Semi-Supervised Cluster Extraction via a Compressive Sensing Approach*

Ming-Jun Lai † and Daniel Mckenzie †‡

Abstract. We use techniques from compressive sensing to design a local clustering algorithm by treating the cluster indicator vector as a sparse solution to a linear system whose coefficient matrix is the graph Laplacian. If the graph is drawn from the Stochastic Block Model we are able to prove that the fraction of misclassified vertices goes to zero as the size of the graph increases. Numerical experiments on simulated and real-life graphs demonstrate the effectiveness and speed of our approach. Finally, we explore the application of our algorithm to semi-supervised learning.

Key words. Stochastic Block model, Compressive Sensing, Sparse Solution, Local Clustering, Community Detection, Semi-Supervised.

AMS subject classifications. 68Q25, 68R10, 68U05, 94A12

1. Introduction.

Finding clusters is a problem of primary interest when analyzing networks. This is because vertices which are in the same cluster can reasonably be assumed to have some latent similarity. Thus, clustering techniques can be used to find communities in social networks [25, 45] functionally similar molecules in protein-protein interaction networks [33], or deduce political affiliation from a network of blogs connected by hyperlinks [3]. Moreover, even data sets which are not presented as graphs can profitably be studied by first creating an auxiliary graph (such as a k-nearest-neighbors graph) and then applying graph clustering techniques. This has been successfully applied to image segmentation [42], natural language processing [19] and differentiating types of breast cancer [21].

We shall informally think of a cluster as a subset of vertices, $C \subset V$ with many edges between vertices in C, and few edges to the rest of the graph, C^c. For a toy example, consider the college football network of Girvan and Newman [25], represented in Figure 1. The vertices of this network correspond to the 115 colleges fielding (American) football teams that played in NCAA Division 1A in Fall 2000. Two vertices are connected by an edge if they played against one another during the regular season. As can be seen from either the graph or the adjacency matrix, this graph contains clusters. In this case, the underlying similarity responsible for the clusters are the conferences to which the teams belong. Despite the simplicity of this graph, it exhibits two subtle clustering related phenomena. The first is the presence of background vertices, illustrated in black. These correspond to the five independent schools - Central Florida, Connecticut, Navy, Notre Dame and Utah State. These schools do not belong to any conference, and thus should not be placed into any cluster. The second is the presence of clusters at multiple scales. For example, the cluster corresponding to the

‡danmac29@uga.edu

*Submitted to the editors 08/12/2018

Funding: The first author was partially supported by the National Science Foundation under the grant #DMS 1521537. The second author acknowledges the financial assistance of the National Research Foundation of South Africa (NRF).

†University of Georgia, Athens, GA

South Eastern Conference (shown in red) could be further divided into two equally sized sub-clusters, both of which form cliques. In the context of this problem, this would reveal further valuable information, as the two sub-clusters correspond to the East and West Divisions of this Conference. Hence it is of practical importance to have clustering algorithms which can be set to find clusters of different sizes, and which are not forced to assign background vertices to a cluster.

Figure 1: Two representations of the college football network of [25]

Of course many real-world graphs of interest today are significantly larger than the college football network. For truly massive graphs it can be computationally intractable to partition the entire vertex set into clusters. Moreover, if one is only interested in the cluster containing several vertices of interest, this is unnecessary. Thus, in the last decade or so, there has been intensive research into local clustering algorithms (see, for example, [43, 27, 31, 38]) loosely defined to be algorithms with complexity proportional to the size of the cluster, not the whole graph.

In this paper we introduce a two-step local clustering algorithm, drawing on ideas from the signal processing field of compressive sensing. Our algorithm, which we call Semi-Supervised Cluster Pursuit (`SSCP`), is computationally efficient, provably accurate, able to find clusters at multiple scales and is not confounded by the presence of background vertices. We prove that for graphs drawn from the Stochastic Block Model (SBM) our algorithm misclassifies at most $o(n_0)$ vertices, where n_0 is the size of the cluster of interest. We further show that, under certain assumptions on the parameters of the SBM, `SSCP` runs in $O(\log^3(n)n)$ operations. Finally we verify, via extensive experimentation on real and artificial graphs, that the performance of our algorithm is comparable, and some cases exceeds, that of many state-of-the-art algorithms. In the interest of reproducibility, we make all our code available at: danielmckenzie.github.io.

The rest of this paper is laid out as follows. In the remainder of §1, we introduce some notation and review the existing literature. In §2 we introduce the `SSCP` algorithm and include a brief overview of the theory of Compressive Sensing. Most of the technical work of this paper

is in §3, where we prove the weak consistency of SSCP. We relegate several particularly technical results to an appendix. Finally in §4 we provide extensive numerical experiments.

1.1. Notation and Definitions. We restrict our attention to finite, simple, undirected graphs $G = (V, E)$, possibly with edge weights. We identify the vertex set V with the integers $[n] := \{1, \dots, n\}$ and denote an edge between vertices i and j as $\{i, j\} \in E$. The (possibly weighted) adjacency matrix of G will be denoted as A. By d_i we mean the degree of the i-th vertex, computed as $d_i = \sum_j A_{ij}$. For quantities such as d_i (and later λ_i and r_i) that are indexed by $i \in [n]$, let $d_{\max} := \max_i d_i$ and similarly $d_{\min} := \min_i d_i$. Denote by D the diagonal matrix whose (i, i) entry is d_i.

Definition 1.1 (Laplacians of graphs). *The* normalized, random walk *Laplacian is defined as* $L = I - D^{-1}A$. *We shall simply refer to it as* the Laplacian. *The* signless *Laplacian is defined as* $L^+ = I + D^{-1}A$ *while the* normalized, symmetric *Laplacian is:* $L^{sym} := I - D^{-1/2}AD^{-1/2}$.

For any $S \subset V$, we denote by G_S the induced sub-graph with vertices S and edges all $\{i, j\} \in E$ with $i, j \in S$. For any $S \subset [n]$ we define an *indicator vector* $\mathbf{1}_S \in \mathbb{R}^n$ by $(\mathbf{1}_S)_i = 1$ if $i \in S$ and $(\mathbf{1}_S)_i = 0$ otherwise. $|S|$ will always denote the cardinality of S. For any matrix B, by B_S we mean the submatrix of B consisting of the columns b_i for all $i \in S$. Suppose for every n we have a probabilistic model $\mathcal{G}^{(n)}$ of graphs on n vertices containing a cluster $C^{(n)}$, for example the stochastic block model introduced in the next section. Let $\mathcal{A}$ be any algorithm for graph clustering problem with output $C^{\#}$. We say that $\mathcal{A}$ is *weakly consistent* if

$$\mathbb{P}\left[\left|C^{\#}\Delta C^{(n)}\right| / \left|C^{(n)}\right| \leq o(1)\right] = 1 - o(1),$$

where for any two sets A and B, $A\Delta B := (A \setminus (A \cap B)) \cup (B \setminus (A \cap B))$ denotes their symmetric difference. Note this is analogous to the *almost exact recovery condition* for partitioned clustering given in [1]. See [27] for a slightly different formulation of this problem.

1.2. Random Graphs. In order to study how well our algorithm performs, it is useful to have a statistical model of graph with latent clusters. The model we shall use in this paper is the Stochastic Block Model (SBM). As pointed out elsewhere (for example in [1]), the SBM strikes a good balance between theoretical tractability and realistically modelling real-world networks.

Definition 1.2. *Let* $\mathbf{n} = (n_1, \dots, n_k)$ *be a vector of positive integers, and let* P *be a* $k \times k$ *symmetric matrix with* $P_{ab} \in [0, 1]$ *for all* a, b. *We say a graph* $G = (V, E)$ *is drawn from* $SBM(\mathbf{n}, P)$ *(and shall write* $G \sim SBM(\mathbf{n}, P)$*) if there exists a latent partition* $V = C_1 \cup C_2 \dots \cup C_k$ *with* $|C_i| = n_i$ *such that any vertices* $i \in C_a$ *and* $j \in C_b$ *are connected by an edge with probability* P_{ab}, *and all edges are inserted independently.*

In [1] and elsewhere, a slightly more general definition is given where it is only required that the expected value of $|C_a|$ is n_a, but the above shall suffice for our purposes. In the special case where all the n_a are equal, $P_{aa} = p$ for all a and $P_{ab} = q$ for all $a \neq b$ we say that G is drawn from the *Symmetric Stochastic Block Model*, and write $G \sim \mathrm{SSBM}(n, k, p, q)$. In this case the clusters are all of size $n_0 := n/k$. We will also use a simpler model of random graph, the Erdős - Rènyi (ER) graph.

Definition 1.3. *We say $G = (V, E)$ is drawn from $ER(n, p)$ (written $G \sim ER(n, p)$) if $\mathbb{P}[\{i, j\} \in E] = p$ for $i, j \in [n]$.*

Note that if $G \sim \text{SSBM}(n, k, p, q)$ then for all $a \in [k]$ $G_{C_a} \sim \text{ER}(n, p)$. We shall use this simple observation repeatedly.

Remark 1.4. Certainly, the Stochastic Block Model is not the only model of random graph studied with regards to clustering. In [32], Lancichinetti, Fortunato and Radicchi proposed a set of models designed to display certain phenomena — such as overlapping communities and a wide range of degrees — that are observed in real-world networks. In [5], random graphs are generated using a *preferential attachment* rule, generating a power-law degree distribution, which is often empirically observed in real-world networks. It would be an interesting topic for future research to investigate how our algorithm applies to such models.

1.3. Some Existing Related Work. Local community detection algorithms (also known as Cluster Extraction algorithms in the statistics literature) seek to find a "good" cluster $C^{\#}$ given a set of seed vertices Γ. In the computer science literature it is usually required that $\Gamma \subset C^{\#}$ (this is the case for `HKGrow` and `Losp++`) while in the statistics literature this is not always the case (see `ESSC`). If desired, this procedure can be iterated a (possibly predefined, possibly data-determined) number of times, finding clusters $C_1, \ldots, C_k$ while not requiring that they cover the vertex set. Depending on the algorithm, the C_a may be allowed to overlap. The set of vertices not assigned a cluster is referred to as the *background vertices*. That is, $V^{\text{background}} := V \setminus \bigcup_{a=1}^{k} C_a$. We review several such algorithms here.

The Extraction of Statistically Significant Communities (`ESSC`) algorithm. The key insight behind this approach is to view communities as fixed points of the update rule:

$$S(B) := \{u \in V : \ u \text{ is strongly connected to } B\} \quad \text{where } B \subset V \tag{1.1}$$

In [46] the idea of a vertex being strongly connected to a set is formalized as a procedure analogous to a statistical p-test. Precisely, denote by G^0 the graph under consideration, and let $d^0(u : B)$ denote the number of edges between a vertex u and a set of vertices B. Assume a null-model for graphs, $\mathcal{G}$ on the same vertex set and with the same degree sequence, but without any *a priori* cluster structure. Let $\hat{d}(u : B)$ be a random variable denoting the number of edges between u and B for graphs drawn from $\mathcal{G}$. If the probability of $\hat{d}(u : B)$ being larger than the value $d^0(u : B)$ is smaller than some threshold value α (usually taken to be 0.05) then say that u is strongly connected to B. Thus (1.1) can be written as:

$$S(B) = \left\{u \in V : \ \mathbb{P}\left[\hat{d}(u : B) \geq d^0(u : B)\right] \leq \alpha\right\}. \tag{1.2}$$

The authors in [46] show that, if $\mathcal{G}$ is taken to be the configuration model, then $d^0(u : B)$ is approximately a binomial random variable, hence the probability in the update rule can be easily computed. The algorithm is initialized with a set of seed vertices B^0 consisting of the highest degree vertex and its neighbors. The update rule (1.2) is then used: $B^{n+1} = S(B^n)$, until $B^{n+1} = B^n$ or a maximum number of iterations is reached. This resulting cluster is then removed and the process may be repeated, terminating when the empty set is returned as a fixed point of the update rule (1.2). No theoretical guarantee of success is given in [46], but experimental results suggest that the algorithm works well.

The `HKGrow` *algorithm.* This algorithm, introduced in [31], is part of a family of cluster extraction algorithms known as diffusion methods. `HKGrow` is based on the idea that if one unit of heat is initially distributed over a small set of seed vertices, and then allowed to spread over the graph via the heat equation, it will concentrate in the cluster containing the seed vertices. More formally, for any seed set $S \subset V$, let $\mathbf{s} = \frac{1}{|S|}\mathbf{1}_S$ and define $\mathbf{h} = \exp(-tL)\,\mathbf{s} := \left(\sum_{k=0}^{\infty}(-t)^k L^k/k!\right)\mathbf{s}$, for an appropriate value of t to be specified by the user. Normalize $\mathbf{h}$ by degree: $\mathbf{v} = D^{-1}\mathbf{h}$, and let $j_1, \ldots, j_n$ be a permutation of $[n]$ such that $v_{j_1} \geq v_{j_2} \geq \ldots, v_{j_n}$. `HKGrow` returns the cluster defined as $C^{\#} = \{j_1, \ldots, j_{k^*}\}$ where

$$k^* = \arg\max\{\mathrm{Cond}(\{j_1, \ldots, j_k\}) \text{ for } k = 1, \ldots, n\} \tag{1.3}$$

For any subset of vertices $U \subset V$, $\mathrm{Cond}(U)$ denotes its *conductance*, defined as follows. Let $\delta U := \{\{i,j\} \in E : \ i \in U \text{ and } j \notin U\}$ denote the boundary of U and let $\mathrm{Vol}(U) = \sum_{i \in U} d_i$ denote its volume, then $\mathrm{Cond}(U) = |\delta U|/\mathrm{Vol}(U)$. From work of Chung [15] it is known that if S is contained in a set of low conductance then $C^{\#}$ will be of similarly low conductance. Experimental results provided in [31] verify this, and show that the performance of `HKGrow` is on par with the Pagerank diffusion method of [4].

The `LOSP++` *algorithm.* This algorithm is a representative of the family of Local Spectral Methods (see also `LEMON` [34] and `LOSP` [26]). `LOSP++`, introduced in [27], works as follows. Given a set of seed vertices S, first extract a subgraph $\widetilde{G}$ from G which is very likely to contain the community C which contains S. Let $\widetilde{A}$ denote the adjacency matrix of $\widetilde{G}$ and denote by N the random walk transition matrix $N = D^{-1}\widetilde{A}$. Define $\mathbf{p}_0 = \mathbf{s} = \frac{1}{|S|}\mathbf{1}_S$ and let $\mathbf{p}_i = N^i\mathbf{p}_0$ denote the distribution of the i-th step of the random walk with initial distribution $\mathbf{p}_0$. For small values of d and k, to be fixed by the user, construct the matrix $V_d^{(k)} = [\mathbf{p}_k, \ldots, \mathbf{p}_{k+d-1}]$. Now let $\mathbf{y}^{\#}$ denote the solution to the linear programming problem:

$$\text{argmin } \|\mathbf{y}\|_1 \text{ such that: } \mathbf{y} \in \mathrm{range}(V_d^k),\ \mathbf{y} \geq 0,\ y_i \geq 1/|S| \text{ for all } i \in S.$$

For a user specified size parameter $\hat{n_0}$, define $C^{\#}$ to be the set of indices of the n_0 largest entries in $\mathbf{y}^{\#}$. In [27] both theoretical and experimental arguments that $C^{\#}$ will be a low conductance cluster containing S are given.

