of the ego network using radix sort since an upper bound on the mutual friend counts is given by the size of the ego network.

Figure 4 demonstrates that Mutual-Friends at least matches the performance of spectral clustering on large networks. As for FPR, spectral clustering performs better when $Q = 2$, because we are setting the number of communities to $K = 2$ for spectral clustering. For larger values of $Q$, Mutual-Friends is either comparable to or outperforms spectral clustering, although $K$ is misspecified. In comparison, [6] found that degree clustering did not work well in small graphs simulated from three relatively balanced communities, since the tails of the per-community degree distributions mix for small $n$. In our case, the assumption $\pi_{ii} \gg \pi_{ij}$ boosts the signal for the particular task we are interested in.
Figure 4: In the top four plots, blue lines depict the FNR (proportion of incorrect classifications out of the nodes that are truly outside the ego community) of Mutual-Friends. Green corresponds to spectral clustering on the graph Laplacian with $K = 2$. Lower is better. The bottom four plots are the same but they depict the FPR (proportion of incorrect classifications out of the nodes that are truly inside the ego community).

4.2 Spectral Clustering with $K$ Set to the Correct Number of Communities

For this simulation, $Q$ ranges from 2 to 20. The size of each community is given by

$$\alpha = \left( \frac{1}{2}, \frac{1}{2(Q-1)}, \ldots, \frac{1}{2(Q-1)} \right).$$

This means that one community contains half of the nodes in the network, while the remaining $Q - 1$ communities share the remaining half of the nodes. The number of nodes is fixed at 2000 and $\pi$ is also fixed with $\pi_{ii} = w = 0.3$ and $\pi_{ij} = b = 0.2$ for all $i$ and $j \neq i$.

The task remains the same: to cluster the nodes into two groups, one of which should generally contain the largest community. Note that we are not sampling a global SBM and then clustering the ego network; instead, since ego networks are essentially SBMs with one large community and several small communities, we directly sample graphs from such an unbalanced SBM. As an aside, community detection in unbalanced SBMs has been studied by [5], but only in the case where $Q = 2$.

Figure 5 shows that as $Q$ increases, spectral clustering with $K = Q$ (technically the number of observed communities, since some communities may have zero nodes in the sampled graph) only performs about as well as random guessing; this is because for large $Q$, spectral clustering begins to cluster most of the nodes with the largest community's nodes. On the other hand, spectral clustering with $K = 2$ works well generally, and Mutual-Friends also matches its performance for larger $Q$. This is because when $Q = 2$, both communities are the same size, and the degree distribution will be the mixture of two identical binomials, making the degrees non-separable.
5 Conclusions

These results offer a framework for understanding the sampling bias induced by sampling via edges rather than nodes.

**Mutual-Friends** is introduced as a competing method within the local graph clustering literature. It utilizes the sampling bias observed in edge-induced subgraphs to obtain a good clustering result for large networks with many communities that can be modeled as an SBM. Although it solves a sub-problem of the one solved by modified PageRank methods, Algorithm 2 is proposed as a simple extension which competes directly with such methods.

Although **Mutual-Friends** has good theoretical properties and compares favorably to standard spectral-based techniques, a challenge to applying **Mutual-Friends** to real social network data is that social networks are not generally able to be modeled with a standard SBM since users do not generally belong to one community only, and the algorithm is not robust to a violation of this assumption. This means that the mutual friend counts will not be clearly separated and the algorithm will not detect an ego community.

Mixed membership SBMs (MMBs) [2] are a natural generalization of the standard SBM, wherein nodes, in some sense, belong to a mixture of communities. Deriving a mixed membership version of **Mutual-Friends** is therefore naturally of interest.

Acknowledgments

We thank Cosma Shalizi, Brian Junker, Jing Lei, Neil Spencer, Alden Green, Momin Malik, and the rest of the Networkshop group at Carnegie Mellon for many valuable discussions throughout the development of this work. This work was partially supported by AFOSR grant #FA9550-14-1-0141.
References


Appendix

Proof of Theorem 1

The proof of Theorem 1 requires a concentration-type bound for \( T_i^* \), the normalized mutual friend count. This is possible by the following lemma, regarding the unnormalized mutual friend count, \( D_i^* \).

