Overlap Graph Clustering via Successive Removal

Avik Ray[†], Javad Ghaderi[‡], Sujay Sanghavi[†], Sanjay Shakkottai[†]
University of Texas at Austin[†], and Columbia University[‡]

***Abstract*— One of the fundamental questions in the study of complex networks is community detection, i.e., given a graph that represents interactions in a real system, can we group vertices with similar interests together? In many applications, we are often in a setting where vertices may potentially belong to multiple communities. In this paper, we propose an efficient algorithm for overlapping community detection which can successively recover all the communities. We provide theoretical guarantees on the performance of the algorithm by leveraging convex relaxation and exploiting the fact that in many networks there are often vertices that only belong to one community.**

I. Introduction

The study of community structure in graphs has been of great interest in several domains (e.g., sociology, biology, computer science, machine learning). The problem can be briefly described as follows: Given a graph in which edges represent similarities among various vertices, can we group related vertices with similar interests together?

The majority of research in community detection in graphs is for the setting where each node is constrained to be in only one community (i.e., communities do not overlap). The canonical model here is the *planted partition* or the *stochastic block model* [1], [2], where two nodes in the same community are more likely to share an edge compared to two nodes in different communities. Hence, a community can be thought of as a subgraph with greater edge density than the edge density across the communities. A more practical and difficult problem is the case where the communities overlap, thus enabling nodes to be part of multiple communities.

The theoretical study of models and algorithms for overlapping communities is very limited. In this paper, we consider a natural extension of stochastic block model to the overlapping communities: two nodes that share at least one community are more likely to be connected that two nodes that do not share any communities. We present a new overlapping community detection algorithm and theoretically show that it can successively recover all the communities under this model.

The main ideas behind our algorithm can be explained as follows. Suppose an oracle identifies a subset of nodes from each community, call these as *labeled nodes*. Then the community membership of an unlabeled node can be determined by computing the number of edges that it shares with each of the labeled subset of nodes. If each community contains a sufficient number of such labeled nodes, then we can ensure to correctly identify the community membership of all unlabeled nodes. However, recovering the so-called labeled nodes from each community in the absence of any oracle is a non-trivial task. The key observation is that in many real networks each community has a subset of *pure nodes* which do not belong to any other community. We propose an algorithm that can efficiently identify the subsets of labeled nodes in these graphs through an iterative procedure of degree thresholding and clustering, by exploiting the existence of pure nodes.

This research was supported by ARO grants W911NF-11-1-0265 and W911NF-14-1-0387, and NSF grant CNS-1320175.

Contributions: The main contributions of this paper can be summarized as follows.

- We propose a new overlapping community detection algorithm called $DetectOverlapComm$ to recover overlapping communities in a graph when each community has a set of pure nodes.
- Under a simple random graph generating model for overlapping communities, we evaluate the performance of the $DetectOverlapComm$ algorithm, proving that it can successfully recover all the communities with high probability for both dense graphs ($O(n)$ average degree) and sparse graphs ($O(\log n)$ average degree) with n nodes.
- We further test the performance of our algorithm on both real and synthetic data sets to validate the analytical results.

Related work: Community detection and graph clustering have been well studied in literature. Community detection algorithms have been studied for cases when the communities are disjoint or overlapping. Here, we briefly highlight the most relevant studies that provide analytical results for the corresponding algorithms. An extensive survey of these algorithms can be found in [3], [4], [12]. Several generative models for graphs with communities have been proposed to study the theoretical performance of these algorithms. In the disjoint community case, and with the planted partition or the stochastic block model [1], [2], several spectral [2], [5]–[7] and convex optimization based algorithms [8]–[10] have been shown to provably recover all the underlying communities in the graph. However generative models for overlapping communities have been less studied. In [11] Arora et al. describe an expected degree random graph model where nodes in each community have a fixed affinity to connect to other nodes in the same community. Then they propose randomized algorithms to recover the communities for dense graphs. Anandkumar et al. in [12], [13] study a mixed membership model for overlapping communities [14] where each node can probabilistically identify with multiple

communities. They propose a tensor based algorithm which guarantees to find these mixed community membership of the nodes. For this type of mixed membership model, variational inference based algorithms have been proposed in [16], [17]. In [15] Balcan et al. propose a search and refinement based algorithm for detecting endogenously formed overlapping communities. We finally refer to [12] for a detailed discussion of various approaches.

Basic notations and definitions: We consider a graph $G = (V, E)$ with K overlapping communities. V_k is used to denote the set of nodes in the k-th community. $U_k \subset V_k$ denotes the set of *pure nodes* for each community k, i.e., the set of nodes that belong to only community k. Nodes which belong to multiple communities are referred to as *mixed nodes*. The subgraph of G restricted to the nodes in $S \subset V$ is denoted as $G_S = (S, E_S)$. For any node $i \in V$ and $A \subset V$, $d_A(i)$ denotes the number of edges from i to the set A and $d(i) = d_V(i)$ is the degree of node i.

Organization: The rest of this paper is organized as follows. We describe our algorithm in Section II. The main results are presented in Section III. Our experimental results are discussed in Section IV. Section V contains the proofs of the key results. We finally end the paper with conclusions.

II. Algorithm Description

In this section we give an overview of our main algorithm called $DetectOverlapComm$.