There are certainly other algorithms that fall under the local community detection/ cluster extraction umbrella, such as `Nibble` [43], algorithms which seek to optimize a local modularity score [48] and locally-biased spectral methods [38].

1.3.1. Fundamental Bounds for Recovery.

Recent work of Abbe, Sandon and others has culminated in a theoretical bound beyond which it is impossible to detect cluster membership in the SBM with accuracy better than that of a random guess:

Theorem 1.5 (See [2]). *Exact recovery in the $SSBM(n, k, a\log(n)/n, b\log(n)/n)$ is solvable if $\frac{1}{k}\left(\sqrt{a} - \sqrt{b}\right)^2 > 1$ and not solvable if $\frac{1}{k}\left(\sqrt{a} - \sqrt{b}\right)^2 < 1$. Moreover, when exact recovery is possible, there exist efficient algorithms to do so.*

There exist analogous statements for graphs drawn from the non-symmetric block model. This motivates us to consider values of p and q of the form $c\log(n)/n$ in our theoretical

analysis of SSCP (see §3) although our current analysis requires an additional factor in p, $p = a\omega\log(n)/n$ where ω is any function of n such that $\omega \to \infty$. In our numerical experiments, we take $\omega = \log(n)$. Removing this extra factor is an interesting problem for future research.

2. The SSCP Algorithm.

Our algorithm was inspired by a serendipitous observation that the problem of determining the indicator vector, $\mathbf{1}_C$, of a cluster C can be rephrased as a compressive sensing problem. Before elaborating on this, let us briefly review some of the pertinent results of this field of signal processing.

2.1. Compressive Sensing.

Candés, Donoho and their collaborators in [20, 11, 12] initialized the study of compressive sensing, which offers theoretical analysis and algorithmic tools for solving the minimization problem:

$$\text{argmin}\|\Phi\mathbf{x} - \mathbf{y}\|_2 \text{ subject to } \|\mathbf{x}\|_0 \leq s \tag{2.1}$$

In the case where $\Phi \in \mathbb{R}^{m\times n}$ with $m << n$, making the linear system $\Phi\mathbf{x} = \mathbf{y}$ underdetermined. For any $\mathbf{v} \in \mathbb{R}^n$, define $\|\mathbf{v}\|_0 := |\text{supp}(\mathbf{v})| = |\{i : \ v_i \neq 0\}|$. The matrix Φ is typically referred to as a *sensing matrix*. There are many algorithms (e.g. [6, 7, 10, 23]) to solve problem (2.1), but the one we shall focus on is the SubspacePursuit algorithm introduced in [18]:

Algorithm 2.1 SubspacePursuit ([18])

Inputs: $\mathbf{y}$, Φ and an integer $s \geq 1$

Initialization:

(1) $T^0 = \mathcal{L}_s(\Phi^\top\mathbf{y})$.
(2) $\mathbf{x}^0 = \text{argmin}\{\|\mathbf{y} - \Phi_{T^0}\mathbf{x}\|_2 : \ \text{supp}(\mathbf{x}) \subset T^0\}$
(3) $\mathbf{r}^0 = \mathbf{y} - \Phi_{T^0}\mathbf{x}^0$

Iteration:

for $k = 1 : m$ **do**

(1) $\hat{T}^k = T^{k-1} \cup \mathcal{L}_s\left(\Phi^\top\mathbf{r}^{k-1}\right)$
(2) $\mathbf{u} = \text{argmin}\{\|\mathbf{y} - \Phi_{\hat{T}^k}\mathbf{x}\|_2 : \ \mathbf{x} \in \mathbb{R}^N \text{ and } \text{supp}(\mathbf{x}) \subset \hat{T}^k\}$
(3) $T^k = \mathcal{L}_s(\mathbf{u})$ and $\mathbf{x}^k = \mathcal{H}_s(\mathbf{u})$
(4) $\mathbf{r}^k = \mathbf{y} - \Phi_{T^k}\mathbf{x}^k$

end for

Here $\mathcal{L}_s(\cdot)$ and $\mathcal{H}_s(\cdot)$ are thresholding operators:

$$\mathcal{L}_s(\mathbf{v}) := \{i \in [n] : \ v_i \text{ among } s \text{ largest-in-magnitude entries in } \mathbf{v}\}$$

$$\mathcal{H}_s(\mathbf{v})_i := \begin{cases} v_i & \text{if } i \in \mathcal{L}_s(\mathbf{v}) \\ 0 & \text{otherwise} \end{cases}$$

In quantifying when (2.1) has a unique solution, the following constant is often used (see [22])

Definition 2.1. *The s Restricted Isometry Constant (s-RIC) of $\Phi \in \mathbb{R}^{m\times n}$, written $\delta_s(\Phi)$, is defined to be the smallest value of $\delta > 0$ such that, for all $\mathbf{x} \in \mathbb{R}^n$ with $\|\mathbf{x}\|_0 \leq s$, we have:*

$$(1 - \delta)\|\mathbf{x}\|_2^2 \leq \|\Phi\mathbf{x}\|_2^2 \leq (1 + \delta)\|\mathbf{x}\|_2^2.$$

If $\delta_s(\Phi) < 1$ we often say that Φ has the Restricted Isometry Property *(RIP).*

Lemma 2.2. *For any $\Omega \subset [n]$ with $|\Omega| \geq s$ one can easily check that $\delta_s(\Phi_\Omega) \leq \delta_s(\Phi)$.*

Proof. This follows most easily from an alternative characterization of δ_s (see Chpt. 6 of [22]): that is, $\delta_s(\Phi) = \max_{S\subset[n],|S|\leq s} \|\Phi_S^\top \Phi_S - I\|_2$. Indeed, we have

$$\delta_s(\Phi_\Omega) = \max_{S'\subset\Omega,|S'|\leq s} \|\Phi_{S'}^\top \Phi_{S'} - I\|_2 \leq \max_{S\subset[n],|S|\leq s} \|\Phi_S^\top \Phi_S - I\|_2 = \delta_s(\Phi)$$

■

One of the reasons for the remarkable usefulness of compressive sensing is its robustness to error, both additive (*i.e.* in $\mathbf{y}$) and multiplicative (*i.e.* in Φ). More precisely, suppose that a signal $\hat{\mathbf{y}} = \hat{\Phi}\mathbf{x}^*$ is acquired, but that we do not know the sensing matrix $\hat{\Phi}$ precisely. Instead, we have access only to $\Phi = \hat{\Phi} + M$, for some small perturbation M. This models the scenario where a sensing matrix Φ is designed, and then implemented in hardware (for example as an MRI coil) where a certain amount of error becomes unavoidable. Suppose further that there is a small amount of noise in the measurement process, so that the signal we actually receive is $\mathbf{y} = \hat{\mathbf{y}} + \mathbf{e}$. Can one hope to approximate a sparse vector $\mathbf{x}^*$ from $\mathbf{y}$ well, given only Φ? This question is answered in the affirmative by several authors, starting with the work of [28]. For `SubspacePursuit`, we have the following result of Li:

Theorem 2.3. *Let $\mathbf{x}^*$, $\mathbf{y}$ $\hat{\mathbf{y}}$, Φ and $\hat{\Phi}$ be as above and suppose that $\|\mathbf{x}^*\|_0 \leq s$. For any $t \in [n]$, let $\delta_t := \delta_t(\Phi)$. Define the following constants:*

$$\epsilon_{\mathbf{y}} := \|\mathbf{e}\|_2/\|\hat{\mathbf{y}}\|_2 \text{ and } \epsilon_\Phi^s = \|M\|_2^s/\|\hat{\Phi}\|_2^s$$

where for any matrix B, $\|B\|_2^{(s)} := \max\{\|B_S\|_2 : S \subset [n] \text{ and } |S| = s\}$. Define further:

$$\rho = \frac{\sqrt{2\delta_{3s}^2(1+\delta_{3s}^2)}}{1-\delta_{3s}^2} \quad \text{and} \quad \tau = \frac{(\sqrt{2}+2)\delta_{3s}}{\sqrt{1-\delta_{3s}^2}}(1-\delta_{3s})(1-\rho) + \frac{2\sqrt{2}+1}{(1-\delta_{3s})(1-\rho)}$$

Assume $\delta_{3s} \leq 0.4859$ and let $\mathbf{x}^m$ be the output of `SubspacePursuit` *applied to problem (2.1) after m iterations. Then:*

$$\frac{\|\mathbf{x}^* - \mathbf{x}^m\|_2}{\|\mathbf{x}^*\|_2} \leq \rho^m + \tau\frac{\sqrt{1+\delta_s}}{1-\epsilon_\Phi^s}(\epsilon_\Phi^s + \epsilon_{\mathbf{y}}).$$

Proof. This is Corollary 1 in [35]. Note that our convention on hats is different to theirs — our Φ is their $\hat{\Phi}$, hence our ρ is their $\hat{\rho}$ and so on. ■

2.2. Cluster Extraction as Compressive Sensing. The eigenvectors of the Laplacian L are the key ingredient in Spectral Clustering algorithms. The following theorem is usually used in theoretical justifications of their success:

Theorem 2.4. *Let $C_1, \ldots, C_k$ denote the connected components of a graph G. Then the cluster indicator vectors $\mathbf{1}_{C_1}, \ldots, \mathbf{1}_{C_k}$ form a basis for the kernel of L.*

Proof. See proposition 4 of [36]. ■

Now suppose that G has clusters $C_1, \ldots, C_k$. By definition, clusters have few edges between them, and so it is useful to write G as the union of two edge-disjoint subgraphs, defined as follows: let $G^{\text{in}} = (V, E^{\text{in}})$ have only in-cluster edges, $E^{\text{in}} = \{\{i,j\} \in E : i,j \in C_a \text{ for } a \in [k]\}$, and let $G^{\text{out}} = (V, E^{\text{out}})$ consist only of the out-of-cluster edges, $\{\{i,j\} \in E : i \in C_a \text{ and } j \in C_b \text{ for } a \neq b\}$. Denote by A^{in} and L^{in} (resp. A^{out} and L^{out}) the adjacency matrix and Laplacian of G^{in} (resp. G^{out}). Similarly, d_i^{in} (resp. d_i^{out}) shall denote the degree of the vertex i in the graph G^{in} (resp. G^{out}). Note that $C_1, \ldots, C_k$ are now the connected components of G^{in}, and so $L^{\text{in}}\mathbf{1}_{C_a} = 0$ for all $a \in [k]$.

As $G = G^{\text{in}} \cup G^{\text{out}}$ we have $A = A^{\text{in}} + A^{\text{out}}$ and $d_i = d_i^{\text{in}} + d_i^{\text{out}}$. For future reference, define $r_i := d_i^{\text{out}}/d_i^{\text{in}}$. It is not the case that $L = L^{\text{in}} + L^{\text{out}}$, but we shall show in §3 that $L = L^{\text{in}} + M$ with $\|M\|_2 << \|L^{\text{in}}\|_2$. Without loss of generality assume that $v_1 \in C_1$ and denote $n_1 = |C_1|$. Let ℓ_i (resp. ℓ_i^{in}, ℓ_i^{out} and ℓ_i^+) denote the i-th column of L (resp. L^{in}, L^{out} and L^+). Then:

$$0 = L^{\text{in}}\mathbf{1}_{C_1} = [\ell_1^{\text{in}}, L_{-1}^{\text{in}}] \begin{bmatrix} 1 \\ \mathbf{1}_{C_1\setminus\{1\}} \end{bmatrix} = \ell_1^{\text{in}} + L_{-1}^{\text{in}}\mathbf{1}_{C_1\setminus\{1\}} \tag{2.2}$$

or in other words, $\mathbf{1}_{C_1\setminus\{1\}}$ is a solution to the linear system $L_{-1}^{\text{in}}\mathbf{x} = -\ell_1^{\text{in}}$. This system is underdetermined, but crucially $\|\mathbf{1}_{C_1\setminus\{1\}}\|_0 = n_1 - 1$. That is, as long as C_1 is not too large, $\mathbf{1}_{C_1\setminus\{1\}}$ is *sparse*. Thus we may hope to recover $\mathbf{1}_{C_1\setminus\{1\}}$ exactly by solving the problem:

$$\operatorname{argmin}\left\{\|L_{-1}^{\text{in}}\mathbf{x} + \ell_1^{\text{in}}\|_2 \text{ subject to } \|\mathbf{x}\|_0 \leq n_1 - 1\right\}.$$

Of course we do not have access to L^{in}. Instead, we have L, a noisy version of L^{in}. However, given that $L = L^{\text{in}} + M$, we may hope to use the results of §2.1, particularly Theorem 2.3, to show that if $\mathbf{x}^{\#}$ is the solution to:

$$\operatorname{argmin}\{\|L_{-1}\mathbf{x} + \ell_1\|_2 \text{ subject to } \|\mathbf{x}\|_0 \leq n_1 - 1\} \tag{2.3}$$

then $\mathbf{x}^{\#} \approx \mathbf{1}_{C_1\setminus\{1\}}$. Unfortunately problem (2.3) turns out to be poorly conditioned, as $\delta_{n_1-1}(L) \approx 1$. Thus, we propose a two-stage approach. In the first stage (Algorithm 2.2) we determine a superset $\Omega \supset C_1$ of size $(1+\epsilon)n_1$ while in the second stage (Algorithm 2.3) we extract C_1 from Ω by solving a compressive sensing problem to find a vector supported on *the complement* of C_1 in Ω. Specifically, observe that if $C_1 \subset \Omega$, then $0 = L^{\text{in}}\mathbf{1}_{C_1} = L_\Omega^{\text{in}}\mathbf{1}_{C_1}$. It follows that:

$$L_\Omega^{\text{in}}\mathbf{1}_\Omega = L_\Omega^{\text{in}}\left(\mathbf{1}_{C_1} + \mathbf{1}_{\Omega\setminus C_1}\right) = 0 + L_\Omega^{\text{in}}\mathbf{1}_{\Omega\setminus C_1} \Rightarrow L_\Omega^{\text{in}}\mathbf{1}_{\Omega\setminus C_1} = L_\Omega^{\text{in}}\mathbf{1}_\Omega. \tag{2.4}$$

