# Searching For A Single Community in a Graph

Avik Ray
ECE, UT Austin
avik@utexas.edu

Sujay Sanghavi
ECE, UT Austin
sanghavi@mail.utexas.edu

Sanjay Shakkottai
ECE, UT Austin
shakkott@austin.utexas.edu

## ABSTRACT

In standard graph clustering/community detection, one is interested in partitioning the graph into more densely connected subsets of nodes. In contrast, the *search* problem of this paper aims to only find the nodes in a *single* such community, the target, out of the many communities that may exist. To do so , we are given suitable side information about the target; for example, a very small number of nodes from the target are labeled as such.

We consider a general yet simple notion of side information: all nodes are assumed to have random weights, with nodes in the target having higher weights on average. Given these weights and the graph, we develop a variant of the method of moments that identifies nodes in the target more reliably, and with lower computation, than generic community detection methods that do not use side information and partition the entire graph.

## Keywords

community detection, stochastic block model, side-information

## 1. INTRODUCTION

Community detection, or graph clustering, is the classic problem of finding subsets of nodes such that each subset has higher connectivity within itself, as compared to the average connectivity of the graph as a whole. Typically, when graphs represent similarity or affinity relationships between nodes, these subsets represent communities of similar nodes. Also typically, this problem has primarily been considered in the unsupervised setting, where the only input is the graph itself and the objective is to partition all or most of the nodes.

In this paper we look at a different, but related, community detection task, which we will refer to as the *search problem*. Our objective is to use the graph to find a single community of nodes – which we will call the *target community* – for which we have been given some relevant but quite noisy side information. We would like to do so more reliably, and with lower computation, than existing methods that do not use side information.

Our motivations are two-fold: *(i)* it is often the case that the network analyst is looking for nodes with a-priori specified characteristics e.g. in military/intelligence settings, and *(ii)* it is rare that we are faced with a "pure" graph analysis problem; typically there is extra non-graphical side information that, if used properly, could make the inference task easier.

**Our contributions** are as follows.

**(i)** We develop a simple yet generic framework for how side-information is to be specified: each node is given a (possibly random) weight, with nodes in the target community having higher weight on average than nodes not in the target – we call these *biased weights*.

**(ii)** Given such biased weights, we develop a variant of the $2^{nd}$ order method of moments – to find the nodes in the target community. We call this *Community Search* algorithm.

**(iii)** Our main results characterize the effectiveness of this algorithm in finding the target community; we study this in the standard stochastic block model setting with many communities. Analytically, we show that it matches (potentially upto log factors) the analytical guarantees of the state of the art unsupervised community detection methods; empirically, we show that the method outperforms these methods even with very noisy side information (e.g. very small number of labeled nodes), and has significantly lower computational complexity.

## 2. SETTINGS AND ALGORITHM

Consider a graph $G = (V, E)$ with $n$ nodes and $k$ communities that partition the vertex set as $V = \cup_{i=1}^{k} V_i$. Let $\alpha_i = |V_i|/n$ be the fraction of nodes in the $i$-th community. The edge set $E$ is generated according to the standard stochastic block model [2] with edge probabilities $p, q$, $0 < q < p < 1$.

In the search problem we are interested in the recovery of *one* target community, say $V_1$. The side-information is in the form of *biased node weights.* Each node $j \in V$ is assigned a weight $w_j > 0$ by a random process; these weights satisfy the condition $E[w_j | j \in V_1] > E[w_j | j \in V_i]$ for all $i \neq 1$. These biased weights may be computed using a set of *labeled nodes* from the target community $\mathcal{L} \subset V_1$.

The main goal is to solve this search problem faster than the time required to recover all $k$ communities, and without any loss in estimation accuracy.

**Algorithm:** We now describe our main algorithm called **Community Search**. $X$ be the adjacency matrix of the



*SIGMETRICS '16 June 14-18, 2016, Antibes Juan-Les-Pins, France*



ACM ISBN 978-1-4503-4266-7/16/06.

DOI: http://dx.doi.org/10.1145/2896377.2901494

graph $G$. Also define community membership vectors $\mu_1, \ldots, \mu_k$ where $\mu_i \in \mathbb{R}^n$, as follows. Let $\mu_{j,i}$ be the $j-$th coordinate of vector $\mu_i$. Then $\mu_{j,i} = p$ if $j \in V_i$, $\mu_{j,i} = q$ otherwise. Note that these $\mu_i$-s are linearly independent and the community memberships of the nodes can be obtained from these membership vectors via thresholding. The aim of our algorithm is to estimate the membership vector of the first community $\mu_1$ (which can then be used to recover nodes in $V_1$). For any two subsets $P, Q \subset [n]$ let $X_{P,Q}$ denote the submatrix of $X$ corresponding to the rows and columns in set $P$ and $Q$ respectively. The input parameters to Algorithm 1 are the adjacency matrix $X$, number of communities $k$, the set of biased node weights $(w_1, \ldots, w_n)$, and a threshold $\tau$. The output is the community estimate $\hat{V}_1$.