**Lemma 2.**

\[
D_i^* := \sum_{j \in [n] \setminus \{e, i\}} A_{ij} A_{je} \mid Z_e, Z_i = s, \{A_{\ell e}\}_{\ell \neq i}, A_{ie} = 1 \sim \text{Bin} \left( \sum_{j \in [n] \setminus \{e, i\}} a_{je}, \hat{\pi}_{Z_e, s} \right).
\]

This result depends on a rather intricate set of conditional random variables and events. Several conditions must be met in order for mathematically-useful statements about ego neighbors to be made: the block membership of each neighbor should be known (as usual), but the block membership of the ego node itself should also be conditioned on (much of the behavior of the ego network is dictated by the ego node’s own block membership); in addition, the event that the neighbors are connected to the ego node has to be conditioned on.

**Proof of Lemma 2.** For each term in the sum, observe that

\[
\mathbb{P}(A_{ij} A_{je} = 1 \mid Z_e, Z_i = s, \{A_{\ell e}\}_{\ell \neq i}, A_{ie} = 1) = \begin{cases} 
0, & A_{je} = 0 \\
\mathbb{P}(A_{ij} = 1 \mid Z_e, Z_i = s, \{A_{\ell e}\}_{\ell \neq i}, A_{ie} = 1, A_{je} = 1) = \begin{cases} 
0, & A_{je} = 0 \\
\mathbb{P}(A_{ij} = 1 \mid Z_e, Z_i = s, A_{je}, A_{ie} = 1, A_{je} = 1) = \begin{cases} 
0, & A_{je} = 0 \\
\hat{\pi}_{Z_e, s}, & A_{je} = 1.
\end{cases}
\end{cases}
\end{cases}
\]

In particular, this implies that \( \{A_{ij} \mid Z_e, Z_i, \{A_{\ell e}\}_{\ell \neq i}\} \) for all \( j \in [n] \setminus \{e, i\} : A_{ie} = 1 \) are identically and independently distributed (since they are \( d \)-separated) as Bernoulli random variables with probability \( \hat{\pi}_{Z_e, s} \).

The distribution of \( D_i^* \) conditional on the above conditioning set is therefore

\[
\mathbb{P}(D_i^* = k \mid Z_e, Z_i = s, \{A_{\ell e}\}_{\ell \neq i}, A_{ie} = 1) = \mathbb{P} \left( \sum_{j \in [n] \setminus \{e,i\}} A_{ij} A_{je} = k \mid Z_e, Z_i = s, \{A_{\ell e}\}_{\ell \neq i}, A_{ie} = 1 \right)
\]

\[
= \mathbb{P} \left( \sum_{j \in [n] \setminus \{e,i\}, A_{je}=1} A_{ij} = k \mid Z_e, Z_i = s, \{A_{\ell e}\}_{\ell \neq i}, A_{ie} = 1 \right),
\]

which, by the preceding paragraph, is binomial with the desired parameters. \( \Box \)

We also use the fact that as the network grows, the number of nodes in each community will concentrate. We denote by \( B_q \) the number of nodes in the ego network which are in community \( q \):

\[
B_q | \{Z_i\}_i \overset{iid}{\sim} \text{Bin} \left( n_q := \sum_{i \in [n] \setminus \{e\}} 1_{z_i = q}, \pi_{z_i, q} \right).
\]
Let \( N_q \) be the (random) number of nodes in community \( q \), and \( n_q \) be realization of this random variable. Then, the simultaneous concentration of all community sizes within the ego network is
\[
\mathcal{B} := \bigcap_{q \in [Q]} \left\{ |B_q - N_q \pi_{z_q}| \leq \sqrt{n \log n} \right\}.
\]

**Lemma 3.** The probability that at least one block fails to concentrate, conditioned on the block assignments, goes to zero as \( 1/n^2 \). In other words, \[
P \left( B^C \mid \mathcal{Z}_i \right) \sim \frac{1}{n^2}.
\]

**Proof of Lemma 3.** The probability that at least one block fails to concentrate according to \( \mathcal{B} \) is:
\[
P \left( B^C \mid \mathcal{Z}_i \right) = \mathbb{P} \left( \bigcup_{q \in [Q]} \left\{ |B_q - N_q \pi_{z_q}| > \sqrt{n \log n} \right\} \mid \mathcal{Z} \right)
\leq \sum_{q \in [Q]} \mathbb{P} \left( |B_q - N_q \pi_{z_q}| > \sqrt{n \log n} \mid \mathcal{Z} \right)
\leq \sum_{q \in [Q]} 2 \exp \left\{ -\frac{2 n \log n}{n} \right\}
= 2 \frac{Q}{n^2}.
\]