Equivalently, if $\mathbf{y}^{\text{in}} := L_\Omega^{\text{in}}\mathbf{1}_\Omega = \sum_{i\in\Omega}\ell_i$ then $\mathbf{1}_{\Omega\setminus C_1}$ is the solution to

$$\operatorname{argmin}\{\|L_\Omega^{\text{in}}\mathbf{x} - \mathbf{y}^{\text{in}}\|_2 : \ \|\mathbf{x}\|_0 \leq \epsilon n_1\} \tag{2.5}$$

This problem is better conditioned, as we shall show that $\delta_{\epsilon n_1}(L_\Omega) = \epsilon + o(1)$. Clearly once $\mathbf{1}_{\Omega\setminus C_1}$ is known, we can find C_1 as $\Omega \setminus \operatorname{supp}\left(\mathbf{1}_{\Omega\setminus C_1}\right)$. In §3, we shall show that if we replace L_Ω^{in} and $\mathbf{y}^{\text{in}}$ with L_Ω and $\mathbf{y} := \sum_{i\in\Omega}\ell_i$ and let $\mathbf{x}^{\#}$ denote the solution to:

$$\operatorname{argmin}\{\|L_\Omega\mathbf{x} - \mathbf{y}\|_2 : \ \|\mathbf{x}\|_0 \leq \epsilon n_1\} \tag{2.6}$$

Then $\mathbf{x}^{\#} \approx \mathbf{1}_{\Omega \setminus C_1}$ and $\text{supp}(\mathbf{x}^{\#}) \approx \Omega \setminus C_1$. We now describe our algorithm in pseudocode. In line 3 of Algorithm 2.2, $\tilde{\mathcal{L}}_s$ denotes the thresholding operator defined as

$$\tilde{\mathcal{L}}_s(\mathbf{v}) = \{i \in [n] : \ v_i \text{ among } s \text{ largest entries in } \mathbf{v}\}.$$

Algorithm 2.2 Semi-Supervised Thresholding

Input: Adjacency matrix A, a thresholding parameter $\epsilon \in (0,1)$, $\Gamma \subset C$ and $\hat{n_0} \approx |C|$
Compute $L^+ = I + D^{-1}A$ and compute $\mathbf{b} = \sum_{i \in \Gamma} \ell_i^+$.
Let $\mathbf{v} = (L^+_{\Gamma^c})^\top \mathbf{b}$
Define $\widetilde{\Omega} = \tilde{\mathcal{L}}_{(1+\epsilon)\hat{n_0}}(\mathbf{v})$
Output: $\Omega = \widetilde{\Omega} \cup \Gamma$

Algorithm 2.3 `ClusterPursuit`

Input: Adjacency matrix $A \in \mathbb{R}^{n \times n}$, rejection parameter $R \in (0,1)$, Ω and sparsity parameter s
Compute $L = I - D^{-1}A$ and compute $\mathbf{y} = \sum_{i \in \Omega} \ell_i$. Let $\mathbf{x}^m$ be the solution to

$$\text{argmin}\{\|L_\Omega \mathbf{x} - \mathbf{y}\|_2 : \ \|\mathbf{x}\|_0 \leq s\} \tag{2.7}$$

obtained after $m = O(\log(n))$ iterations of `SubspacePursuit`
Output: $C^{\#} = \Omega \setminus W$, where $W^{\#} = \{i : \ x_i^m > R\}$.

Remark 2.5. Several comments on the parameters of Algorithms 2.2 and 2.3 are in order. A natural choice of R is $R = 0$, in which case $W^{\#}$ is simply the (non-negative) support of $\mathbf{x}$. If $|C|$ is known, then setting $\hat{n_0} = |C|$ in Algorithm 2.2 and $s = \epsilon|C|$ in Algorithm 2.4 is natural, as $|\Omega \setminus C| = \epsilon \hat{n_0}$. In practice, the size of C is only approximately known, and we have found greater success with setting $\hat{n_0}$ to be an upper bound on the expected size of $|C|$, while setting $s = 1.2\epsilon\hat{n_0}$ and $R \approx 0.5$. This allows `ClusterPursuit` to explore a greater range of cluster sizes, as $|W^{\#}|$ is between 0 and s for any $R > 0$, hence $|C^{\#}|$ is between $|\Omega|$ and $|\Omega| - s$. That m can be taken to be $O(\log(n))$ will follow from the proof of Theorem 3.15. In practice, we set $m = 5\log(n)$.

Algorithm 2.4 Semi-Supervised Cluster Pursuit (`SSCP`)

Input: Adjacency matrix A, parameters $\epsilon, R \in (0,1)$, $\Gamma \subset C$, $\hat{n_0} \approx |C|$ and $s \approx \epsilon\hat{n_0}$.
Step 1 Perform Algorithm 2.2 with input $(A, \epsilon, \Gamma, \hat{n_0})$ to obtain Ω.
Step 2 Perform Algorithm 2.3 (`ClusterPursuit`) with input (A, R, Ω, s) to obtain $C^{\#}$.
Output: $C^{\#}$

3. Theoretical Analysis.

In this section we prove that `SSCP` is weakly consistent for the SSBM. Without loss of generality we assume we are trying to extract C_1. Our main result is:

Theorem 3.1. *Let $G \sim SSBM(n,k,p,q)$ with $p = \omega\log(n_0)/n_0$ for ω such that $\omega \to \infty$ as $n \to \infty$, $q = b\log(n)/n$ for b constant and $k = O(1)$. Let Γ be a set of gn_0 vertices drawn uniformly at random from C_1, where $g \in (0,1)$ is independent of n_0. Fix any $\epsilon \in (0, 0.15)$, set $R = 0$, $\hat{n_0} = n_0$ and $s = \epsilon n_0$. Let $C_1^{\#}$ denote the output of* SSCP *run with these inputs. Then:*

$$\mathbb{P}\left[\frac{\left|C_1 \Delta C_1^{\#}\right|}{|C_1|} \leq o(1)\right] = 1 - o(1).$$

Proof. In Theorem 3.8 we show that Algorithm 2.2 returns an Ω containing a fraction $1 - o(1)$ of the vertices of C_1 with probability $1 - o(1)$. Theorem 3.15 will then show that given such an Ω, `ClusterPursuit` will output a cluster $C_1^{\#}$ such that $|C_1^{\#} \Delta C_1| = o(n_0)$ with probability $1 - o(1)$, completing the proof. ∎

Henceforth, when an event happens with probability $1 - o(1)$, we shall say it happens *almost surely*, or *a.s.*. Note that if a finite collection of events happen almost surely, then their intersection also occurs almost surely. We shall use this observation repeatedly.

3.1. Concentration in Erdős - Rènyi Graphs. The proof of Theorem 3.1 relies on two *concentration phenomena* in Erdős - Rènyi graphs. The first is that the maximum and minimum degrees of an Erdős - Rènyi graph are within a small deviation of their expected value, *a.s.* The second is that the second eigenvalue of the Laplacian of an ER graph is within an $o(1)$ term of its expected value, *a.s.*

Theorem 3.2 (see [8, 9]). *Let $G \sim ER(n,q)$ with $q = (b + o(1))\log(n)/n$. There exist a function $\eta_\Delta(b)$ satisfying $0 < \eta_\Delta(b) < 1$ and $\lim_{b\to\infty} \eta_\Delta(b) = 0$ such that*

$$d_{max}(G) = (1 + \eta_\Delta(b))b\log n + o(1) \leq 2b\log(n) + o(1) \text{ a.s.}$$

Theorem 3.3 (see [24], Theorem 3.4 (ii)). *If $G \sim ER(n_0,p)$ with $p = \omega\log(n_0)/n_0$ where $\omega \to \infty$, then $d_{\min}(G) = (1 - o(1))\omega\log(n_0)$ and $d_{\max}(G) = (1 + o(1))\omega\log(n_0)$ a.s.*

Theorem 3.4. *Suppose that $G \sim ER(n_0,p)$ with $p = \omega\log(n_0)$ where $\omega \to \infty$. Then we have almost surely (1) $\lambda_{\max}(A) \leq (1+o(1))\omega\log(n_0)$; (2) $\lambda_i(A) \leq o(\omega\log(n_0))$ for $\lambda_i < \lambda_{\max}$; and (3) $|\lambda_i(L) - 1| \leq \sqrt{\frac{6\log(2n_0)}{\omega\log(n_0)}} = o(1)$ for all $i > 1$.*

Proof. See Theorems 3 and 4 in [16]. In their notation, $m = w_{\min} = pn_0 = \omega\log n_0$. Their results refer to L^{sym}, but one can easily show that L^{sym} and L have the same spectrum. ∎

3.2. Reducing from the SBM to the ER model. Let G^{in} and G^{out} be as in §2.2. If $G \sim \text{SSBM}(n,k,p,q)$ then G^{in} consists of k disjoint i.i.d graphs, $G_{C_a} \sim \text{ER}(n_0,p)$. The graph G^{out} is not an Erdős - Rènyi graph, as there is 0 probability of it containing an edge between two vertices in the same cluster (because we have removed them). However, we can profitably think of G^{out} as a subgraph of some $\widetilde{G^{\text{out}}} \sim \text{ER}(n,q)$. In particular, any upper bounds on the degrees of vertices in $\widetilde{G^{\text{out}}}$ are automatically bounds on the degrees in G^{out}. Thus, we have the following corollaries of Theorems 3.3 and 3.2:

Corollary 3.5. *If $G \sim SSBM(n,k,p,q)$ with $q = b\log(n)/n$, $d^{out}_{\max}(G) \le 2b\log n + o(1)$ a.s.*

Proof. Consider G^{out} as a subgraph of $\widetilde{G^{\text{out}}} \sim \text{ER}(n,q)$ and apply Theorem 3.2 ∎

Corollary 3.6. *If $G \sim SSBM(n,k,p,q)$ with $k = O(1)$ and $p = \omega\log(n_0)/n_0$ where $\omega \to \infty$, then $d^{in}_{\min}(G) = (1-o(1))\omega\log(n_0)$ and $d^{in}_{\max}(G) = (1+o(1))\omega\log(n_0)$ a.s.*

Proof. If $i \in C_a$ then $d^{\text{in}}_i = d_i(G_a)$, where $G_a = G_{C_a} \sim \text{ER}(n_0,p)$. Clearly:

$$d^{\text{in}}_{\max}(G) = \max_i d^{\text{in}}_i = \max_a d_{\max}(G_a)$$

By Theorem 3.3, $d_{\max}(G_a) = (1+o_{n_0}(1))\omega\log(n_0)$ a.s. Note that the $d_{\max}(G_a)$ are i.i.d random variables, and since we are taking a maximum over $k = \mathcal{O}(1)$ of them, it follows that $\max_a d_{\max}(G_a) \le (1+o_{n_0}(1))\omega\log(n_0)$ a.s. Moreover, as $n_0 = n/k$, $o_{n_0}(1) = o_n(1)$. The proof for $d^{\text{in}}_{\min}(G)$ is similar. ∎

Corollary 3.7. *$G \sim SSBM(n,k,p,q)$ with $p = \omega\log(n_0)/n_0$ where $\omega \to \infty$, $q = b\log(n)/n$ and $k = O(1)$. Recall that $r_i := d^{out}_i/d^{in}_i$. Then $r_{\max} \le d^{out}_{\max}/d^{in}_{\min} = o(1)$ a.s.*

Proof. First of all, it is clear that for any i, $d^{\text{out}}_i/d^{\text{in}}_i \le d^{\text{out}}_{\max}/d^{\text{in}}_{\min}$. From Corollaries 3.5 and 3.6 we have:

$$\begin{aligned}\frac{d^{\text{out}}_{\max}}{d^{\text{in}}_{\min}} &\le \frac{2b\log n + o(1)}{(1-o(1))\omega\log(n_0)} = \frac{2b\log n + o(1)}{(1-o(1))\omega(\log(n)-\log(k))} \quad \text{as } n = kn_0 \\ &= \frac{2b+o(1)}{(1-o(1))\omega(1-o(1))} = o(1) \text{ since } k = O(1) \text{ and } \omega \to \infty\end{aligned}$$

∎

3.3. Reliably Finding Supersets.

Let Ω denote the output of Algorithm 2.2, run with inputs as in Theorem 3.1. Further, let $U = C_1 \setminus (C_1 \cap \Omega)$ denote the "missed" indices, and $W = \Omega \setminus (C_1 \cap \Omega)$ denote the "bad" indices (i.e. vertices in Ω that are not in C_1). Let $|U| = un_0$, in which case $|W| = (\epsilon+u)n_0$, as by construction $|\Omega| = (1+\epsilon)n_0$. We prove that $u = o(1)$:

Theorem 3.8. *Let $G \sim SSBM(n,k,p,q)$ with $k = O(1)$, $p = \omega\log(n_0)/n_0$ with $\omega \to \infty$ and $q = b\log(n)/n$. Let $\Gamma \subset C_1$ with $|\Gamma| = gn_0$ for some constant $g \in (0,1)$. For any $\epsilon > 0$, if Ω is the output of Algorithm 2.2, with inputs ϵ, Γ and n_0, then $|C_1 \setminus (C_1 \cap \Omega)| = o(n_0)$.*

Proof. As in line 3 of Algorithm 2.2, define $\mathbf{v} := (L^+_{\Gamma^c})^\top \mathbf{b}$, where $\mathbf{b} = \sum_{i\in\Gamma} \ell^+_i$. Observe:

$$\left((L^+)^T \ell^+_j\right)_i = \langle \ell^+_i, \ell^+_j \rangle = \left(\frac{1}{d_i} + \frac{1}{d_j}\right) A_{ij} + \sum_{k=1}^{n} \frac{A_{ik}A_{kj}}{d_k^2}. \tag{3.1}$$

By the definition of the thresholding operator $\mathcal{L}(\cdot)$, we must have $v_i \le v_j$ for every $i \in U$ and $j \in W$. We sum first over W and then sum over U to have

$$(\epsilon+u)n_0 v_i \le \sum_{j\in W} v_j \text{ and } (\epsilon+u)n_0 \sum_{i\in U} v_i \le un_0 \sum_{j\in W} v_j,$$

respectively. It follows that:

$$\sum_{i\in U} v_i \leq \frac{u}{\epsilon+u}\sum_{j\in W} v_j \leq \sum_{j\in W} v_j. \tag{3.2}$$

Looking ahead, we shall show that if inequality (3.2) holds then $u = o(1)$. Now:

$$\sum_{i\in U} v_i = \sum_{i\in U}\left((L^+_{\Gamma^c})^\top \mathbf{b}\right)_i = \sum_{i\in U}\left(\sum_{j\in\Gamma}(L^+_{\Gamma^c})^\top \ell^+_j\right)_i = \sum_{i\in U}\sum_{j\in\Gamma}\langle \ell^+_i, \ell^+_j\rangle.$$