A. Overlapping Community Detection

The task of the overlapping community detection algorithm is to find the true community membership of each node $i \in V$ from the graph G. This is also equivalent to finding the sets $V_1, \ldots, V_K$. The main idea of the $DetectOverlapComm$ algorithm is as follows. The algorithm has three main steps. In the **first step** it calls a subroutine called $FindPureNode$ which detects a set $\widehat{U}_k$ of pure nodes from each community k. Then in the **second step** two edge density parameters $\hat{p}, \hat{q}$ are estimated using these pure node sets. Since the edge density[1] within the nodes of a community is higher than the average density in the graph, the parameter $\hat{p}$ quantifies this edge density so that we can determine which subgraph qualifies as a community. Real graphs can be noisy resulting in edges also between nodes which do not share any community. This noise level edge density is quantified by the second parameter $\hat{q}$. Finally in the **third step** the community membership of the remaining nodes are estimated by thresholding the number of edges they share with each of these pure node sets $\widehat{U}_1, \ldots, \widehat{U}_K$, where the thresholds are determined by $\hat{p}, \hat{q}$ and the size of the pure node sets. The $DetectOverlapComm$ algorithm takes as input the graph G, a parameter γ that characterizes the minimum size of the pure node sets to be recovered, initial edge density estimates p_0, q_0, and outputs the community estimates $\widehat{V}_1, \ldots, \widehat{V}_K$. The pseudo-code is given in Algorithm 1.

[1]The edge density of a set of nodes A is the ratio of number of actual edges shared between nodes in A to the number of all possible edges.

Algorithm 1 $DetectOverlapComm(G, \gamma, p_0, q_0)$

1: $\widehat{U}_1, \ldots, \widehat{U}_K \leftarrow FindPureNode(G, \gamma, p_0, q_0)$
2: $\hat{p} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|\widehat{U}_k|} \sum_{i \in \widehat{U}_k} d_{\widehat{U}_k}(i)$
3: $\hat{q} \leftarrow \frac{1}{\binom{K}{2}} \sum_{k_1, k_2 \in [K]} \frac{1}{|\widehat{U}_{k_1}||\widehat{U}_{k_2}|} \sum_{i \in \widehat{U}_{k_1}, j \in \widehat{U}_{k_2}} (d_{\widehat{U}_{k_2}}(i) + d_{\widehat{U}_{k_1}}(j))$
4: **for** $k = 1$ to K **do**
5: $\quad \widehat{V}_k \leftarrow \widehat{U}_k$
6: $\quad \theta_k \leftarrow |\widehat{U}_k|(\hat{p} + \hat{q})/2$
7: $\quad$ **for** $i \in V \setminus \cup_{k=1}^{K} \widehat{U}_k$ **do**
8: $\quad\quad$ **if** $d_{\widehat{U}_k}(i) \geq \theta_k$ **then**
9: $\quad\quad\quad \widehat{V}_k \leftarrow \widehat{V}_k \cup \{i\}$
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: **end for**
13: Output $\widehat{V}_1, \ldots, \widehat{V}_K$

Note that Algorithm 1 works even if lines 1, 2, 3 are replaced by an oracle which returns non-overlapping sets of nodes from each community and also returns appropriate $\hat{p}, \hat{q}$.

B. Recovering pure node clusters

Before we describe the $FindPureNode$ algorithm to detect pure node sets, we briefly review the convex optimization based method for clustering non-overlapping communities in a graph. This method is used as a subroutine by the $FindPureNode$ algorithm.

Non-overlapping community detection via convex optimization:

Let A denote the adjacency matrix for the graph G, $\Omega(A)$ be its support (i.e., the set of nonzero elements of A), and $\Omega(A)^c$ be the complement of this support. Let $\mathcal{P}_{\Omega(A)}B$ be the projection of the matrix B on the support of A. $||Y||_*$ denotes the nuclear norm (singular values' sum) of matrix Y. Further, the ℓ_1 norm of a matrix M is defined as $||M||_1 = \sum_{i,j} |M(i,j)|$. A convex optimization based algorithm [8], [10] clusters a set of nodes into non-overlapping communities by solving the following optimization problem (CP1)

$$\text{(CP1)} \quad \min_{Y,B} \quad ||Y||_* + c_1||\mathcal{P}_{\Omega(A)}B||_1 + c_2||\mathcal{P}_{\Omega(A)^c}B||_1$$
$$\text{s.t.} \quad Y + B = A$$
$$0 \leq Y_{i,j} \leq 1, \forall (i,j)$$

Here c_1, c_2 are some suitably chosen weights. For a graph, with non-overlapping communities, (CP1) can recover the ideal cluster matrix Y^* where $Y^*_{i,j} = 1$ if nodes i, j belong to the same cluster and $Y^*_{i,j} = 0$ otherwise.

By solving (CP1) we can recover all communities of size greater than $\Omega(\sqrt{n})$ [8]. In fact, even in presence of communities of size less than $\sqrt{n}$, solving (CP1) with appropriate parameters can still recover the larger communities [10]. We will use this particular version of the algorithm for (CP1) called $RecoverBigFullObs$ in [10] as a

subroutine for our $FindPureNodes$ algorithm. We rename $RecoverBigFullObs$ as $ClusterCP$ in our algorithm for the ease of exposition. Note that we can solve (CP1) without the knowledge of the number of non-overlapping communities present in the graph.

Finding pure nodes:

We now describe the algorithm $FindPureNode$ to recover sets of the pure node clusters $\widehat{U}_1, \ldots, \widehat{U}_k$. The algorithm is based on successive degree thresholding and clustering using the $RecoverBigFullObs$ subroutine (called $ClusterCP$ in Algorithm 2). The key idea is that, for any community k, a pure node $i \in V_k$ is likely to have a lower degree than a mixed node $j \in V_k$. This is because a pure node is part of only one community k and shares more edges with only nodes in V_k. On the other hand, a mixed node j is also part of at least one more community l besides k, hence it shares large number of edges with nodes in $V_k \cup V_l$. However note that there may still be pure nodes in larger communities with degree higher than a mixed node in a smaller community. Nevertheless, some of the smallest degree nodes in the entire graph will be pure nodes (those from the smallest communities). By choosing an appropriate threshold degree θ, nodes with degree less than θ in V will only be pure nodes from some of these smallest communities. Now clustering or grouping these pure nodes into communities is equivalent to clustering for non-overlapping communities since a pure node belongs to only one community. This can be achieved using convex optimization based clustering algorithms discussed above (CP1). Once we recover a pure node cluster $\widehat{U}_k \subset V_k$, we can use this set of nodes as a labeled reference set to find all other pure and mixed node in this community since these nodes will share many edges with the pure node set $\widehat{U}_k$. Therefore, we can recover this community V_k.