**Algorithm 1** Community Search

**Input:** Adjacency matrix $X$, $k$, biased weights $(w_1, \ldots, w_n)$, threshold $\tau$

**Output:** $\hat{V}_1$

1: Partition nodes into four sets $P_1, P_2, P_3, P_4$ at random. Let $n_i = |P_i|$
2: Compute matrices $\hat{A}_1 = \frac{1}{\sqrt{n_3}} X_{P_1,P_3}$, $\hat{A}_2 = \frac{1}{\sqrt{n_3}} X_{P_2,P_3}$
3: Compute vector $\hat{m}_1 = \frac{1}{n_1} \sum_{j \in P_1} X_{P_1,j}$
4: Compute matrix $\hat{B} = \frac{1}{n_4} \sum_{j \in P_4} w_j X_{P_1,j} X_{P_2,j}^T$
5: $\hat{\mu}_{P_1}, \hat{\alpha}_1 \leftarrow$ SearchSubroutine$(\hat{A}_1, \hat{A}_2, \hat{B}, \hat{m}_1, k)$
6: Compute $V_{P_1} = \{j \in P_1 : \hat{\mu}_{P_1,j} > \tau\}$
7: Repeat steps 2-5 with $P_i$'s rotated in order to estimate $\hat{\mu}_{P_2}, \hat{\mu}_{P_3}, \hat{\mu}_{P_4}$. Use them to compute $V_{P_2}, V_{P_3}, V_{P_4}$. Return community $\hat{V}_1 = V_{P_1} \cup V_{P_2} \cup V_{P_3} \cup V_{P_4}$

**Algorithm 2** SearchSubroutine

**Input:** $\hat{A}_1, \hat{A}_2, \hat{B}, \hat{m}_1, k$

**Output:** $\hat{\mu}_1, \hat{\alpha}_1$

1: Compute rank $k$-svd of matrices $\hat{A}_1, \hat{A}_2 : \hat{A}_1 = U_1 D_1 V_1^T$, $\hat{A}_2 = U_2 D_2 V_2^T$
2: Compute matrices $W_1 = U_1 D_1^{-1}$, $W_2 = U_2 D_2^{-1}$
3: Let $u_1$ be the largest left singular vector of $W_1^T \hat{B} W_2$
4: Compute $z = U_1 D_1 u_1$
5: Compute $a = u_1^T W_1^T \hat{m}_1$
6: Return $\hat{\mu}_1 \leftarrow z/a$ and $\hat{\alpha}_1 \leftarrow a^2$

## 3. RESULTS

In this section we present our main results. When side information is available in the form of biased weights $w_j$ for each node $j \in V$, these weights need to be *informative* about the target community $V_1$ so that it could be recovered. We quantify this in the following set of assumptions.

*(A1) Average weight bias:* Under this condition the expected weight of a node in community $V_1$ is greater than the expected weight of a node in any other community $V_i$. Precisely the weights satisfy: $E[w_j | j \in V_1] > E[w_j | j \in V_i]$, $\forall i \neq 1$

*(A2) Weight concentration:* Let $\alpha_{max} = \max_{i \in [k]} \alpha_i$ and $\alpha_{min} = \min_{i \in [k]} \alpha_i$. Define $\sigma_1(R) := E[w_j | j \in V_1]$, $\gamma_2 := \max_{i \in [k], j \in V} |w_j - E[w_j | j \in V_i]|$, and $\xi(n) = o(\sqrt{\log n})$ be any slowly growing function. Then with high probability the maximum deviation of the weights are bounded as follows.

$$\frac{\gamma_2}{\sigma_1(R)} = O\left(\min\left\{\frac{\alpha_{min}^4 (p-q)^4}{\alpha_{max}^4 p^4 \xi(n)}, \frac{\alpha_{min}^5 \sqrt{n} (p-q)^5}{\alpha_{max}^4 p^{4.5} \xi(n)} - 1\right\}\right)$$

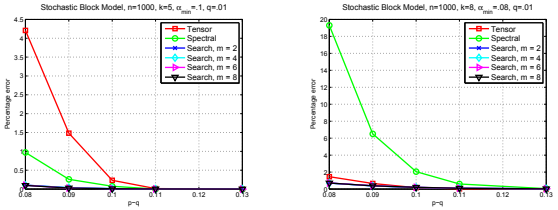
*(A3) $p, q$ separation:* Let $p, q, n$ satisfy $\frac{(p-q)^3}{p^2} = \tilde{\Omega}\left(\frac{\alpha_{max}}{\alpha_{min}^3 n}\right)$

This condition fundamentally determines when communities are identifiable in a stochastic block model and similar conditions are required for other community detection algorithms [2, 1].

Theorem 1 shows that under the above assumptions on the biased weights Algorithm 1 can reconstruct community $V_1$ with high accuracy.

THEOREM 1. *Consider a $(n, k, p, q)$ stochastic block model satisfying condition (A3). Given biased weights $(w_1, \ldots, w_n)$ satisfying conditions (A1), (A2), then Algorithm 1 recovers community $V_1$ with fraction of error nodes $o(1)$ with high probability.*

Biased weights, as required in Theorem 1, can be obtained from a small set of labeled nodes $\mathcal{L}$ as follows. Weight $w_i$ is simply the number of nodes in $\mathcal{L}$ that are at distance of $r+1$ hops from node $i$.




(a) $n = 1000$, $k = 5$ (b) $n = 1000$, $k = 8$

**Figure 1: Figure comparing the average error performance of Community Search algorithm with Spectral clustering [3] and Tensor decomposition [1] algorithms in two stochastic block models with $n = 1000$, $k \in \{5, 8\}$. We use $m$ labeled node from target cluster as side information and compute biased weights. The Community Search algorithm outperforms both Spectral clustering and Tensor decomposition.**

**Experiments:** In synthetic graphs generated using stochastic block model we observe that the Community Search algorithm exhibits greater accuracy (Figure 1) and more than 3X speedup over Spectral clustering [3] and Tensor decomposition [1] algorithms.

## 4. ACKNOWLEDGMENTS

We would like to acknowledge support from NSF grants CNS-1320175, 0954059; ARO grants W911NF-15-1-0227, W911NF-14-1-0387; and the US DoT supported D- STOP Tier 1 University Transportation Center.

## 5. REFERENCES

[1] A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade. A tensor spectral approach to learning mixed membership community models, 2013. http://arxiv.org/abs/1302.2684.

[2] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[3] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.