From equation (3.1) we deduce that $\langle \ell^+_i, \ell^+_j\rangle \geq \sum_{k=1}^n \frac{A_{ik}A_{kj}}{d_k^2}$. Moreover:

$$\sum_{k=1}^n \frac{A_{ik}A_{kj}}{d_k^2} \geq \frac{1}{d^2_{\max}}\sum_{k=1}^n A_{ik}A_{kj} \geq \frac{1}{d^2_{\max}}\sum_{k\in C_1} A_{ik}A_{kj}$$

and so:

$$\sum_{i\in U} v_i \geq \frac{1}{d^2_{\max}}\sum_{i\in U}\sum_{j\in\Gamma}\sum_{k\in C_1} A_{ik}A_{kj} \tag{3.3}$$

The triple sum above is precisely the number of length two paths from U to Γ contained in the Erdős - Rènyi graph $G_{C_1} \sim \mathrm{ER}(n_0, p)$. In [14] a neat formula for this quantity, which they call it $e_2(U,\Gamma)$, is given. Specifically, they show that for any family of graphs $\mathcal{G}_p$ such that for $G \sim \mathcal{G}_p$ we have $\lambda_1(A) = (1+o(1))pn$ and $\lambda_i(A) = o(pn)$ for $i \geq 2$, then for any $X, Y \subset V$:

$$\left|e_2(X,Y) - p^2 n|X||Y|\right| = o(p^2 n^3)$$

As the aforementioned condition on the eigenvalues of A holds for $\mathrm{ER}(n_0,p)$ a.s. (see Theorem 3.4) we conclude that

$$\begin{aligned}\sum_{i\in U}\sum_{j\in\Gamma}\sum_{k\in C_1} A_{ik}A_{kj} &= e_2(U,\Gamma) \geq p^2 n_0|U||\Gamma| - o(p^2 n_0^3) \text{ a.s.}\\ &= \left(\frac{\omega^2\log^2(n_0)}{n_0^2}\right) n_0(un_0)(gn_0) - o\left(\frac{\omega^2\log^2(n_0)}{n_0^2}n_0^3\right)\\ &= ug\omega^2\log^2(n_0)n_0 - o(\omega^2\log^2(n_0)n_0).\end{aligned}$$

By Corollaries 3.5 and 3.6 above, $d_{\max} \leq d^{\text{in}}_{\max} + d^{\text{out}}_{\max} \leq (1+o(1))\omega\log(n_0) + 2b\log n + o(1) = (1+o(1))\omega\log(n_0)$ a.s.. Putting this all together we get that:

$$\sum_{i\in U} v_i \geq ugn_0 - o(n_0) \quad \text{a.s.} \tag{3.4}$$

We now consider the right hand side of (3.2). Rewrite the sum as an inner product:

$$\sum_{j\in W} v_j = \sum_{j\in W} 1 v_j = \langle \mathbf{1}_W, \mathbf{v}\rangle.$$

In a similar vein, rewrite $\mathbf{b} = \sum_{i\in\Gamma} \ell_i^+ = L^+\mathbf{1}_\Gamma$. Now recall that $\mathbf{v} = (L^+)^\top \mathbf{b} = (L^+)^\top L^+\mathbf{1}_\Gamma$. It follows that:

$$\sum_{j\in W} v_j = \langle \mathbf{1}_W, \mathbf{v}\rangle = \langle \mathbf{1}_W, \left(L^+\right)^\top L^+\mathbf{1}_\Gamma\rangle = \langle L^+\mathbf{1}_W, L^+\mathbf{1}_\Gamma\rangle$$

Split L^+ into four submatrices as follows:

$$L^1 \in \mathbb{R}^{n_0\times n_0} : \; L^1_{ij} = L^+_{ij} \text{ for } i,j \in C_1;$$
$$L^2 \in \mathbb{R}^{n_0\times (n-n_0)} : \; L^2_{ij} = L^+_{ij} \text{ for } i \in C_1, j \notin C_1;$$
$$L^3 \in \mathbb{R}^{(n-n_0)\times n_0} : \; L^3_{ij} = L^+_{ij} \text{ for } i \notin C_1, j \in C_1;$$
$$L^4 \in \mathbb{R}^{(n-n_0)\times (n-n_0)} : \; L^4_{ij} = L^+_{ij} \text{ for } i,j \in C_1^c.$$

If we imagine the vertices to be ordered such that $C = \{1,\dots,n_0\}$ and $C_1^c = \{n_0+1,\dots,n\}$ then this decomposition looks like $L^+ = \left[\begin{smallmatrix} L^1 & L^2 \\ L^3 & L^4 \end{smallmatrix}\right]$. Because $W \subset C^c$ and $\Gamma \subset C$:

$$L^+\mathbf{1}_\Gamma = \begin{bmatrix} L^1\mathbf{1}_\Gamma \\ L^3\mathbf{1}_\Gamma \end{bmatrix} \text{ and } L^+\mathbf{1}_W = \begin{bmatrix} L^2\mathbf{1}_W \\ L^4\mathbf{1}_W \end{bmatrix}.$$

Hence, we have $\langle L^+\mathbf{1}_W, L^+\mathbf{1}_\Gamma\rangle = \langle L^2\mathbf{1}_W, L^1\mathbf{1}_\Gamma\rangle + \langle L^4\mathbf{1}_W, L^3\mathbf{1}_\Gamma\rangle$. In the lemma below, we provide bounds on $\|L^i\|_1$ and $\|L^i\|_\infty$ for $i = 1,\dots,4$. We use these bounds to finish the proof:

$$\langle L^2\mathbf{1}_W, L^1\mathbf{1}_\Gamma\rangle \le \|L^2\mathbf{1}_W\|_\infty \|L^1\mathbf{1}_\Gamma\|_1 \le \|L^2\|_\infty \|\mathbf{1}_W\|_\infty \|L^1\|_1 \|\mathbf{1}_\Gamma\|_1 \le (o(1))(1)(2)|\Gamma|,$$
$$\langle L^4\mathbf{1}_W, L^3\mathbf{1}_\Gamma\rangle \le \|L^4\mathbf{1}_W\|_\infty \|L^3\mathbf{1}_\Gamma\|_1 \le \|L^4\|_\infty \|\mathbf{1}_W\|_\infty \|L^3\|_1 \|\mathbf{1}_\Gamma\|_1 \le (2+o(1))(1)(o(1))|\Gamma|.$$

Both terms are bounded by $g\left(o(n_0)\right)$. Hence:

$$\sum_{j\in W} v_j = \langle L^+\mathbf{1}_W, L^+\mathbf{1}_\Gamma\rangle = \langle L^2\mathbf{1}_W, L^1\mathbf{1}_\Gamma\rangle + \langle L^4\mathbf{1}_W, L^3\mathbf{1}_\Gamma\rangle = g\left(o(n_0)\right)$$

Returning to (3.2), we have $ugn_0 - o(1) \le g\left(o(n_0)\right)$ and so $u \le o(1) + o(1/n_0) = o(1)$ a.s. ■

Lemma 3.9. *Let L^1, L^2, L^3 and L^4 be as in the above proof. Then $\|L^2\|_\infty, \|L^3\|_1 \le o(1)$, $\|L^1\|_1 \le 2$ and $\|L^4\|_\infty \le 2 + o(1)$ a.s.*

Proof. For any matrix B, $\|B\|_1 = \max_i \sum_j |B_{ij}|$ and $\|B\|_\infty = \max_j \sum_i |B_{ij}|$. Now:

$$\|L^2\|_\infty = \max_{j\in C_1^c} \sum_{i\in C_1} \left|L^+_{ij}\right| = \max_{j\in C_1^c} \sum_{i\in C_1} \frac{A^{\text{out}}_{ij}}{d_i} \le \frac{1}{d_{\min}} \max_{j\in C_1^c} d^{\text{out}}_j \le \frac{d^{\text{out}}_{\max}}{d_{\min}} = o(1) \text{ by Corollary 3.7}$$

and the proof for $\|L^3\|_1$ is very similar. For L^1:

$$\|L^1\|_1 = \max_{i\in C_1} \sum_{j\in C_1} \left|L^+_{ij}\right| = \max_{i\in C_1} \left(1 + \sum_{j\in C_1} \frac{A^{\text{in}}_{ij}}{d_i}\right) = \max_{i\in C_1} \left(1 + \frac{d^{\text{in}}_i}{d_i}\right) \le 2$$

while for L^4:

$$\|L^4\|_\infty = \max_{j\in C_1^c} \sum_{i\in C_1^c} \left|L^+_{ij}\right| = \max_{j\in C_1^c} \left(1 + \sum_{i\in C_1^c} \frac{A^{\text{in}}_{ij}}{d_i}\right) \le \max_{j\in C_1^c} \left(1 + \frac{1}{d^{\text{in}}_{\min}} \sum_{i\in C_1^c} A^{\text{in}}_{ij}\right) \le 1 + \frac{d^{\text{in}}_{\max}}{d^{\text{in}}_{\min}}$$

and by Corollary 3.6, $d^{\text{in}}_{\max}/d^{\text{in}}_{\min} = (1+o(1))/(1-o(1)) = 1+o(1)$. ∎

3.4. Extracting C_1 from Ω. As mentioned in §2.2, it is not the case that $L = L^{\text{in}} + L^{\text{out}}$. Instead, we write $L = L^{\text{in}} + M$, where M can be thought of as a perturbation, or error, term:

Theorem 3.10. *Suppose that $G \sim SSBM(n,k,p,q)$ with $p = \omega\log(n_0)/n_0$ with $\omega \to \infty$, $q = b\log(n)/n$ and $k = \mathcal{O}(1)$. Then $\|M\|_2 \le o(1)$.*

Proof. Letting δ_{ij} denote the Kronecker delta symbol, observe that

$$L_{ij} := \delta_{ij} - \frac{1}{d_i} A_{ij} = \delta_{ij} - \frac{1}{d^{\text{in}}_i + d^{\text{out}}_i}\left(A^{\text{in}}_{ij} + A^{\text{out}}_{ij}\right).$$

We shall use the following easily verifiable one dimensional version of the Woodbury formula:

$$\frac{1}{d^{\text{in}}_i + d^{\text{out}}_i} = \frac{1}{d^{\text{in}}_i} - \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right)$$

Thus:

$$\begin{aligned}
L_{ij} &= \delta_{ij} - \left(\frac{1}{d^{\text{in}}_i} - \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right)\right)\left(A^{\text{in}}_{ij} + A^{\text{out}}_{ij}\right) \\
&= \left(\delta_{ij} - \frac{1}{d^{\text{in}}_i} A^{\text{in}}_{ij}\right) - \frac{1}{d^{\text{in}}_i} A^{\text{out}}_{ij} + \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right)\left(A^{\text{in}}_{ij} + A^{\text{out}}_{ij}\right) \\
&= L^{\text{in}}_{ij} - \frac{1}{d^{\text{in}}_i}\left(1 - \frac{r_i}{r_i+1}\right) A^{\text{out}}_{ij} + \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right) A^{\text{in}}_{ij} \\
&= L^{\text{in}}_{ij} - \frac{1}{d^{\text{in}}_i}\left(\frac{1}{r_i+1}\right) A^{\text{out}}_{ij} + \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right) A^{\text{in}}_{ij}.
\end{aligned}$$

That is, $M_{ij} = -\frac{1}{d^{\text{in}}_i}\left(\frac{1}{r_i+1}\right) A^{\text{out}}_{ij} + \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right) A^{\text{in}}_{ij}$. To bound the spectral norm we use Gershgorin's disks, noting that $M_{ii} = 0$ for all i:

$$\begin{aligned}
\|M\|_2 &= \max_i\{|\lambda_i| : \ \lambda_i \text{ eigenvalue of } M\} \le \max_i \sum_j |M_{ij}| \\
&= \max_i \frac{1}{d^{\text{in}}_i}\left(\frac{1}{r_i+1}\right)\sum_j A^{\text{out}}_{ij} + \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right)\sum_j A^{\text{in}}_{ij} \\
&= \max_i \left\{\frac{1}{d^{\text{in}}_i}\left(\frac{1}{r_i+1}\right)(d^{\text{out}}_i) + \frac{1}{d^{\text{in}}_i}\left(\frac{r_i}{r_i+1}\right)(d^{\text{in}}_i)\right\} \\
&= \max_i \left\{\left(\frac{r_i}{r_i+1}\right) + \left(\frac{r_i}{r_i+1}\right)\right\} \le 2r_{\max} = o(1) \text{ a.s. by Corollary 3.7}
\end{aligned}$$ ∎

Recall that `ClusterPursuit` works by running `SubspacePursuit` for m iterations on the compressive sensing problem: $\text{argmin}\{\|L_\Omega \mathbf{x} - \mathbf{y}\|_2 : \|\mathbf{x}\|_0 \leq s\}$ to obtain $\mathbf{x}^m$, and then obtaining an approximation to $W = \Omega \setminus (C_1 \cap \Omega)$ by considering the support of $\mathbf{x}^m$. We now use the theory of §2.1 to show that this is a provably good approximation. From equation (2.5) we have that $\mathbf{1}_{\Omega \setminus C_1}$ is a solution to:

$$\text{argmin}\{\|L_\Omega^{\text{in}} \mathbf{x} - \mathbf{y}^{\text{in}}\|_2 : \|\mathbf{x}\|_0 \leq \epsilon n_0\}$$

Under the assumption that $|\Omega| = (1+\epsilon)n_0$ and $C_1 \subset \Omega$. What if C_1 is not completely contained in Ω?

Lemma 3.11. *Suppose that* $|C_1 \setminus (\Omega \cap C_1)| = o(n_0)$. *Then*

$$L_\Omega^{in} \mathbf{1}_{\Omega \setminus (\Omega \cap C_1)} = L_\Omega^{in} \mathbf{1}_\Omega + \mathbf{e}_1$$

where $\|\mathbf{e}\|_2 = o(\sqrt{n_0})$.

Proof. Let $U := C_1 \setminus (\Omega \cap C_1)$ and $W := \Omega \setminus (\Omega \cap C_1)$. Then

$$L^{\text{in}} \mathbf{1}_\Omega + L^{\text{in}} \mathbf{1}_U = L^{\text{in}} (\mathbf{1}_{C_1 \cap \Omega} + \mathbf{1}_W) + L^{\text{in}} \mathbf{1}_U = L^{\text{in}} (\mathbf{1}_{C_1 \cap \Omega} + \mathbf{1}_U) + L^{\text{in}} \mathbf{1}_W = 0 + L^{\text{in}} \mathbf{1}_W \tag{3.5}$$

as $\mathbf{1}_{C_1 \cap \Omega} + \mathbf{1}_U = \mathbf{1}_{C_1}$. Letting $\mathbf{e}_1 = L^{\text{in}} \mathbf{1}_U$ we have the result as $\|\mathbf{e}_1\|_2 \leq \|L^{\text{in}}\|_2 \|\mathbf{1}_U\|_2 = (2)(\sqrt{|U|}) = 2o(\sqrt{n_0})$ ∎

Of course we do not have access to $\mathbf{y}^{\text{in}}$, only $\mathbf{y}$. In the next lemma we prove that this introduces an error term with ℓ_2 norm of order $o(\sqrt{n_0})$.