By removing the nodes in V_k from V, we now have a reduced problem with one less community. Therefore, by *successively applying these three steps of degree thresholding, clustering, and removing communities*, we can recover pure node sets $\widehat{U}_k$ from all communities $k \in [K]$.

Note that we do not recover all the communities $V_1, \ldots, V_K$ but only pure node sets from each of these communities. To see this, suppose in the first iteration we recover pure node set from community 1, then we identify and remove all nodes from V_1. However since many of these nodes are mixed nodes, such nodes also get removed from the remaining communities. In the second iteration, suppose we recover pure node set from community 2, we then identify and remove all remaining nodes from V_2, many of them may have already been removed in the first iteration through V_1. Therefore we cannot identify the entire community V_2. This algorithm has analogs to Successive Interference Cancellation in multiuser communication [18]. Let d_{max} be the maximum degree of a node in G. The $FindPureNode$ algorithm takes as input the graph G, the pure node cluster size parameter γ, initial edge density estimates p_0, q_0, and outputs the pure node sets $\widehat{U}_1, \ldots, \widehat{U}_K$. The pseudo-code is presented in Algorithm 2.

Algorithm 2 $FindPureNode(G, \gamma, p_0, q_0)$

1: $V' \leftarrow V, continue \leftarrow 1$
2: $\mathcal{U} \leftarrow \emptyset$
3: **while** $V' \neq \emptyset$ and $continue = 1$ **do**
4: $\quad \theta \leftarrow 1$
5: $\quad$ Split V' randomly into two sets P and Q each of size $|V'|/2$
6: $\quad$ For each $i \in P$ compute $d_Q(i)$
7: $\quad \gamma_0 \leftarrow 0$
8: $\quad$ **while** $\gamma_0 < \gamma/4$ and $\theta \leq d_{max}$ **do**
9: $\quad\quad \theta \leftarrow \theta + 1$
10: $\quad\quad U' \leftarrow \{i \in P : d_Q(i) \leq \theta\}$
11: $\quad\quad$ Estimate $\hat{p}, \hat{q}$ from $\mathcal{U}, p_0, q_0$
12: $\quad\quad U'_1, \ldots, U'_r \leftarrow ClusterCP(U', A_{U'}, \hat{p}, \hat{q})$
13: $\quad\quad \gamma_0 \leftarrow \max_{j=1,\ldots,r} |U'_j|$
14: $\quad\quad$ **if** $\gamma_0 \geq \gamma/4$ **then**
15: $\quad\quad\quad \mathcal{U} \leftarrow \mathcal{U} \cup_{j:|U'_j| \geq \gamma/4} U'_j$
16: $\quad\quad$ **end if**
17: $\quad$ **end while**
18: $\quad W \leftarrow \{j \in V : \exists U \in \mathcal{U}, d_U(j) \geq |U|(\hat{p} + \hat{q})/2\}$
19: $\quad V' \leftarrow V \backslash W$
20: $\quad$ **if** V' remain unchanged **then**
21: $\quad\quad continue \leftarrow 0$
22: $\quad$ **end if**
23: **end while**
24: Output $\mathcal{U} = \widehat{U}_1, \ldots, \widehat{U}_K$

Recall that the convex optimization based clustering algorithm $ClusterCP$ ($RecoverBigFullObs$ in [10]) requires as input the graph, its adjacency matrix and edge density estimates $\hat{p}, \hat{q}$ but does not require the number of communities. Hence, in line 12 of Algorithm 2, r is the number of non-overlapping clusters found by the $ClusterCP$ algorithm, the sets being $U'_1, \ldots, U'_r$ [2].

The $ClusterCP$ algorithm also requires edge density estimates $\hat{p}, \hat{q}$ as input. This is computed in line 11 of Algorithm 2. This can be done in several ways. The $FindPureNode$ algorithm is provided with initial edge density estimates p_0, q_0 as input. This can be estimated from a labeled training dataset where we know a small subset of nodes from a few communities. Until at least two pure node sets have been recovered this p_0, q_0 can be used as estimates $\hat{p}, \hat{q}$. However once two pure node sets are recovered $\hat{p}, \hat{q}$ can be estimated as in lines 2,3 of Algorithm 1. Another way to find initial estimates of $\hat{p}, \hat{q}$ is by using the first two eigenvalues of the matrix $A_{U'} - I_{U'}$ as shown in [8], where $I_{U'}$ is the identity matrix of size $|U'|$.

III. Main Results

In this section, we present out main results. The proofs are provided in Section V. First, we describe three sufficient con-

[2]Note that r can vary in each iteration depending on number of pure node clusters currently in the set U'.

ditions under which the $DetectOverlapComm$ algorithm is guaranteed to recover all overlapping communities from the graph. We use the following notations: $\alpha_k := |V_k|, \forall k \in [K]$, $\alpha := \min_{k\in[K]} \alpha_k$, and $\gamma := \min_{k\in[K]} |U_k|$.

A. Sufficient Conditions

Random graph generative model. For every node $i \in V$, define its community membership set $C_i = \{k \in [K] : i \in V_k\}$. We know that the nodes within each community are more densely connected than nodes in separate communities. To capture this, we consider a simple random graph generative model as follows.