**Proof of Theorem 1.** By the Law of Total Probability and the Law of Total Expectation,
\[
P \left( d_n^* > t \right) = \mathbb{E}_Z \left[ \mathbb{P} \left( d_n^* > t \mid \mathcal{B}, Z \right) \right] + \mathbb{P} \left( d_n^* > t \mid \mathcal{B}^C, Z \right) \mathbb{P} \left( \mathcal{B}^C \mid Z \right)
\leq \mathbb{E}_Z \left[ \mathbb{P} \left( d_n^* > t \mid \mathcal{B}, Z \right) \right] + \mathbb{E}_Z \left[ \mathbb{P} \left( \mathcal{B}^C \mid Z \right) \right].
\]

By Lemma 3 the second term in the expectation goes to zero independently of \( t \), since it is not a function of \( t \). Hence,
\[
P \left( d_n^* > t \right) \leq \mathbb{E}_{Z,(A)} \left[ \mathbb{P} \left( d_n^* > t \mid \mathcal{B}, Z, \{ A_{iE} \} \right) \right] + O(1)
= \mathbb{E}_{Z,(A)} \left[ \mathbb{P} \left( \bigcup_{s \in [Q]} \bigcup_{i \in [n] \setminus \{e\}} \left( \{ T^*_s - \hat{\pi}_{zs} < t \} \right) \mid \mathcal{B}, Z, \{ A_{iE} \} \right) \right] + O(1)
\]
\[
\leq \mathbb{E}_{Z,(A)} \left[ \sum_{s \in [Q]} \sum_{i \in [n] \setminus \{e\}} \mathbb{P} \left( \{ T^*_s - \hat{\pi}_{zs} < t \} \mid \mathcal{B}, Z, \{ A_{iE} \} \right) \right] + O(1)
\]
\[
= \mathbb{E}_{Z,(A)} \left[ \sum_{s \in [Q]} \sum_{i \in [n] \setminus \{e\}} 1_{Z_{ie} = \text{true}} \mathbb{P} \left( \{ T^*_s - \hat{\pi}_{zs} < t \} \mid \mathcal{B}, Z, \{ A_{iE} \} \right) \right] + O(1).
\]
Writing out the expectation as a sum,
\[
= \sum_{\{a_{iE}\} \in \{0,1\}^{n-1}} \sum_{s \in [Q]} \sum_{i \in [n] \setminus \{e\}} \sum_{s \in [Q]} 1_{Z_{ie} = \text{true}} \mathbb{P} \left( \{ T^*_s - \hat{\pi}_{zs} < t \} \mid \mathcal{B}, Z, \{ A_{iE} \} \right) \mathbb{P} \left( \{ A_{iE} \} \mid Z \right) + O(1).
\]

14
Summing over all $z_j$ for $j \neq c$,

$$= \sum_{(a_{le})_{e \in [Q]}} \sum_{s \in [n](\epsilon) \setminus \{c\}} \sum_{z_e \in [Q]} a_{le} \mathbb{P} \left( |T_i^* - \hat{\pi}_{ze, c}| > t | (B, Z_c, Z_i = s), \{ A_{le} \}_{e \in [Q]} \right) \mathbb{P} \left( (A_{le})_{e \in [Q]}, Z_c, Z_i = s \right) + O(1)$$

$$= \sum_{(a_{le})_{e \in [Q]}} \sum_{s \in [n](\epsilon) \setminus \{c\}} \sum_{z_e \in [Q]} a_{le} \mathbb{P} \left( |T_i^* - \hat{\pi}_{ze, c}| > t | (B, Z_c, Z_i = s), \{ A_{le} \}_{e \in [Q]}, A_{ie} = 1 \right) \mathbb{P} \left( (A_{le})_{e \in [Q]}, A_{ie} = 1, Z_c, Z_i = s \right) + O(1)$$

$$\leq \sum_{(a_{le})_{e \in [Q]}} \sum_{s \in [n](\epsilon) \setminus \{c\}} \sum_{z_e \in [Q]} a_{le} 2 \exp \left\{ -2 \sum_{q \in [Q]} (n \pi_{ze, q} - \sqrt{n \log n - 1})^2 \right\} \mathbb{P} \left( (A_{le})_{e \in [Q]}, A_{ie} = 1, Z_c, Z_i = s \right) + O(1).$$