Lemma 3.12. *Let* $\mathbf{y} := \sum_{i \in \Omega} \ell_i$ *and* $\mathbf{y}^{in} = \sum_{i \in \Omega} \ell_i^{in}$. *Then* $\mathbf{y} = \mathbf{y}^{in} + \mathbf{e}_2$ *with* $\|\mathbf{e}_2\|_2 = o(\sqrt{n_0})$

Proof. Clearly $\mathbf{e}_2 := \mathbf{y} - \mathbf{y}^{\text{in}} = L\mathbf{1}_\Omega - L^{\text{in}} \mathbf{1}_\Omega = M \mathbf{1}_\Omega$. By Theorem 3.10, $\|M\|_2 \leq o(1)$. So ∎

$$\|\mathbf{e}_2\|_2 \leq \|M\|_2 \|\mathbf{1}_\Omega\|_2 \leq o(1) \left(\sqrt{(1+\epsilon)n_0}\right) = o(\sqrt{n_0}).$$

The net result of Lemma 3.11 and 3.12 is that $L_\Omega^{\text{in}} \mathbf{1}_{\Omega \setminus \Omega \cap C_1} = \mathbf{y}^{\text{in}} - \mathbf{e}_2 + \mathbf{e}_1 =: \mathbf{y}^{\text{in}} + \mathbf{e}$ with $\|\mathbf{e}\|_2 = o(\sqrt{n_0})$. In the notation of Theorem 2.3, we think of L_Ω as Φ, the noisy measurement matrix, and L_Ω^{in} as $\widehat{\Phi}$. Similarly, we think of $\mathbf{y}^{\text{in}}$ as $\hat{\mathbf{y}}$, and the $\mathbf{y}$ defined above as the noisy signal.

Theorem 3.13. *Let* $G \sim SSBM(n, k, p, q)$ *with* $p = \omega \log(n_0)/n_0$ *and* $q = b \log(n)/n$, *where* $\omega \to \infty$. *Suppose further that* $k = O(1)$. *For any* $t = \gamma n_0$ *with* $\gamma \in (0,1)$, $\delta_t(L_\Omega) \leq \gamma + o(1)$ *almost surely.*

Proof. This proof is deferred to the appendix. ∎

Finally, we compute the various constants necessary to apply Theorem 2.3.

Lemma 3.14. *Let* $G \sim SSBM(n, k, p, q)$ *with* $p = \omega \ln(n)/n$ *and* $q = b \ln(n)/n$ *where* $\omega \to \infty$. *Suppose further that* $k = O(1)$. *For any* $s = \epsilon n_0$ *with* $0 < \epsilon < 0.15$, *we have that* $\rho \leq 0.8751$,$\tau = O(1)$ *and* $\epsilon_\Phi^s, \epsilon_{\mathbf{y}} = o(1)$ *a.s. (these quantities are all defined in Theorem 2.3).*

Proof. We leave the proof to the appendix. ∎

Putting all of the above together, we can show that `ClusterPursuit` succeeds, i.e. if $C_1^{\#}$ is the output and C_1 is the true cluster, then $|C_1 \Delta C_1^{\#}| = o(n_0)$.

Theorem 3.15. *Let $G \sim SSBM(n,k,p,q)$ with $k = \mathcal{O}(1)$ and $p = \omega \log(n_0)/n_0$, $q = b\log(n)/n$, where $\omega \to \infty$. Suppose that, for $\epsilon < 0.15$, $\Omega \subset [n]$ is such that $|\Omega| = (1+\epsilon)n_0$ and $|C_1 \setminus (\Omega \cap C_1)| = o(n_0)$. Let $C_1^{\#}$ denote the output of* `ClusterPursuit` *with inputs $R = 0$, Ω and $s = \epsilon n_0$. Then $|C_1^{\#} \Delta C_1| = o(n_0)$ a.s.*

Proof. By Theorem 3.13, $\delta_s := \delta_s(L_\Omega) \leq \epsilon + o(1)$ and $\delta_{3s} := \delta_{3s}(L_\Omega) \leq 3\epsilon + o(1)$. Since $3\epsilon < 0.45$, we may take the $o(1)$ term to be small enough such that $\delta_{3s}(L_\Omega) \leq 0.45$. We now appeal to Theorem 2.3, using the values of $\rho, \tau, \epsilon_\Phi$ and $\epsilon_{\mathbf{y}}$ computed in Lemma 3.14. Let $\mathbf{x}^m$ denote the output of `SubspacePursuit` run for m iterations on the problem

$$\operatorname{argmin}\{\|L_\Omega \mathbf{x} - \mathbf{y}\|_2 : \ \|\mathbf{x}\|_0 \leq \epsilon n_0\}. \tag{3.6}$$

By Theorem 2.3, we have that

$$\frac{\|\mathbf{1}_{\Omega\setminus(\Omega\cap C_1)} - \mathbf{x}^m\|_2}{\|\mathbf{1}_{\Omega\setminus(\Omega\cap C_1)}\|_2} \leq \rho^m + \tau \frac{\sqrt{1+\delta_s}}{1-\epsilon_\Phi^s}(\epsilon_\Phi^s + \epsilon_{\mathbf{y}})$$

By Lemma 3.14, the second term on the right-hand side is $o(1)$. Taking $m = \log_\rho(1/n) = O(\log(n))$, we obtain that $\rho^m = 1/n = o(1)$ and so:

$$\frac{\|\mathbf{1}_{\Omega\setminus(\Omega\cap C_1)} - \mathbf{x}^m\|_2}{\|\mathbf{1}_{\Omega\setminus(\Omega\cap C_1)}\|_2} \leq o(1)$$

As before, define $U = C_1 \setminus (\Omega \cap C_1)$. By assumption $|U| = o(n_0)$. It follows that $|\Omega \setminus (\Omega \cap C_1)| = |\Omega| - |\Omega \cap C_1| = (1+\epsilon)n_0 - (n_0 - |U|) = \epsilon n_0 + o(n_0)$. Hence $\|\mathbf{1}_{\Omega\setminus(\Omega\cap C_1)}\|_2 = \sqrt{\epsilon n_0} + o(\sqrt{n_0})$ and thus:

$$\|\mathbf{1}_{\Omega\setminus(\Omega\cap C_1)} - \mathbf{x}^m\|_2 \leq o(\sqrt{n_0}).$$

From the following lemma, it follows that $|\operatorname{supp}(\mathbf{x}^m) \Delta (\Omega \setminus (\Omega \cap C_1))| = o(n_0)$, and consequently, as $C_1^{\#} = \Omega \setminus \operatorname{supp}(\mathbf{x}^m)$ we have that $|C_1^{\#} \Delta (\Omega \cap C_1)| = o(n_0)$. Accounting for U, we have that

$$|C^{\#} \Delta C_1| = |C^{\#} \Delta (\Omega \cap C_1)| + |U| = o(n_0) + o(n_0) = o(n_0) \text{ a.s.}$$

∎

Lemma 3.16. *Let $T \subset [n]$ and $\mathbf{v} \in \mathbb{R}^n$. If $\|\mathbf{1}_T - \mathbf{v}\|_2 \leq D$ and $|\operatorname{supp}(\mathbf{v})| \leq |T|$ then $|T \Delta supp(\mathbf{v})| \leq 2D^2$.*

Proof. Recall that $T \Delta \operatorname{supp}(\mathbf{v}) = (T \setminus (T \cap \operatorname{supp}(\mathbf{v}))) \cup (\operatorname{supp}(\mathbf{v}) \setminus (T \cap \operatorname{supp}(\mathbf{v})))$ and these two sets are disjoint. Now:

$$\begin{aligned} &|T \setminus (T \cap \operatorname{supp}(\mathbf{v}))| = |T| - |T \cap \operatorname{supp}(\mathbf{v})| \\ \text{and } &|\operatorname{supp}(\mathbf{v}) \setminus (T \cap \operatorname{supp}(\mathbf{v}))| = |\operatorname{supp}(\mathbf{v})| - |T \cap \operatorname{supp}(\mathbf{v})| \leq |T| - |T \cap \operatorname{supp}(\mathbf{v})| \\ \Rightarrow &|T \Delta \operatorname{supp}(\mathbf{v})| \leq 2\left(|T| - |T \cap \operatorname{supp}(\mathbf{v})|\right) = 2\left(T \setminus (T \cap \operatorname{supp}(\mathbf{v}))\right). \end{aligned}$$

But $T \setminus (T \cap \mathrm{supp}(\mathbf{v}))$ cannot be too large as:

$$D \geq \|\mathbf{1}_T - \mathbf{v}\|_2 \geq \| (\mathbf{1}_T - \mathbf{v}) |_{T\setminus(T\cap\mathrm{supp}(\mathbf{v}))}\|_2 = \|\mathbf{1}_{T\setminus(T\cap\mathrm{supp}(\mathbf{v}))}\|_2 = \sqrt{|T \setminus (T \cap \mathrm{supp}(\mathbf{v}))\,|}.$$

Thus $|T \setminus (T \cap \mathrm{supp}(\mathbf{v}))\,| \leq D^2$, and the result follows. ∎

3.5. Computational Complexity. Here we bound the operation count required by `SSCP`. For continuity, we focus on the case where $G \sim \mathrm{SSBM}(n, k, p, q)$ with parameters as in Theorem 3.1. Our analysis is inspired by the analysis of a similar algorithm, `CoSaMP`, in [37].

Theorem 3.17. *Suppose* `SSCP` *is run on* $G \sim SSBM(n, k, p, q)$ *with parameters exactly as in Theorem 3.1. If* $\omega = O(\log(n))$*, then* `SSCP` *requires* $O(n\log^3(n))$ *operations.*

Proof. Assume throughout that A is stored as a sparse matrix. There are three main steps in `SSCP`, namely: (1) Computing L and L^+; (2) The thresholding step of Algorithm 2; and (3) Solving the sparse recovery problem at the heart of `ClusterPursuit` using `SubspacePursuit`.

We shall bound the complexity of each of these individually.

(1) Computing each d_i requires $d_i \leq d_{\max}$ additions. This is done n times to compute D, requiring $O(d_{\max}n)$ operations. As D is diagonal, the cost of computing $D^{-1}A$ is equal to the number of non-zero entries in A, which is bounded by $d_{\max}n$. By Corollaries 3.5 and 3.6, $d_{\max} \leq d^{\mathrm{in}}_{\max} + d^{\mathrm{out}}_{\max} = (1 + o(1))\omega\log(n_0) + 2b\log(n) + o(1) = O(\omega\log(n))$. Hence computing L and L^+ require $O(\omega\log(n)n)$ operations.

(2) Sorting the entries of a vector $\mathbf{v}$ in decreasing order, and then selecting the $(1+\epsilon)n_0$-largest of them, as in line 4 of Algorithm 2.2, takes at most $O(n\log(n)$ operations ([37]). Hence the computational cost of determining $\tilde{\Omega} = \tilde{\mathcal{L}}_{(1+\epsilon)\hat{n}_0}(\mathbf{v})$ is dominated by the cost of computing $\mathbf{v} := \left(L^+_{\Gamma^c}\right)^\top \mathbf{b}$. Each row of $\left(L^+_{\Gamma^c}\right)^\top$ contains at most $d_{\max}+1 \leq O(\omega\log(n))$ non-zero entries, hence this matrix-vector multiply requires at most $O(\omega\log(n)n)$ computations.

(3) The computational cost of solving the perturbed sparse recovery problem (2.5) using `SubspacePursuit` is equal to the number of iterations, m, times the cost of each iteration. The cost of each iteration is determined by calculating the cost of each step in the iterative part of `SubspacePursuit` (see Algorithm 2.1):

(3.1) Computing $\mathcal{L}_s(L^\top_\Omega \mathbf{r}^{k-1})$ is dominated by the cost of the matrix-vector multiply $L^\top_\Omega \mathbf{r}^{k-1}$. Each row of $L^\top_\Omega$ has at most $d_{\max}$ non-zero entries, hence the cost of this step is $O(\omega\log(n)n)$.

(3.2) Solving the least square problem in step (2) is the most computationally expensive step. We recommend using an iterative method, such as conjugate gradient (in our implementation we use MATLAB's backslash operation). Fortunately, as pointed out in [37], the matrix in question, $L_\Omega|_{\hat{T}^k} = L_{\hat{T}^k}$ is extremely well conditioned. This is because $|\hat{T}^k| = 2s$ and by assumption $\delta_{2s}(L) \leq \delta_{3s}(L)$. As in the proof of Theorem 3.15, we may assume that $\delta_{3s}(L) \leq 0.45$, for large enough n. By [37], specifically Proposition 3.1 and the discussion of §5, this implies that the condition number is small:

$$\kappa(L^\top_{\hat{T}^k} L^{\hat{T}_k}) := \frac{\lambda_{\max}(L^\top_{\hat{T}^k} L_{\hat{T}^k})}{\lambda_{\min}(L^\top_{\hat{T}^k} L_{\hat{T}^k})} \leq \frac{1+\delta_{2s}}{1-\delta_{2s}} \leq 2.64$$

The upshot of this is that it only requires a constant number of iterations of conjugate gradient to approximate the solution to the least-squares problem, $\mathbf{u}$, to within an acceptable tolerance. The cost of each iteration of conjugate gradient is equal to the cost of a matrix vector multiply by $L_{\hat{T}_k}$ or $L_{\hat{T}_k}^\top$, which is $O(\omega \log(n)n)$.

(3.3) The cost of sorting and thresholding (step (3)) is $\mathcal{O}(n \log(n))$.

(3.4) Finally the cost of computing the new residual $\mathbf{r}^k$ in step (4) is dominated by the matrix vector multiply $L_{T^k}^\top \mathbf{r}^k$, hence is $O(\omega \log(n)n)$.

Thus the cost of a single iteration of `SubspacePursuit` is $O(\omega \log(n)n)$. By the proof of Theorem 3.15, it suffices to take $m = O(\log(n))$, hence the cost of running `SubspacePursuit` is $O(\omega \log^2(n)n)$.

It follows that the computational cost of `SSCP` is dominated by the `SubspacePursuit` step, and is $O(\omega \log^2(n)n)$. If $\omega = O(\log(n))$, then $O(\omega \log^2(n)n) = O(\log^3(n)n)$. ∎

4. Experimental Results.

4.1. Implementation of algorithms.

All algorithms considered were run in MATLAB.