(A1) The edges in G are generated independently, where the probability of an edge (i, j) is p when $C_i \cap C_j \neq \emptyset$, and is q when $C_i \cap C_j = \emptyset$, for $0 < q < p < 1$.

Therefore any two nodes which share at least one community are connected with a higher probability than nodes which do not share any common community[3]. Note that when the communities are non-overlapping this model reduces to the classical planted partition model [1], [7], [8].

Overlap size. This condition specifies the size of overlap between any pair of communities. Two communities with very large overlaps are inherently difficult to learn since it becomes statistically harder to differentiate between the two. Recall $\alpha = \min_{k\in[K]} \alpha_k$ is the size of the smallest community. Let $\bar{d}_{max} = \max_{k\in[K]} \alpha_k p + (n - \alpha_k)q$ be the maximum expected degree of a node. The condition is as follows.

(A2) For any two communities $k_1, k_2 \in [K]$, there exists $\beta > 0$ such that

$$|V_{k_1} \cap V_{k_2}^c| \geq \beta = \Omega\left(\frac{\bar{d}_{max}}{(p-q)}\sqrt{\frac{\log n}{\min(\alpha p, (n-2\alpha)q)}}\right).$$

Consequently, the maximum number of nodes that community k_1 shares with another community k_2 is upper bounded by $\alpha_{k_1} - \beta$.

Pure nodes. The third condition guarantees that every community has a set of pure nodes which only belong to that community. This assumption is reasonable in large-scale networks with several communities. Examples include a co-authorship network (authors with sole-area interests) [19], and a protein-protein interaction network (proteins with only one functionality) [20].

(A3) For any community $k \in [K]$, there exists a subset of nodes $U_k \subseteq V_k$ which belong to only community k. There exists $\gamma > 0$ such that $\min_{k\in[K]} |U_k| \geq \gamma$, where

$$\gamma = \max\left(\frac{C_1\sqrt{p(1-q)\Gamma}}{(p-q)}\log^2\Gamma, \frac{C_2 p^2}{(p-q)^2 q}\log n\right),$$

where $\Gamma = \sum_{k\in[K]} |U_k|$ and C_1, C_2 are constants.

[3]Our results also extend to the setting where probability of en edge between nodes i, j is an increasing function of the number of communities they share, i.e., $|C_i \cap C_j|$.

B. Results

We will now present our main theorem. First we state the following lemma which guarantees that under conditions (A1) and (A2), with high probability, the degree of any pure nodes will be less than a mixed node in the same community. This is the main reason why the degree thresholding can separate pure nodes from mixed nodes of same community.

Lemma 1 *Consider a graph with K overlapping communities satisfying assumption (A1), (A2). Let P, Q be a random unbiased split of V. Consider a community k, and let $i \in P \cap V_k$ be a pure node and $j \in P \cap V_k$ be a mixed node. Then there exists $\theta \in \mathbb{R}$ such that $d_Q(j) > \theta > d_Q(i)$ with high probability.*

We comment that the random split in line 5 of Algorithm 2 is mainly required for the proof. This makes the out-degree $d_Q(i)$ of nodes $i \in P$ statistically independent of the edges within subgraph G_P. This is a standard technique used in many community detection algorithms [7], [12]. Lemma 1 shows that the degree properties of a pure node i and mixed node j are also reflected in the out-degrees $d_Q(i), d_Q(j)$ after the random split. Theorem 1 shows that with high probability the $FindPureNode$ algorithm correctly recovers subsets of pure nodes from each community. We can then ensure that these sets of pure nodes can be used as reference nodes to find the community membership of all the mixed nodes in $DetectOverlapComm$ algorithm.

Theorem 1 *Consider a graph with K overlapping communities satisfying assumptions (A1), (A2), and (A3). Given $p_0 = p, q_0 = q$, the $FindPureNodes$ algorithm can correctly recover the pure node clusters $\widehat{U}_1, \ldots, \widehat{U}_K$, with high probability, such that $\widehat{U}_k \subseteq U_k$ and $\min_{k\in[K]} |\widehat{U}_k| \geq \gamma/4$.*

Theorem 2 guarantees recovery of all communities under conditions (A1), (A2) and (A3) with high probability. For the Theorem we assume $p_0 = p, q_0 = q$ are given as input. Proposition 2 in section V shows that with K pure nodes sets of size at least γ, we can estimate p, q with high accuracy.

Theorem 2 *Consider graph G with K overlapping communities satisfying assumptions (A1), (A2), and (A3). Then with high probability $DetectOverlapComm$ algorithm correctly recovers the communities $V_1, \ldots, V_K$ with $p_0 = p, q_0 = q$.*

Remark 1 *For a graph G with non-overlapping communities the $DetectOverlapComm$ algorithm has the same performance as the convex optimization based clustering algorithm in [10]. Since there are no mixed nodes, any choice of threshold θ suffices. The algorithm still choose θ depending on relative size of the communities until it recovers at least $\gamma/4$ nodes from one community. However we can take $\gamma = \alpha$ and obtain the guarantees in [10].*

Remark 2 *For dense graphs when $p = \Theta(1), q = \Theta(1), (p - q) = \Theta(1)$, let $\alpha = \Omega\left(n^{2/3}\right)$, $K = O(1)$.*

Then Theorem 2 requires $\beta = \Omega\left(n^{2/3}\right), \gamma = \Omega(\log n)$ *to guarantee successful recovery.*

Remark 3 *For sparse graphs for* $p = \Theta\left(\frac{\log n}{n}\right), q = \Theta\left(\frac{\log n}{n}\right), p - q = \Theta\left(\frac{\log n}{n}\right)$ *Theorem 2 requires* $\alpha = \Theta(n), \beta = \Theta(n), \gamma = \Theta(n)$.