We can simplify $\mathbb{P} \left( \{ A_{le} \}_{e \neq i}, A_{ie} = 1, Z_c, Z_i = s \right)$ as:

$$\mathbb{P} \left( \{ A_{le} \}_{e \neq i}, A_{ie} = 1, Z_c, Z_i = s \right) = \mathbb{P} (A_{ie} = 1 | Z_c, Z_i = s) \mathbb{P} \left( \{ A_{le} \}_{e \neq i} | Z_c, Z_i = s \right) \mathbb{P} (Z_c, Z_i = s)$$

$$= \pi_{ze, a} \left( \prod_{j \in [n] \setminus \{e, i\} s} \pi_{z_j}^a \left( 1 - \tilde{\pi}_{ze} \right)^{1-a_j} \right) \mathbb{P} (Z_c) \mathbb{P} (Z_i = s)$$

$$= \pi_{ze, s} \left( \prod_{j \in [n] \setminus \{e, i\} s} \pi_{z_j}^a \left( 1 - \tilde{\pi}_{ze} \right)^{1-a_j} \right) \alpha_{ze} \alpha_s$$

Going back to the bound of $\mathbb{P} (d_n^* > t)$, we make the substitution $k = \sum_{j \in [n] \setminus \{e, i\} s} a_{je}$:

$$\mathbb{P} (d_n^* > t) \leq \sum_{(a_{le})_{e \in [Q]}} \sum_{s \in [n](\epsilon) \setminus \{c\}} \sum_{z_e \in [Q]} a_{le} 2 \exp \left\{ -2 \left( \sum_{q \in [Q]} n \pi_{ze, q} - \sqrt{n \log n - 1} \right)^2 \right\} \pi_{ze, s} \left( \tilde{\pi}_{ze} \left( 1 - \tilde{\pi}_{ze} \right)^{n-2-k} \right) \alpha_{ze} \alpha_s + O(1)$$

$$= \sum_{k=0}^{n-1} \binom{n-1}{k} \sum_{s \in [n](\epsilon) \setminus \{c\}} (k+1)2 \exp \left\{ -2 \left( \sum_{q \in [Q]} n \pi_{ze, q} - \sqrt{n \log n - 1} \right)^2 \right\} \pi_{ze, s} \tilde{\pi}_{ze}^k \left( 1 - \tilde{\pi}_{ze} \right)^{n-2-k} \alpha_{ze} \alpha_s + O(1)$$

$$= 2 \sum_{s \in [n](\epsilon) \setminus \{c\}} \frac{\pi_{ze, s} \alpha_{ze} \alpha_s}{1 - \tilde{\pi}_{ze}} \exp \left\{ -2 \left( \sum_{q \in [Q]} n \pi_{ze, q} - \sqrt{n \log n - 1} \right)^2 \right\} \left( 1 + (n-1)\tilde{\pi}_{ze} \right) + O(1),$$

which converges to 0 as $n \to \infty$ for all $t \in O(\sqrt{(\log(n))/n})$. Notably, all of the constants are known in this bound.

**Recovering the Entire Ego Community**

Each community of a stochastic block model essentially acts as an Erdős-Renyi graph. The diameter of such graphs is known to be small \[\mathbb{E}\]. This fact suggests the naïve procedure outlined in Algorithm 2 for discovering the whole ego community using Mutual-Friends. Within 20 iterations of Mutual-Friends on a graph with 2000 nodes drawn from an SBM, we are able to recover approximately 98% of all of the nodes in the same community as the ego node in the graph. For reference, only about 22% of all of the nodes in the same community as the ego node are actually contained in the ego network, so with a small number of iterations of the Mutual-Friends algorithm we are able to recover several times more nodes in the same community as the ego node. Figure 6 illustrates several runs of Algorithm 2.
Algorithm 2: Recovering the entire community that the ego belongs to

Data: Ego node $e$; number of steps to take $N$

Result: Ideally, the maximal set of nodes $S$ such that $Z_i = Z_e$ and for all $i \in S$

begin
    $\mathcal{U} \leftarrow \{e\}$;
    $(D_i^*)_i \leftarrow \text{neighbors}(e)$;
    $S \leftarrow \text{Mutual-Friends}(e, (D_i^*)_i)$;
    for $i \in [N]$ do
        $\mathcal{V} \sim \text{Unif}(S \setminus \mathcal{U})$;
        $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathcal{V}\}$;
        $(D_i^*)_i \leftarrow \text{neighbors}(\mathcal{V})$;
        $S \leftarrow S \cup \text{Mutual-Friends}(\mathcal{V}, (D_i^*)_i)$;
    end
    return $S$;
end

Figure 6: Each line represents a single run of Algorithm 2 for increasing iteration number $N$. All runs were performed on a single sample from an SBM with parameters $w = 0.2$ and $b = 0.01$ with $\alpha = (0.5, 0.5)$. 