`SSCP`. The implementation of `SSCP` used is available as the function `SSCPMain`. We set the parameters $\epsilon = 0.2$, $R = 0.5$ and $s = 1.2\epsilon\hat{n}_0$. Unless otherwise indicated, $\hat{n}_0$ was set to be the true size of the cluster of interest.

`ESSC`. The algorithm we refer to as `ESSC` is technically the sub-routine referred to as `Community-Search` on pg. 1863 of [46] and as `Main.Search` in the R package for `ESSC` (available at http://jdwilson-statistics.com/publications/). We use a MATLAB implementation of this algorithm written by the second author. We compared the accuracy and run time of our MATLAB version to that of the R version, and found them to be nearly identical. We set the maximum number of iterations to 50 and the parameter $\alpha = 0.05$

`LOSP++`. We use the MATLAB implementation provided by the authors of [27], available at https://github.com/KunHe2015/LOSP. We use a diffusion parameter $\alpha = 0.1$ and the "light lazy" random walk. As for `SSCP`, $\hat{n}_0$ is set to be the true size of the cluster of interest, unless otherwise indicated.

`HKGrow`. We use the MATLAB implementation of this algorithm available at https://www.cs.purdue.edu/homes/dgleich/codes/hkgrow/. This implementation requires no input parameters.

The size of the seed set Γ given to `SSCP`, `LOSP++` and `HKGrow` is gn_0, where $g \in (0, 0.1)$ and n_0 is the true size of the cluster of interest. `ESSC` is seeded with the neighborhood of the highest degree vertex in the cluster of interest, as done in [46], unless otherwise indicated.

4.2. Measures of cluster quality.

When there exists a known, ground truth cluster C, we measure the accuracy of cluster extraction using the *Jaccard Index*: $\text{Jac}(C, C^\#) := \left|C \cap C^\#\right| / \left|C \cup C^\#\right|$. The maximum value of $\text{Jac}(C, C^\#)$ is 1, and this occurs when $C = C^\#$. The Jaccard index has a minimum value of 0, which is achieved when C and $C^\#$ are disjoint. We shall also have occasion to use conductance as a measure of cluster quality, as defined in §2.1. Note that lower values of conductance indicate better clusters.

4.3. The Synthetic Data sets.

We consider graphs drawn from three different stochastic block models. In all cases we take $g = 0.02$. In experiment 1, we consider graphs drawn from $\text{SBM}(\mathbf{n}, P_1)$, where $\mathbf{n} = (n_1, 10n_1)$ and in experiment 2 we draw graphs from $\text{SBM}(\mathbf{n}, P_2)$

where again $\mathbf{n} = (n_1, 10n_1)$. The connection probability matrices are:

$$P_1 = \begin{bmatrix} 3\log^2(n)/n & \log(n)/n \\ \log(n)/n & 3\log^2(n)/n \end{bmatrix} \text{ and } P_2 = \begin{bmatrix} 5\log^2(n)/n & \log(n)/(2n) \\ \log(n)/(2n) & \log(n)/(2n) \end{bmatrix}.$$

In experiment 3 we use the symmetric SBM, SSBM(n, k, p, q) for $k = 10$, $n = 10n_1$, $p = 3(\log(n))^2/n$ and $q = \log(n)/n$. See Figure 2 for a visualization of the adjacency matrices, rearranged so as to reveal the latent clusters. In all cases we focus on extracting the smaller cluster, C_1 (although in the third experiment all clusters are the same size). In all cases, we vary the size of C_1, namely n_1, from 100 to 600.

Figure 2: The adjacency matrices of typical graphs for each of the three benchmarks, permuted to reveal the ground truth clusters. From left to right: Experiments 1–3

	SSCP		HKGrow		LOSP++		ESSC	
	Jaccard	Time	Jaccard	Time	Jaccard	Time	Jaccard	Time
$n_1 = 100$	0.84	0.03	0.93	0.007	0.73	0.03	0.75	19.05
$n_1 = 200$	0.88	0.12	1.00	0.02	0.76	0.08	0.75	97.96
$n_1 = 300$	0.91	0.22	1.00	0.02	0.80	0.18	-	-
$n_1 = 400$	0.92	0.44	1.00	0.02	0.81	0.31	-	-
$n_1 = 500$	0.95	0.74	1	0.02	0.88	0.51	-	-

Table 1: Results of Experiment 1 - one small cluster and one large cluster. Note that ESSC did not finish running in a reasonable time for $n_1 \geq 300$.

	SSCP		HKGrow		LOSP++		ESSC	
	Jaccard	Time	Jaccard	Time	Jaccard	Time	Jaccard	Time
$n_1 = 200$	0.76	0.02	0.29	0.02	0.91	0.03	0.76	0.41
$n_1 = 300$	0.75	0.02	0.30	0.03	0.93	0.01	0.79	0.87
$n_1 = 400$	0.72	0.04	0.09	0.04	0.94	0.02	0.80	1.40
$n_1 = 500$	0.71	0.03	0.09	0.05	0.96	0.04	0.81	2.00
$n_1 = 600$	0.69	0.06	0.11	0.07	0.97	0.05	0.84	2.67

Table 2: Results of Experiment 2 - one cluster with many background vertices

	SSCP		HKGrow		LOSP++		ESSC	
	Jaccard	Time	Jaccard	Time	Jaccard	Time	Jaccard	Time
$n_1 = 100$	0.73	0.01	0.34	0.02	0.66	0.03	0.79	0.32
$n_1 = 200$	0.85	0.04	0.84	0.01	0.78	0.01	0.70	1.21
$n_1 = 300$	0.88	0.08	1	0.02	0.81	0.05	0.80	2.34
$n_1 = 400$	0.92	0.22	1	0.03	0.84	0.1	0.99	2.49
$n_1 = 500$	0.94	0.34	1	0.03	0.87	0.13	0.94	6.6

Table 3: Results of Experiment 3 - ten identical clusters

Remark 4.1. The precise values of the coefficients of $\log^2(n)/n$ and $\log(n)/n$ in all experiments are essentially arbitrary, and varying them does not qualitatively effect our results. The interested reader is invited to investigate further—all benchmarking scripts used are contained in the SSCP package.

4.4. The Real Data Sets. The `facebook100` dataset consists of anonymized Facebook "friendship" networks at 100 American universities, and was first introduced and studied in [45]. It contains, for each college or university, a graph whose vertices correspond to undergraduates with a Facebook account at that institution. Edges connect students who were friends on Facebook the day (in September 2005) the data was collected. Certain demographic markers (year of entry, gender, residence, high school etc.) were also collected in an anonymized format. We focus on four schools, California Institute of Technology (Caltech), Rice, University of California, Santa Cruz (UCSC) and Smith College, identified by Traud *et. al.* ([45]) as being most strongly clustered by residence. We treat the residence assignments as the ground truth clusters. We note that there are always some students whose residential affiliation is unknown; we treat these as background vertices. For each cluster, we run each algorithm ten times, each time with a different set of uniformly randomly selected seed vertices. For `SSSCP`, `HKGrow` and `LOSP++` the seed set consists of g(size of cluster) vertices, where $g = 0.05$ for Smith and Caltech while $g = 0.02$ for Rice and UCSC. For `ESSC`, the seed set is the neighborhood of a certain vertex in the ground truth cluster. We tried taking this vertex to be the highest degree vertex in the cluster (as in [46]) as well as selecting this vertex uniformly at random. Experimentally, we observed better results for the latter, so we report these. We note that for the larger networks (*i.e.* Smith, Rice and UCSC) `ESSC` did not converge within a reasonable amount of time. The results reported in Table 5 are averaged over all clusters, and over all ten independent trials for each cluster.

	Vertices	Clusters	Max cluster size	Min cluster size	Mean cluster size
Caltech	769	8	99	44	74.63
Smith	2970	36	113	12	70.17
Rice	4087	9	414	382	396
UCSC	8991	10	925	622	773.7

Table 4: Basic properties of the four social networks studied.

	SSCP		HKGrow		LOSP++		ESSC	
	Jaccard	Time	Jaccard	Time	Jaccard	Time	Jaccard	Time
Caltech	0.43	0.01	0.27	0.004	0.38	0.01	0.43	3.72
Smith	0.33	0.02	0.06	0.02	0.31	0.04	-	-
Rice	0.39	0.14	0.43	0.03	0.42	0.10	-	-
UCSC	0.28	0.35	0.16	0.04	0.28	0.31	-	-

Table 5: Results for four social networks from the `facebook100` data set. Quantities displayed are averaged over ten independent trials per cluster and over all clusters.

The `polblogs` data set This data set consists of 1224 political blogs collected in the leadup to the 2004 U.S presidential election by Adamic and Glance [3]. Vertices are connected if there is a hyperlink between them. The political leanings of the blogs — liberal vs. conservative — were recorded, and it was shown in [3] that partitioning the vertices into two clusters, C^{lib} and C^{cons} based on political leaning gives a good clustering. However as noted by several authors, e.g. [39] and [46], the structure of this network is actually a bit more complicated. For example, Olhede and Wolfe [39] suggest that the community structure of this network can more accurately be described by 17 smaller communities of approximately 70 vertices each. In this experiment, we investigate the ability of `SSCP` to find clusters at different scales. We seed `SSCP`, `LOSP++` and `HKGrow` with ten vertices. We attempted to run `ESSC` seeded, as in the other experiments, with the neighbourhood of a vertex but did not observe good results.[1] However, when we gave `ESSC` the same set of seed vertices as the other algorithms we observed much better performance, and so it is these results we report. For `SSCP` and `LOSP++`, we try two different scale parameters: $\hat{n_0}$ equal to the true size of the liberal cluster, and also $\hat{n_0} = 80$, based on the suggestion of Olhede and Wolfe [39] mentioned earlier. For both values of $\hat{n}_0$, we conduct ten independent trials. In each trial the seed set Γ is drawn uniformly at random from the set of liberal vertices with high degree (that is, degree greater than 10) The results are recorded in Table 6. We repeat this process for the conservative vertices. Note that when $\hat{n_0} = 80$, no ground truth is available so we use conductance as our measure of cluster quality. `ESSC` took approximately 9 seconds for each run, `SSCP` took approximately 0.08 seconds, while `HKGrow` and `LOSP++` took approximately 0.02 seconds.

	SSCP			HKGrow			LOSP++			ESSC		
	Cond.	$\bar{n_0}$	σ	Cond.	$\bar{n_0}$	σ	Cond.	$\bar{n_0}$	σ	Cond.	$\bar{n_0}$	σ
Lib. large	0.31	571.7	9.25	0.14	482.8	72.9	0.17	588	-	0.09	495.2	11.41
Lib. small	2.77	72	0	-	-	-	1.52	75	-	-	-	-
Cons. large	0.19	612	6.32	0.09	639.9	27.46	0.18	636	-	0.09	601	13.95
Cons. small	3.39	72	0	-	-	-	2.13	75	-	-	-	-

Table 6: Results for the `polblogs` data set. $\bar{n_0}$ (resp. σ) denotes the mean of (resp. standard deviation in) the sizes of clusters found.

The MNIST Data set This data set, available at http://yann.lecun.com/exdb/mnist/,

[1] The failure of `ESSC` here is easily explainable. Recall that `ESSC` is designed to extract *significant* communities, and is not forced to return a community containing the seed set. For this data set, `ESSC` gravitated towards the cluster of conservative vertices, even when seeded with the neighborhood of a liberal vertex

consists of 60,000 training and 10,000 test images of handwritten digits. We do not consider the full data set, but rather sample 20,000 images at random from the training set. We do this so that all three algorithms run in a reasonable amount of time. We perform an elementary preprocessing step, which we now describe. After performing PCA on the set of images, we retain only the 50 leading principal components to obtain a feature vector $\mathbf{x}_i$ for each image. We then form an affinity matrix A using the local scaling of Zelnik-Manor and Perona [47]. Specifically:

$$\tilde{A}_{ij} = \begin{cases} \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma_i \sigma_j}\right) & \text{if } \mathbf{x}_j \text{ is one of } \mathbf{x}_i\text{'s } K \text{ nearest neighbors} \\ 0 & \text{otherwise} \end{cases}$$

where σ_i is a local scaling parameter: $\sigma_i = \|\mathbf{x}_i - \mathbf{x}_{[r,i]}\|_2$ and $\mathbf{x}_{[r,i]}$ denotes the r-th nearest point to $\mathbf{x}_i$.

	SSCP		HKGrow		LOSP++	
	Jaccard	Time	Jaccard	Time	Jaccard	Time
$g = 0.01$	0.80	3.11	0.63	0.05	0.67	0.93
$g = 0.02$	0.84	3.65	0.65	0.05	0.66	1.61
$g = 0.05$	0.90	3.65	0.75	0.06	0.75	3.48

Table 7: Results for the MNIST data set, averaged over ten independent trials per digit and over all ten digits. The size of the seed sets is always $g \times$ (size of cluster).

Following [29], we set $K = 15$ and $r = 7$. Finally, as the matrix $\tilde{A}$ is not symmetric, we symmetrize by defining $A = \tilde{A}^\top \tilde{A}$, which we interpret as a weighted adjacency matrix. As there are 10 digits, there are naturally 10 clusters in the graph defined by A. For each digit, we run 10 trials of `SSCP`, `HKGrow` and `LOSP++` seeded with g(size of cluster) images selected uniformly at random from the cluster, for $g = 0.01, 0.02$ and 0.05. We do not test `ESSC` as it is not designed to handle weighted graphs[2]. `SSCP` and `LOSP++` are given the exact cluster size as $\hat{n}_0$. We present the Jaccard indices and run times, averaged over the ten independent trials and over all ten clusters, in Table 7.

Semi-Supervised classification of the MNIST dataset The problem of separating a data set into a predefined number of classes, given a small subset of labeled data (*i.e.* data points whose class memberships are known) is known in the machine learning literature as *semi-supervised learning*, and is a problem of growing interest. Here, we demonstrate that `SSCP` can be used as the core of an effective and efficient semi-supervised classifier. We implement an iterated version of `SSCP` (available in the SSCP package as `ISSCP2`) described in pseudocode as Algorithm 4.1. As before, k will denote the number of classes/clusters. Let $\Gamma_a \subset C_a$ denote the labeled data in the a-th class. `ISSCP2` takes as input an adjacency matrix A, which we compute using the same preprocessing step as the previous MNIST experiment, the labeled data $\{\Gamma_1, \ldots, \Gamma_k\}$ and estimates $\{\hat{n}_1, \ldots, \hat{n}_k\}$ of the sizes of $C_1, \ldots, C_k$. For the

[2] Recently, the authors learned of the thesis of Palowitch [40], which extends the `ESSC` framework to weighted networks. However, it requires one to specify a null-model of graph resembling the data, except without any clusters, and it is not always clear how to do so in practice

call to SSCP when finding the a-th cluster, we fix the parameters as $\epsilon = 0.2$, $R = 0.4$ and $s = 1.2\epsilon\hat{n}_a$. We experiment with setting $\hat{n}_a$ to be the true size of C_a and $\hat{n}_a = n/k$ for all a, and observe that it affects the classification accuracy only slightly.