The following proposition characterizes the runtime of the $DetectOverlapComm$ algorithm.

Proposition 1 *Under assumptions (A1), (A2), (A3) in a graph* G *with* K *overlapping communities the* $DetectOverlapComm$ *algorithm has a runtime of* $O\left(\overline{d}_{max} K\Gamma^3 + n\Gamma\right)$ *with high probability.*

C. Performance Comparison

A good theoretical performance comparison of various overlapping community detection algorithms is presented in [12], Section 1.3. In comparison, $DetectOverlapComm$ algorithm has the following performance.

- Compared to the randomized algorithms by Arora et al. in [11] where the results focus on dense graphs, the $DetectOverlapComm$ algorithm can recover communities even in sparse graphs with $O(n \log n)$ edges. It also has a much smaller runtime with $\Theta(n)$ sized communities.
- In dense graphs with $O(\log n)$ pure nodes, the $DetectOverlapComm$ algorithm has a runtime of $O\left(nK \log^3 n\right)$, which is smaller than $O(n^2 K)$ runtime of the tensor based algorithm by Anandkumar et al. [12].
- The overlapping community detection algorithm by Balcan et al. in [15] requires that for any node, the number of its edges within its community is larger than its number of edges out of the community. Our algorithm does not require such restrictions. Also in sparse graphs these algorithms require a quasi-polynomial runtime in n (with $O(\log n)$ average degree) unlike the $DetectOverlapComm$ algorithm whose runtime is at most polynomial.

IV. Numerical Experiments

In this section we evaluate the performance of the $DetectOverlapComm$ (OCD) algorithm on both real and synthetic datasets. We compare the performance of our algorithm with the popular Link Partition (LP) algorithm for overlapping community detection [21]. We use the C++ implementation of the LP algorithm from [22]. Recall that the split in line 5 of Algorithm 2 is mainly required for the analysis. Hence for these experiments we skip this step and take $d_Q(i)$ as the degree of node i in Algorithm 2.

A. Real Dataset

We run both $DetectOverlapComm$ and LP algorithms in a co-authorship network dataset (1589 network science researchers) used in [23]. Since the overall graph is disconnected, we run both algorithms on the largest connected component having 379 nodes. The graph is sparse having just 914 edges. Since the actual ground truth communities are not known, we evaluate the results based on five different widely used performance score functions in community detection [24]. The score functions are Cut Ratio (CR), Conductance (C), Fraction over median degree (FOMD), Triangle Participation Ratio (TPR) and Modularity (MOD) which were shown to be good representative community scoring functions in [24]. We run the $DetectOverlapComm$ algorithm with different values of input parameters. Table I shows the various scores obtained (averaged over all communities) using $\gamma = 5, p_0 = .5, q_0 = .01$ when the algorithm recovers 48 communities with 331 nodes. 48 nodes were not assigned to any communities as these were found to be outliers. For LP algorithm note that we can obtain different number of communities by varying the threshold in the hierarchical edge clustering step. To make a fair comparison we choose a threshold such that the number of recovered communities is closest to one obtained by Algorithm 1 since scores like CR and C are expected to be better (less) as the number of communities decrease. Hence for threshold of .15 the LP outputs 42 communities which is also when the communities have the highest modularity score. The average scores over all 42 communities are shown in Table I. In comparison we see that the communities obtained by $DetectOverlapComm$ algorithm has a better CR, C, FOMD and TPR while that of LP algorithm has a better MOD.

Algorithm	CR	C	FOMD	MOD	TPR
DOC	.0058	.3120	.3894	.6125	.9028
LP	.0072	.3265	.1706	.7415	.7620

TABLE I

TABLE SHOWING THE PERFORMANCE OF $DetectOverlapComm$ (DOC) AND LINK PARTITION (LP) [21] ALGORITHMS ON REAL CO-AUTHORSHIP NETWORK DATASET [23] WITH 379 NODES. THE PERFORMANCE IS COMPARED IN TERMS OF AVERAGE COMMUNITY SCORING METRICS CUT RATIO (CR), CONDUCTANCE (C), FRACTION OVER MEDIAN DEGREE (FOMD), TRIANGLE PARTICIPATION RATIO (TPR) AND MODULARITY (MOD) FOR THE CASE WHEN DOC AND LP OUTPUTS 48 AND 42 COMMUNITIES RESPECTIVELY. COMMUNITIES OBTAINED BY DOC HAVE A BETTER C, CR, TPR AND FOMD WHILE LP COMMUNITIES HAVE A BETTER MOD.

B. Synthetic Dataset

We further test the performance of $DetectOverlapComm$ (DOC) and Link Partition (LP) algorithm [21] on synthetic graphs generated according to the model in section III-A. We generated graphs with $n = 1000, K = 5$ and $50 - 70$ pure nodes per community. The size of the communities vary between $430 - 470$. As we showed in section III, the performance of DOC (Algorithm 1) is governed by the difference $p - q$. Figure 1 shows the average error performance of both algorithms as a function of the difference $p - q$ for two different values of p. For DOC, we further consider two cases which are depicted by blue curves and red curves in Figure1: The blue curve shows the performance when the probabilities

p and q are known and hence we simple choose $p_0 = p$ and $q_0 = q$. The red curve shows the performance when p and q are not known and hence we estimate p_0 and q_0 based on the graph using the approach discussed in section II. As it can be observed, the error performance in the latter case (i.e., when the algorithm use the estimated values) is only marginally poorer than the performance when the exact p, q are known, showing that our algorithm is quite robust against such estimation errors. For the LP algorithm, we choose different values of the threshold in the hierarchical clustering step and we plot the minimum error obtained corresponding to the best threshold. The $DetectOverlapComm$ (DOC) algorithm demonstrates a significantly better error performance than the Link Partition (LP) algorithm.

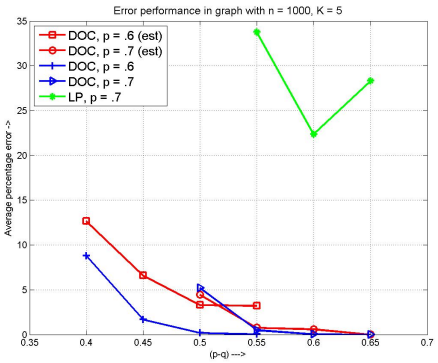

Fig. 1. Figure showing the error performance of $DetectOverlapComm$ (DOC) and Link Partition (LP) algorithm on synthetic graphs with $n = 1000$ and $K = 5$ and different values of $p, p - q$. The blue curve corresponds to the case when $p_0 = p, q_0 = q$ and the red curve corresponds to the case when p_0, q_0 are estimated from the graph. In both case the DOC algorithm shows a much better average error than LP algorithm.

V. Proofs

Proposition 2 *Under assumption (A1) with $K \geq 2$, with the pure node subsets $U_1, \ldots, U_K$ such that $|U_k| \geq \gamma$, the estimation error in $\hat{p}, \hat{q}$ (in lines 2, 3 of Algorithm 1) can be bounded as follows*

$$\begin{aligned} |\hat{p} - p| &\leq \sqrt{\frac{6p \log n}{K\gamma(\gamma - 1)}}, \\ |\hat{q} - q| &\leq \frac{1}{\gamma}\sqrt{\frac{6q \log n}{K(K-1)}}, \end{aligned}$$

with high probability.

Proof: Consider the estimation of $\hat{p}$. Under assumption (A1), the expected number of edges among the pure nodes in the same community (i.e., the intra-cluster edges among the pure nodes) is simply given by

$$S = \sum_{k=1}^{K} \binom{|U_k|}{2} p.$$

Then by the Chernoff bound, with probability at least $1 - e^{-\epsilon^2 Sp/3}$, we have $|\hat{p} - p| \leq \epsilon p$. Now $S \geq K\binom{\gamma}{2}p$ since $|U_k| \geq \gamma$. Hence by taking

$$\epsilon = \sqrt{\frac{6 \log n}{K\gamma(\gamma - 1)p}},$$

with probability greater than $1 - 1/n$, we get the required bound. For $\hat{q}$, the expected number of inter-cluster edges among the pure nodes is at least $\binom{K}{2}\gamma^2$. Again by applying the Chernoff bound, we get the required bound on the estimation error.

■

Proof of Lemma 1

Proof: Let $A = V_k \cap Q$. Now sets P and Q are formed by a equally likely random split of V. Hence using Chernoff bound it is easy to show that with high probability

$$|A| \geq |V_k|/4 = \frac{\alpha_k}{4} \geq \frac{\alpha}{4} \tag{1}$$

From (A1) the degree of any node is distributed as sum of two binomial random variables. Therefore for a pure node $i \in P \cap V_k$ using Chernoff bound with probability at least $1 - e^{-\epsilon^2\mu_1/3} - e^{-\epsilon^2\mu_2/3}$,

$$d_Q(i) \leq (1 + \epsilon)(\mu_1 + \mu_2), \tag{2}$$

where $\mu_1 = |A|p, \mu_2 = (|Q| - |A|)q$. Now for a mixed node $j \in P \cap V_k$ there exits at least another community V_l it is part of. Hence node j shares an edge with probability p with any other node in $V_k \cup V_l$. From assumption (A2) we know that $|V_k^c \cap V_l| \geq \beta$. Using Chernoff bound with high probability $|V_k^c \cap V_l \cap Q| \geq \frac{\beta}{4}$. Again using Chernoff bound the out degree of node j can be lower bounded as

$$d_Q(j) \geq (1 - \epsilon)(\mu_3 + \mu_4) \tag{3}$$

with probability at least $1 - e^{-\epsilon^2\mu_3/3} - e^{-\epsilon^2\mu_4/3}$, where $\mu_3 = (|A| + \beta/4)p, \mu_4 = (|Q| - |A| - \beta/4)q$.

Let

$$\mu = \min\{\mu_1, \mu_2, \mu_3, \mu_4\} = \min\{\mu_1, \mu_4\},$$

and

$$\epsilon = \sqrt{\frac{3}{\mu} \log \frac{4n}{\delta}},$$

then by the union bound we have for any pure node $i \in P \cap V_k$ and mixed node $j \in P \cap V_k$ for any k with probability at least $1 - \delta$,

$$\begin{aligned} d_Q(j) &\geq (1 - \epsilon)(\mu_3 + \mu_4) \\ &= (|A| + \beta/4)(1 - \epsilon)p \\ &\quad + (|Q| - |A| - \beta/4)(1 - \epsilon)q \\ &= (1 - \epsilon)(\mu_1 + \mu_2) + \frac{\beta}{4}(1 - \epsilon)(p - q). \end{aligned}$$

Now from assumption (A2), let

$$\beta = \frac{16\epsilon(\mu_1 + \mu_2)}{(1 - \epsilon)(p - q)} = \Omega\left(\frac{\bar{d}_{max}}{(p - q)\sqrt{\mu}}\sqrt{\log \frac{n}{\delta}}\right).$$

Note that μ scales as $\mu = \Omega\left(\min(\alpha p, (n-2\alpha)q)\right)$ since $|Q| = \Theta(n), |A| = \Omega(\alpha)$ and $\beta \leq \alpha$. Hence, it follows that,

$$\begin{aligned} d_Q(j) &\geq (1-\epsilon)(\mu_1+\mu_2) + \frac{\beta}{4}(1-\epsilon)(p-q) \\ &= (1-\epsilon)(\mu_1+\mu_2) \\ &\quad + \frac{4\epsilon(\mu_1+\mu_2)}{(1-\epsilon)(p-q)}(1-\epsilon)(p-q) \\ &> (1-\epsilon)(\mu_1+\mu_2) + 2\epsilon(\mu_1+\mu_2) \\ &= (1+\epsilon)(\mu_1+\mu_2) \geq d_Q(i). \end{aligned}$$

Therefore,

$$d_Q(j) > \theta > d_Q(i), \tag{4}$$

for

$$\theta = ((1+\epsilon)(\mu_1+\mu_2) + (1-\epsilon)(\mu_3+\mu_4))/2.$$

■

We will use the following lemma in the proof of Theorems 1 and 2.