Note that SSCP, as an extractive algorithm is *a priori* at an innate disadvantage for a multi-class classification problem, because when it is finding the a-th cluster, it only "sees" the labeled data Γ_a. We remedy this by running a subroutine we call HeavyEdges prior to extracting any of the clusters. This function re-weights edges between labeled vertices as follows. For $i, j \in \Gamma_a$, set $w(i, j) = 5$. For $i \in \Gamma_a$ and $j \in \Gamma_b$ with $a \neq b$, set $w(i, j) = 0$. Leave all other edges unaltered. Empirically we found that including HeavyEdges boosts the classification accuracy of ISSCP2 by about 1%.

Algorithm 4.1 Iterated Semi-Supervised Cluster Pursuit (ISSCP2)

Input: Adjacency matrix A, $\Gamma_a \subset C_a$ and $\hat{n}_a \approx |C_a|$ for $a = 1, \dots k$.
Step 1 $A = \texttt{HeavyEdges}(A)$. Set $G^{(1)} = G$ and $A^{(1)} = A$.
Step 2
for $a = 1 : k - 1$ **do**
  $C_a^{\#} = \texttt{SSCP}(A^{(a)}, \epsilon = 0.2, R = 0.4, \Gamma_a, \hat{n}_0 = \hat{n}_a, s = 1.2\epsilon\hat{n}_a)$.
  Let $G^{(a)}$ be the induced subgraph on $V^{(a-1)} \setminus C_a^{\#}$ with the adjacency matrix $A^{(a)}$.
end for
Step 3 Let $\Omega_k = V \setminus \bigcup_{a=1}^{k-1} C_a^{\#}$. Find $C_k^{\#}$ as $C_k^{\#} = \texttt{ClusterPursuit}(A, R = 0.4, \Omega_k, s = 1.2\epsilon\hat{n}_k)$
Step 4 (Optional) Define $V^{\text{background}} = V \setminus \bigcup_{a=1}^{k} C_a^{\#}$.
for $s = 1 : \left|V^{\text{background}}\right|$ **do**
  For vertex $i_s \in V^{\text{background}}$ let $\tilde{a} = \text{argmax}_{a=1}^{k} \sum_{j \in C_a} A_{i_s j}$
  Let $C_{\tilde{a}}^{\#} = C_{\tilde{a}}^{\#} \cup \{i_s\}$
end for
Output: $\{C_1^{\#}, \dots, C_k^{\#}\}$

In Table 8 we report the classification accuracy of ISSCP2, run using the optional fourth step, applied to the entire MNIST data set (test + training, so 70,000 images). In Table 9 we also detail the accuracy of other semi-supervised learning algorithms on the same data set.

Remark 4.2. We note that ISSCP2 will have an advantage over the other methods listed in Table 9 in the following scenario. Suppose instead of a clean data set like MNIST, one is trying to use semi-supervised classification on a data set containing data points which are corrupted beyond classifiability, or data points which do not fit into any of the classes (e.g. if several hundred pictures of handwritten letters were accidentally included into the MNIST data set). ISSCP2, run without Step 4, is not forced to assign a class to these outliers. Instead, it will just declare them to be background vertices in the graph. This is in contrast with all the other methods listed, which are forced to assign a class to every data point.

	$\hat{n}_0$ = exact sizes	$\hat{n}_0 = n/k$
$g = 0.01$	96.82%	95.53%
$g = 0.02$	97.42%	96.73%
$g = 0.03$	97.56%	96.79%
$g = 0.04$	97.58%	96.92%
$g = 0.05$	97.64%	97.06%

Table 8: Accuracy of classification of MNIST data using `ISSCP2` when given exact and approximate cluster sizes.

Method	Labelled	Accuracy
TSVM [17]	1000	95.62%
Deep Generative Model [30]	1000	97.13%
`ISSCP2`	1000	97.15%
Auction Dynamics [29]	700	97.43%
Ladder Networks [41]	1000	99.16%

Table 9: Comparing `ISSCP2` to other, state-of-the-art, semi-supervised methods on MNIST.

4.5. The effect of the parameter $\hat{n}_0$. In this section we test how accurate `SSCP` is when $\hat{n}_0$ differs significantly from the true cluster size, $|C|$. We rerun `SSCP` on graphs generated using the same SBM parameters as Experiments 1–3, (denoted as "two clusters", "cluster + background" and "ten clusters" respectively in Table 10, with inputs $\epsilon = 0.2$, $R = 0.5$ and $s = 1.2\epsilon\hat{n}_0$. In each case, the true size of the cluster of interest, n_1, is set to be 400. We then vary $\hat{n}_0$ from 300 to 500. In Table 10, we present the Jaccard index and the conductance, averaged over ten independent trials, for each of these experiments. Recall that high Jaccard index indicates a good cluster, while low conductance indicates a good cluster. Note that while the Jaccard index is calculated with respect to the ground truth, conductance only takes into account the vertices in the cluster found and the network topology. Thus, Table 10 suggests a data driven approach to finding the optimal cluster size — simply vary n_0, record the conductance, and look for a local minimum. We emphasize that unlike `LOSP++`, `SSCP` is not forced to output a cluster of size precisely $\hat{n}_0$.

	$\hat{n}_0 = 300$		$\hat{n}_0 = 350$		$\hat{n}_0 = 400$		$\hat{n}_0 = 450$		$\hat{n}_0 = 500$	
	Jac.	Cond.	Jac.	Cond.	Jac.	Cond.	Jac.	Cond.	Jac.	Cond.
Two Clusters	0.71	0.77	0.84	0.58	0.86	0.55	0.80	0.61	0.76	0.65
Cluster + background	0.44	0.92	0.52	0.63	0.72	0.33	0.81	0.20	0.75	0.19
Ten Clusters	0.72	0.71	0.86	0.51	0.93	0.43	0.87	0.50	0.80	0.58

Table 10: Using `SSCP`, with $s = 1.2\epsilon\hat{n}_0$ to find C_1 of size $n_1 = 400$. In the 'Two clusters' and 'Ten clusters' cases, there is a clear minimum of conductance when $\hat{n}_0 = n_1$.

4.6. Discussion. Over both synthetic and real data sets, the performance of `SSCP` is remarkably consistent, in both run-time and accuracy. Whereas `HKGrow` and `ESSC` both have types of graph for which they perform poorly (The 'one small and one large cluster' graph of Experiment 1 for `ESSC`, and the 'one cluster plus background' graph of Experiment 2 for `HKGrow`), the accuracy of `SSCP` is never the worst, and is frequently the best. Moreover, unlike `ESSC`, the run-time of `SSCP` depends only on the size of the graph, not its topology. Although the performance of `LOSP++` in extracting small clusters from the `polblogs` data set is slightly better, `SSCP` handles this challenge well, demonstrating that it is capable of extracting clusters at different scales from heterogeneous networks. Finally, the accuracy of `SSCP` on weighted graphs, e.g. the MNIST data set, is markedly better than that of the other algorithms tested.

A. RIP for Graph Laplacians. In this section we prove Theorem 3.13 and Lemma 3.14. We proceed via a series of lemmas. We first show that the RIP holds for L when $G \sim \mathrm{ER}(n_0, p)$. We then show that it still holds when G is a disjoint union of Erdős - Rènyi graphs, equivalently when $G \sim \mathrm{SSBM}(n, k, p, 0)$. Finally, we extend to the case where $G \sim \mathrm{SSBM}(n, k, p, q)$ via a perturbation argument.

Lemma A.1. *Let G be any connected graph on n_0 vertices, and let $t < n_0$. Then:*

$$\delta_t(L) \leq \max\{1 - \lambda_2^2 \left(\frac{d_{\min}}{d_{\max}} - \frac{d_{\max}}{d_{\min}} \frac{t}{n_0} \right), 1 - \lambda_{\max}^2\}.$$

Proof. Recall that the t-Restricted Isometry Constant $\delta_t(L)$ is the smallest δ such that, for any $\mathbf{v}$ with $|\mathrm{supp}(\mathbf{v})| \leq t$ and $\|\mathbf{v}\|_2 = 1$:

$$(1 - \delta) \leq \|L\mathbf{v}\|_2^2 \leq (1 + \delta).$$

We shall prove the theorem by showing that, for any such $\mathbf{v}$, $\|L\mathbf{v}\|_2 \leq \lambda_{\max}$ and $\|L\mathbf{v}\|_2 \geq \lambda_2^2 \left(\frac{d_{\min}}{d_{\max}} - \frac{d_{\max}}{d_{\min}} \frac{t}{n_0} \right)$. The first bound is straightforward:

$$\|L\mathbf{v}\|_2 \leq \|L\|_2 \|\mathbf{v}\|_2 = \lambda_{\max}(1) = \lambda_{\max}$$

The second bound requires some work. Recall that $L = I - D^{-1}A$. This matrix is not symmetric, but $L^{\mathrm{sym}} = I - D^{-1/2}AD^{-1/2}$ is. Moreover, $L^{\mathrm{sym}} = D^{1/2}LD^{-1/2}$, and so L and L^{sym} have the same eigenvalues. Let $\mathbf{w}_1, \ldots, \mathbf{w}_{n_0}$ be an orthonormal eigenbasis for L^{sym}. These eigenvectors are well studied (see, for example, [13]) and in particular $\mathbf{w}_1 = \frac{1}{\sqrt{\mathrm{vol}(G)}} D^{1/2}\mathbf{1}$ where $\mathbf{1}$ is the all-ones vector, and $\mathrm{vol}(G) = \sum_{i \in V} d_i$. Observe that:

$$L\mathbf{v} = D^{-1/2} \left(D^{1/2} L D^{-1/2} \right) D^{1/2}\mathbf{v} = D^{-1/2} L^{\mathrm{sym}} D^{1/2}\mathbf{v} = D^{-1/2} L^{\mathrm{sym}}\mathbf{z},$$

where $\mathbf{z} := D^{1/2}\mathbf{v}$. It follows that:

$$\|L\mathbf{v}\|_2 = \|D^{-1/2} L^{\mathrm{sym}}\mathbf{z}\|_2 \geq \frac{1}{\sqrt{d_{\max}}} \|L^{\mathrm{sym}}\mathbf{z}\|_2. \tag{A.1}$$

Express $\mathbf{z}$ in terms of the orthonormal basis $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$, namely $\mathbf{z} = \sum_{i=1}^{n_0} \alpha_i \mathbf{w}_i$. Then:

$$\|L^{\mathrm{sym}}\mathbf{z}\|_2^2 = \|\sum_{i=1}^{n_0} \alpha_i \lambda_i \mathbf{w}_i\|_2^2 = \|\sum_{i=2}^{n_0} \alpha_i \lambda_i \mathbf{w}_i\|_2^2 \geq \lambda_2^2 \left(\sum_{i=2}^{n_0} \alpha_i^2 \right)$$

and $\sum_{i=2}^{n_0} \alpha_i^2 = \|\mathbf{z}\|_2^2 - \alpha_1^2$. We now bound $\|\mathbf{z}\|_2$ and α_1.

$$\|\mathbf{z}\|_2^2 = \|D^{1/2}\mathbf{v}\|_2^2 \geq \left(\sqrt{d_{\min}} \right)^2 \|\mathbf{v}\|_2^2 = d_{\min}$$

while:

$$\alpha_1 = \langle \mathbf{z}, \mathbf{w}_1 \rangle = \langle D^{1/2}\mathbf{v}, \frac{1}{\sqrt{\mathrm{vol}(G)}} D^{1/2}\mathbf{1} \rangle = \frac{1}{\sqrt{\mathrm{vol}(G)}} \langle \mathbf{v}, D\mathbf{1} \rangle \leq \frac{d_{\max}}{\sqrt{\mathrm{vol}(G)}} \langle v, \mathbf{1} \rangle.$$

We now use the assumptions on $\mathbf{v}$. Specifically $\langle v, \mathbf{1} \rangle \leq \|\mathbf{v}\|_1 \leq \sqrt{t}\|\mathbf{v}\|_2 = \sqrt{t}$ and so

$$\alpha_1 \leq d_{\max} \frac{\sqrt{t}}{\sqrt{\mathrm{vol}(G)}} \leq d_{\max} \frac{\sqrt{t}}{\sqrt{d_{\min} n_0}} = \frac{d_{\max}}{\sqrt{d_{\min}}} \frac{\sqrt{t}}{\sqrt{n_0}}.$$

Returning to equation (A.1):

$$\|L\mathbf{v}\|_2^2 \geq \frac{1}{d_{\max}} \|L^{\mathrm{sym}}\mathbf{z}\|_2^2 \geq \frac{1}{d_{\max}} \lambda_2^2 \left(d_{\min} - \frac{d_{\max}^2}{d_{\min}} \frac{t}{n_0} \right) = \lambda_2^2 \left(\frac{d_{\min}}{d_{\max}} - \frac{d_{\max}}{d_{\min}} \frac{t}{n_0} \right).$$

Lemma A.2. *Suppose that $G \sim ER(n_0, p)$ with $p = \omega \ln(n_0)/n_0$ for some $\omega \to \infty$. Then $\delta_t(L) \leq t/n_0 + o(1)$ a.s.*

Proof. This is a simple consequence of Lemma A.1. If G is as in the hypothesis then $d_{\min} = (1 - o(1))\, n_0 p$ and $d_{\max} = (1 + o(1))\, n_0 p$ a.s. by Theorem 3.3. Moreover $\lambda_2 \geq 1 - o(1)$ and $\lambda_{n_0} \leq 1 + o(1)$ a.s. by Theorem 3.4. Hence:

$$\begin{aligned}
\lambda_2^2 \left(\frac{d_{\min}}{d_{\max}} - \frac{d_{\max}}{d_{\min}} \frac{t}{n_0} \right) &\geq (1 - o(1))^2 \left(\frac{(1 - o(1))\, n_0 p}{(1 + o(1))\, n_0 p} - \frac{(1 + o(1))\, n_0 p}{(1 - o(1))\, n_0 p} \frac{t}{n_0} \right) \\
&\geq (1 - o(1)) \left(\frac{1 - o(1)}{1 + o(1)} - \frac{1 + o(1)}{1 - o(1)} \frac{t}{n_0} \right) \\
&= (1 - o(1)) \left(1 - o(1) - (1 + o(1)) \frac{t}{n_0} \right) \\
&= 1 \left(1 - \frac{t}{n_0} - o(1) \right) - o(1) = 1 - \frac{t}{n_0} - o(1) \quad \text{a.s.}
\end{aligned}$$

Hence by Lemma A.1, we have that

$$\delta_t(L) \leq \max \left\{ 1 - \left(1 - \frac{t}{n_0} - o(1) \right), o(1) \right\} = \frac{t}{n_0} + o(1) \quad \text{a.s.}$$

Lemma A.3. *Suppose that $G \sim SSBM(n, k, p, 0)$ with $k = O(1)$, then $\delta_t(L) \leq \frac{t}{n_0} + o(1)$ a.s.*

Proof. Because $q = 0$, there are no inter-cluster edges, and G is a disjoint union of subgraphs $G_1, \ldots, G_k$, each drawn independently from $\mathrm{ER}(n_0, p)$. It follows that L is block diagonal, with blocks $L_1, \ldots, L_k$, where L_a is the Laplacian of G_a. For a block diagonal matrix, one can easily check that $\delta_t(L) = \max_a \delta_t(L_a)$. By Lemma A.2, $\delta_t(L_a) \leq t/n_0 + o(1)$ a.s. As $k = O(1)$, by the union bound, $\max_a \delta_t(L_a) \leq \frac{t}{n_0} + o(1)$ a.s.