Lemma 2 *Consider any pair of nodes $i \in V_k$, $j \notin V_k$, and a set $A \subset V_k \backslash i$, for any $k \in [K]$. Under assumption (A1) with probability greater than $1-\delta$, $\delta \in (0,1)$, we have $d_A(i) > |A|(p+q)/2 > d_A(j)$ whenever*

$$|A| \geq \frac{48p^2}{(p-q)^2 q} \log \frac{2n}{\delta}.$$

Proof: Since $i \in V_k$ and $j \notin V_k$ any node $l \in A$ shares an edge with i with probability p and shares an edge with j with probability q. Using Chernoff bound for $\epsilon = \frac{p-q}{4p}$ with probability at least $1 - e^{-\epsilon^2|A|p/3}$

$$\begin{aligned} d_A(i) \geq (1-\epsilon)|A|p &> \left(1 - \frac{(p-q)}{2p}\right)|A|p \\ &= (p+q)|A|/2. \end{aligned} \tag{5}$$

Also with probability greater than $1 - e^{-\epsilon^2|A|q/3}$,

$$\begin{aligned} d_A(j) \leq (1+\epsilon)|A|q &< \left(1 + \frac{p-q}{2q}\right)|A|q \\ &= (p+q)|A|/2 \end{aligned} \tag{6}$$

Therefore whenever $|A| \geq \frac{48p^2}{(p-q)^2 q} \log \frac{2n}{\delta}$, taking union bound over all nodes, with probability greater than $1-\delta$, combining equations (5), (6) we have for any $i \in V_k, j \notin V_k$

$$d_A(i) > (p+q)|A|/2 > d_A(j).$$

■

Proof of Theorem 1

Proof: The proof is by induction. From Lemma 1, for any community V_k there exists $\theta(k)$ such that for any pure node $i \in P \cap V_k$ and mixed node $j \in P \cap V_k$,

$$d_Q(i) < \theta(k) < d_Q(j),$$

with high probability. Without loss of generality, assume $\theta(k) \in \mathbb{Z}$. Now let $\theta_0 = \min_{k\in[K]} \theta(k)$. Note that for any $\theta \leq \theta_0$, the set U' does not contain any mixed node since for any mixed node $j \in V_k$, $d_Q(j) > \theta(k) > \theta_0$. Since all the nodes in U' are pure nodes, clustering the nodes in U' is equivalent to clustering for non-overlapping communities. If for some $\theta < \theta_0$ the set U' contain at least $\gamma/4$ pure nodes from a community and $ClusterCP$ / $RecoverBigFullObs$ algorithm successfully recovers this set we are done. If not, then for $\theta = \theta_0$ we know that there exists a community $l = \arg\min_{k\in[K]} \theta(k)$ for which any pure node $i \in P \cap V_l$ has $d_Q(i) < \theta_l = \theta_0$, therefore $i \in U'$. From assumption (A3) since the set of pure nodes $U_l \subset V_l$ has size at least γ, then using Chernoff bound with high probability $|P \cap U_l| \geq \gamma/4$. Thus U' contains at least one subset $U_l' = P \cap U_l$ of pure nodes from a single community of size at least $\gamma/4$. Note that since the edges within the nodes in U' were not used to determine the out degrees $d_Q(i)$, the distribution of edges within U' follows exactly the same distribution as in the classical planted partition model or stochastic block model [2], [8], [10], i.e., between any two pure nodes from the same cluster there is an edge with probability at least p and between pure nodes in different clusters there is an edge with probability at most q. The number of clusters K' in U' satisfies $1 \leq K' \leq K$, but may not have nodes from all K clusters since pure nodes from bigger communities may have larger degree with $d_Q(i) > \theta_0$. Recall that Γ is the total number of pure nodes. Under assumptions (A3) with initial threshold

$$\ell_\sharp = \gamma/4 \geq \frac{b_3 \kappa \sqrt{p(1-q)\Gamma}}{(p-q)} \log^2 \Gamma,$$

Theorem 3 in [10] guarantees that $ClusterCP$ / $RecoverBigFullObs$ algorithm correctly recovers all clusters of size greater than $\ell_\sharp$ with high probability. Hence we can successfully recover cluster U_l'. Now from Lemma 2 and assumption (A3) for any node $i \in V_l$,

$$d_{U_l'}(i) > (p+q)|U_l'|/2.$$

Therefore $V_l \subseteq W$ and community l is removed from V' for the next iteration. Similarly any other community k for which $ClusterCP$ / $RecoverBigFullObs$ algorithm has recovered a large enough pure node cluster U_k' are also removed from V' in the next iteration. Therefore for the next iteration we have a set of nodes V' with at most $K-1$ overlapping communities. Using the same arguments we can show that the number of communities in V' at the $(t+1)$-th iteration is strictly less than the number of communities in the previous t-th iteration. Since the total number of communities K is finite, therefore in at most K iterations the algorithm stops when $V' = \emptyset$. This is when the set $\mathcal{U}$ has pure node clusters $\widehat{U}_k \subseteq U_k$ from each community k with $|\widehat{U}_k| \geq \gamma/4$. Note that when there are outlier nodes which do not belong to any community, at the end of the last iteration V' remain unchanged since $ClusterCP$ algorithm cannot find any more pure node clusters of size at least $\gamma/4$. Hence variable $continue$ is set to 0 and the algorithm terminates.

■

Proof of Theorem 2

Proof: The proof follows from Theorem 1 and Lemma 2. Under assumptions (A1), (A2), and (A3), Theorem 1 guarantees that with high probability we can recover a set of pure nodes $\widehat{U}_k \subseteq U_k$ from each community $k \in [K]$, and for any k, $|\widehat{U}_k| \geq \gamma/4$. Now for any node $j \in V \setminus \cup_{k=1}^{K} \widehat{U}_k$, if $l \in C_j$ then from Lemma 2, for

$$\gamma = \Omega\left(\frac{p^2}{(p-q)^2 q}\log n\right),$$

with high probability

$$d_{\widehat{U}_l}(j) > (p+q)|\widehat{U}_l|/2.$$

Therefore node j will be assigned to set $\widehat{V}_l$. Also if $l \notin C_j$ then $d_{\widehat{U}_l}(j) < (p+q)|\widehat{U}_l|/2$ hence node l is not included in the set $\widehat{V}_l$. Therefore with high probability we correctly recover the set of communities $V_1, \ldots, V_K$. ∎

Proof of Proposition 1

Proof: In each iteration the $ClusterCP$ / $RecoverBigFullObs$ subroutine has a runtime of $O(K\Gamma^3)$. Now the iteration runs till pure node subsets from all communities have been recovered. The pure nodes from the largest community will be the ones to be recovered last. This will happen when the threshold $\theta = O(\bar{d}_{max})$. Therefore the $FindPureNode$ algorithm will have a runtime of $O\left(\bar{d}_{max}K\Gamma^3\right)$ with high probability. Computing out degree to the recovered pure node sets take $O(\Gamma)$ time. Hence assigning communities to all the remaining nodes requires a time of $O(n\Gamma)$. Therefore the total runtime required is $O\left(\bar{d}_{max}K\Gamma^3 + n\Gamma\right)$ with high probability. ∎

VI. Conclusion

In this paper we presented a new algorithm for detecting overlapping communities in a graph. Our algorithm recovers a set of pure nodes from each community using successive steps of degree thresholding, convex optimization based clustering, and node removal. It then uses these sets of pure nodes to find the community memberships of remaining nodes in the graph. Our theoretical results show that the algorithm can correctly recover communities in both dense and sparse graphs. Further the algorithm performs well in our experiments, on both real co-authorship network dataset and synthetic datasets.

While our algorithm in this paper requires the presence of pure nodes in each community, our two-step technique itself is more general and is applicable to the more general setting when the pure nodes do not necessarily exist. In this setting, we can first try to detect a subset of nodes from each community that does not completely overlap with the rest of such subsets from the other communities. Then these nodes can be used as reference sets to find the community membership of remaining nodes. As a future work, we will develop algorithms that can recover such reference set of nodes even in the absence of pure nodes. We will also develop a faster implementation of our algorithm and test its performance on larger datasets.

References

[1] R. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 280–285. IEEE, 1987.

[2] K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 2011.

[3] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[4] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys (CSUR)*, 45(4):43, 2013.

[5] F. McSherry. Spectral partitioning of random graphs. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 529–537. IEEE, 2001.

[6] J. Giesen and D. Mitsche. Reconstructing many partitions using spectral techniques. In *Fundamentals of Computation Theory*, pages 433–444. Springer, 2005.

[7] K. Chaudhuri, F. Chung, and A. Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. *Journal of Machine Learning Research*, 2012:1–23, 2012.

[8] Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[9] S. Oymak and B. Hassibi. Finding dense clusters via" low rank+ sparse" decomposition. *arXiv preprint arXiv:1104.5186*, 2011.

[10] N. Ailon, Y. Chen, and H. Xu. Breaking the small cluster barrier of graph clustering. *arXiv preprint arXiv:1302.4549*, 2013.

[11] S. Aurora, R. Ge, S. Sachdeva, and G. Schoenebeck. Finding overlapping communities in social network: Toward a rigorous approach. In *Proc. of the ACM Conf. on EC*, 2012.

[12] 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.

[13] F. Huang, UN. Niranjan, M. Hakeem, and A. Anandkumar. Fast detection of overlapping communities via online tensor methods, 2013. http://arxiv.org/abs/1309.0787.

[14] Edoardo M Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

[15] M. F. Balcan, C. Borgs, M. Braverman, J. Chayes, and S. H. Teng. Finding endogenously formed communities. In *Proc. of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 767–783. SIAM, 2013.

[16] P. K. Gopalan and D. Blei. Efficient discovery of overlapping communities in massive networks. *Proc. of the National Academy of Sciences*, 110(36):14534–14539, 2013.

[17] P. Gopalan, C. Wang, and D. Blei. Modeling overlapping communities with node popularities. In *Advances in Neural Information Processing Systems*, pages 2850–2858, 2013.

[18] S. Verdu. *Multiuser detection*. Cambridge university press, 1998.

[19] M. Newman. Coauthorship networks and patterns of scientific collaboration. *Proc. Natl. Acad. Sci. USA*, 101:5200–5205, 2004.

[20] K. Voevodski, S. Teng, and Y. Xia. Finding local communities in protein networks. *BMC bioinformatics*, 10(1):297, 2009.

[21] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[22] J. P. Bagrow. Link partition code, 2010. `http://barabasilab.neu.edu/projects/linkcommunities/`.

[23] M. E. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[24] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *Proc. of the ACM SIGKDD Workshop on Mining Data Semantics*, page 3. ACM, 2012.