We shall finish the argument by appealing to the following theorem of Herman and Strohmer. Recall that for any matrix B, $\|B\|_{2,t} := \max\{\|B_T\|_2 : T \subset [n] \text{ and } |T| = t\}$

Theorem A.4 ([28]). *Suppose that $\Phi = \hat{\Phi} + M$. Let $\hat{\delta}_t$ and δ_t denote the t restricted isometry constants of $\hat{\Phi}$ and Φ respectively and recall that $\epsilon_\Phi^t := \|M\|_{2,t}/\|\hat{\Phi}\|_{2,t}$. Then:*

$$\delta_t \leq (1+\hat{\delta}_t)\left(1+\epsilon_\Phi^t\right)^2 - 1.$$

Proof. (*of Theorem 3.13*) Recall that, if $G \sim \mathrm{SSBM}(n,k,p,q)$ and L denotes the Laplacian of G, then we may write $L = L^{\text{in}} + M$ where L^{in} is the Laplacian of the in-cluster subgraph $G^{\text{in}} \sim \mathrm{SSBM}(n,k,p,0)$ and $\|M\|_2 \leq o(1)$ by Theorem 3.10. By Lemma A.3 $\hat{\delta}_t := \delta_t(L^{\text{in}}) \leq t/n_0 + o(1)$ a.s. Observe that, for any matrix B,

$$\|B\|_{2,t} := \max_{\substack{T\subset[n]\\|T|=t}} \|B_T\|_2 = \max_{\substack{T\subset[n]\\|T|=t}} \sigma_{\max}(B_T),$$

where $\sigma_{\max}(B_T)$ denotes the maximum singular value of B_T. By the interlacing theorem for singular values ([44]), $\lambda_{t-1}(B) \leq \sigma_{\max}(B_T) \leq \lambda_t(B) \leq \lambda_{\max}(B)$ and so $\|M\|_{2,t} \leq \|M\|_2 \leq o(1)$ a.s. by Theorem 3.10. Similarly, $\|L^{\text{in}}\|_{2,t} \geq \lambda_{t-1}(L^{\text{in}})$. The eigenvalues of L^{in} are the eigenvalues of the L_a, counted with multiplicity. In particular, as long as $t > k+1$ [3], $\lambda_{t-1}(L^{\text{in}}) \geq \min_a \lambda_2(L_a)$. By theorem 3.4, $\lambda_2(L_a) \geq 1 - o(1)$ a.s., and as $k = O(1)$, we may apply the union bound to obtain $\|L^{\text{in}}\|_{2,t} \geq 1 - o(1)$ a.s. Hence:

$$\epsilon_\Phi^t := \frac{\|M\|_{2,t}}{\|L^{\text{in}}\|_{2,t}} \leq \frac{o(1)}{1-o(1)} = o(1) \text{ a.s.} \tag{A.2}$$

Applying theorem A.4:

$$\begin{aligned}\delta_t(L) &\leq \left(1+\frac{t}{n_0}+o(1)\right)(1+o(1))^2 - 1 = \left(1+\frac{t}{n_0}+o(1)\right)(1+o(1)) - 1\\ &= \left(1+\frac{t}{n_0}+o(1)\right) - 1 = \frac{t}{n_0} + o(1) = \gamma + o(1) \text{ if } t = \gamma n_0\end{aligned}$$

■

Proof. (*Of lemma 3.14*) That $\epsilon_\Phi^t = o(1)$ was shown in the proof of Theorem 3.13 (see equation (A.2)). Here, $\epsilon_{\mathbf{y}} := \|\mathbf{e}\|_2/\|\mathbf{y}^{\text{in}}\|_2$ where $\|\mathbf{e}\|_2 = o(n_0)$ by Lemma 3.11 and 3.12. Rearranging equation (3.5) we get that $\mathbf{y}^{\text{in}} = L^{\text{in}}\mathbf{1}_\Omega = L^{\text{in}}(\mathbf{1}_U - \mathbf{1}_W)$ where $U := C_1 \setminus (\Omega \cap C_1)$ and $W := \Omega \setminus (\Omega \cap C_1)$. As in the proof of theorem 3.15, $|U| = o(n_0)$ and $|W| = \epsilon n_0 + o(n_0)$, hence $\|\mathbf{1}_U - \mathbf{1}_W\|_0 = o(n_0) + \epsilon n_0 + o(n_0) \leq 2\epsilon n_0$ for n_0 large enough. It follows that:

$$\|\mathbf{y}^{\text{in}}\|_2^2 = \|L^{\text{in}}(\mathbf{1}_U - \mathbf{1}_W)\|_2^2 \geq (1-\delta_{2\epsilon n_0})\|\mathbf{1}_U - \mathbf{1}_W\|_2^2 \geq (2\epsilon + o(1))(\epsilon n_0 + o(n_0))$$

Where the bound on $\delta_{2\epsilon n_0} = \delta_{2\epsilon n_0}(L^{\text{in}})$ comes from Lemma A.3. As ϵ is fixed, we obtain

$$\epsilon_{\mathbf{y}} = \frac{o(n_0)}{2\epsilon^2 n_0 + o(n_0)} = o(1)$$

Note that the sparsity input for `SubspacePursuit`, namely s, is set equal to ϵn_0. As $\epsilon < 0.15$ by assumption, it follows that $\delta_{3\epsilon n_0} < 0.45 + o(1)$. For n large enough, we may assume that $\delta_{3\epsilon n_0} \leq 0.45$. It follows from direct calculation that $\rho \leq 0.8751$ and $\tau \leq 55.8490$. ■

[3] In the set up we are considering, t is proportional to n, while k is fixed, thus this will always be the case for n large enough

REFERENCES

[1] E. Abbe, Community detection and stochastic block models: recent developments, *arXiv preprint arXiv:1703.10146* (2017).

[2] E. Abbe and C. Sandon, Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. *Proceedings of Foundations of Computer Science (FOCS) 2015*, pp. 670–688 (2015).

[3] L. A. Adamic and N. Glance, The political blogosphere and the 2004 U.S. election: divided they blog. *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43 (2005).

[4] R. Andersen, F.R.K. Chung, and K. Lang, Local graph partitioning using pagerank vectors, *Proceedings of Foundations of Computer Science (FOCS) 2006*, pp. 475–486 (2006).

[5] A.-L. Barabási and A. Réka, Emergence of scaling in random networks, *Science*, Vol. 286 no. 5439, pp. 509–512 (1999).

[6] A. Beck and M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM J. Image Sciences*, Vol. 2, pp. 183–202 (2009).

[7] T. Blumensath and M. E. Davies, Iterative hard thresholding for compressed sensing, *Applied and Computational Harmonic Analysis.*, Vol. 27, pp.265–274 (2009).

[8] B. Bollobás, Vertices of given degree in a random graph, *Journal of Graph Theory*, Vol 6 no. 2, pp. 147–155 (1982).

[9] B. Bollobás, *Random Graphs*, Cambridge University Press (2001).

[10] E. J. Candès, M. B. Wakin, and S. Boyd, Enhancing sparsity by re-weighted ℓ_1 minimization, *Journal of Fourier Analysis and Applications*, Vol. 14, pp. 877–905 (2008).

[11] E. J. Candés, J. K. Romberg, and T. Tao, Stable signal recovery from incomplete and inaccurate measurements, *Communications on Pure and Applied Mathematics*, Vol. 59 issue 8, pp. 1207–1223 (2006).

[12] E. J. Candés and T. Tao, Decoding by linear programming, *IEEE Transactions on Information Theory*, Vol 51, pp. 4203–4215 (2005).

[13] F. R. K. Chung, *Spectral graph theory*, No. 92, American Mathematical Soc. (1997).

[14] F. R. K. Chung and R. Graham, Sparse quasi-random graphs, *Combinatorica*, Vol. 22 no. 2, pp. 217–244 (2002).

[15] F. R. K. Chung, A local graph partitioning algorithm using heat kernel pagerank, *Internet Mathematics*, Vol. 6 no. 3, pp 315–330 (2009).

[16] F. R. K. Chung and M. Radcliffe, On the spectra of general random graphs, *The Electronic Journal of Combinatorics*, Vol. 18 no. 1 (2011).

[17] R. Collobert, F. Sinz, J. Weston and L. Bottou, Large scale transductive SVMs *Journal of Machine Learning Research*, Vol. 7 (2006).

[18] W. Dai and O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, *IEEE Transactions on Information Theory*, Vol. 55 issue 5, pp. 2230–2249 (2009).

[19] I. S. Dhillon, Co-clustering documents and words using spectral graph partitioning, *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 269–274 (2001).

[20] D. L. Donoho and M. Elad, Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization, *Proceedings of the National Academy of Sciences*, Vol. 100 issue 5, pp. 2197–2202 (2003).

[21] S. Doyle, S. Agner and A. Madabhushi, Automated grading of breast cancer histopathology using spectral clustering with textural and architectural image features, *Proceedings of 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 496–499 (2008).

[22] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, Birkhäus Verlag (2013).

[23] S. Foucart and M-J. Lai, Sparsest solutions of underdetermined linear Systems via ℓ_q-minimization for $0 \leq q \leq 1$, *Applied and Computational Harmonic Analysis*, Vol. 26 issue 3, pp. 395–407 (2009).

[24] A. Frieze and M. Karoński, *Introduction to Random Graphs*, Cambridge University Press (2015).

[25] M. Girvan and M.E.J. Newman, Community structure in social and biological networks, *Proceedings of the national academy of sciences*, Vol. 99 no. 12, pp. 7821–7826 (2002).

[26] K. He, P. Shi, J.E. Hopcroft, D. Bindel and Y. Li, Detecting Overlapping Communities from Local Spectral Subspaces, *International Conference on Data Mining (ICDM) 2015* pp. 769–774 (2015).

[27] K. He, P. Shi, J.E. Hopcroft and D. Bindel, Local spectral diffusion for robust community detection, *Twelfth Workshop on Mining and Learning with Graphs* (2016).
[28] M. A. Herman and T. Strohmer, General deviants: An analysis of perturbations in compressed sensing, *IEEE Journal on Selected Topics in Signal Processing*, Vol. 4 issue 2, pp. 342–349 (2010).
[29] M. Jacobs, E. Merkurjev and S. Esedoglu, Auction dynamics: A volume constrained MBO scheme, *Journal of Computational Physics*, Vol. 354, pp. 288–310 (2018).
[30] D. P. Kingma, S. Mohamed, D. J. Rezende and M. Welling, Semi-supervised learning with deep generative models, *Advances in Neural Information Processing Systems* (2014).
[31] K. Kloster and D. Gleich, Heat kernel based community detection, *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1386–1395 (2014).
[32] A. Lancichinetti, S. Fortunato and F. Radicchi, Benchmark graphs for testing community detection algorithms, *Physical Review E*, Vol 78 no. 4 (2008).
[33] A. Lewis, N. Jones, M. Porter and C. Deane, The function of communities in protein interaction networks at multiple scales, *BMC systems biology*, Vol. 4, no. 1 (2010).
[34] Y. Li, K. He, D. Bindel and J. Hopcroft, Uncovering the small community structure in large networks: A local spectral approach, *Proceedings of the 24th international conference on world wide web* pp. 658-668 (2015).
[35] H. Li, Improved analysis of SP and CoSaMP under total perturbations, *EURASIP Journal on Advances in Signal Processing 2016*, no. 1, (2016).
[36] U. von Luxburg, A tutorial on spectral clustering, *Statistics and Computing*, Vol. 17 issue 4, pp. 395–416 (2007).
[37] D. Needell and J. A. Tropp, CoSaMP: Iterative signal recovery from incomplete and inaccurate samples, *Applied and Computational Harmonic Analysis*, Vol. 26 issue 3, pp. 301-321 (2009).
[38] M. Mahoney, L. Orecchia and N. Vishnoi, A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally, *Journal of Machine Learning Research*, Vol. 13 August, pp. 2339–2365 (2012).
[39] S. C. Olhede and P. J. Wolfe, Network histograms and universality of block model approximation, *arXiv preprint arXiv:312-5306v3* (2014).
[40] J. J. Palowitch, Testing-Based community detection for complex networks, *Thesis, University of North Carolina at Chapel Hill* (2017).
[41] A. Rasmus, M. Berglund, M. Honkala, H. Valpola and T. Raiko, Semi-supervised learning with ladder networks, *Advances in Neural Information Processing Systems* (2015).
[42] J. Shi and J. Malik, Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22 issue 8, pp. 888-905 (2000).
[43] D.A. Spielman and S-H Teng, A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning, *SIAM Journal on Computing* Vol. 42 no. 1 pp. 1-26 (2013).
[44] R. C. Thompson, Principal submatrices IX: Interlacing inequalities for singular values of submatrices, *Linear Algebra and Its Applications*, Vol. 5 issue 1, pp. 1-12 (1972).
[45] A. Traud, P. Mucha and M. Porter, Social structure of facebook networks, *Physica A: Statistical Mechanics and its Applications*, Vol. 391 no. 16, pp 4165-4180 (2012)
[46] J. Wilson, S. Wang, P. Mucha, S. Bhamidi, A. Nobel *et. al.*, A testing based extraction algorithm for identifying significant communities in networks, *The Annals of Applied Statistics*, Vol 8, no. 3, pp. 1853 - 1891 (2014).
[47] L. Zelnik-Manor and P. Perona, Self-tuning spectral clustering, *Advances in neural information processing systems* (2005).
[48] Y. Zhao, E. Levina, J. Zhu, Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 108.18 pp. 7321-7326 (2011).