Compressive sensing for cut improvement and local clustering.

Ming-Jun Lai* and Daniel Mckenzie †

Abstract. We show how one can phrase the cut improvement problem for graphs as a sparse recovery problem, whence one can use algorithms originally developed for use in compressive sensing (such as `SubspacePursuit` or `CoSaMP`) to solve it. We show that this approach to cut improvement is fast, both in theory and practice and moreover enjoys statistical guarantees of success when applied to graphs drawn from probabilistic models such as the Stochastic Block Model. Using this new cut improvement approach, which we call `ClusterPursuit`, as an algorithmic primitive we then propose new methods for local clustering and semi-supervised clustering, which enjoy similar guarantees of success and speed. Finally, we verify the promise of our approach with extensive numerical benchmarking.

Key words. Cluster Extraction, Local Clustering, Cut Improvement, Semi-Supervised Clustering, Community Detection, Compressive Sensing, Sparse Solution, Graph Laplacian.

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

1. Introduction.

Finding clusters is a problem of primary interest when analyzing graphs. This is because vertices which are in the same cluster can reasonably be assumed to have some latent similarity. Thus, clustering can be used to find communities in social networks [24, 48, 53] or deduce political affiliation from a network of blogs [5]. Moreover, even data sets which are not presented as graphs can profitably be studied by first creating an auxiliary graph (*eg.* a K- or ϵ-nearest-neighbors graph) and then applying graph clustering techniques. This has been successfully applied to image segmentation [43, 37], image classification [30] and natural language processing [19].

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. See Figure 1 for a few examples. While some graphs may allow a neat partitioning into disjoint clusters (for example the OptDigits graph in Figure 1), for many graphs this is not the case. Some graphs may contain *background vertices*, that is, vertices which do not belong to any cluster (see the College Football graph in Figure 1). Alternatively, graphs may exhibit clusters at multiple scales (See the Senate Co-voting graph in Figure 1). In many cases, one has certain *a priori* information that could be used to improve clustering. For example in the OptDigits graph, we may know that some small subset, $\Gamma \subset V$, all represent images of ones. It is reasonable to assume that algorithms which incorporate this additional information (usually referred to as

*mjlai@uga.edu. Department of Mathematics, University of Georgia, Athens, GA 30602. This research is partially supported by the National Science Foundation under the grant #DMS 1521537.

†mckenzie@math.ucla.edu. Department of Mathematics, University of California, Los Angeles, CA 155505. The financial assistance of the National Research Foundation of South Africa (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and not necessarily to be attributed to the NRF. This research was conducted while this author was a graduate student at the University of Georgia and he gratefully acknowledges support and encouragement received from the Math Department of UGA.

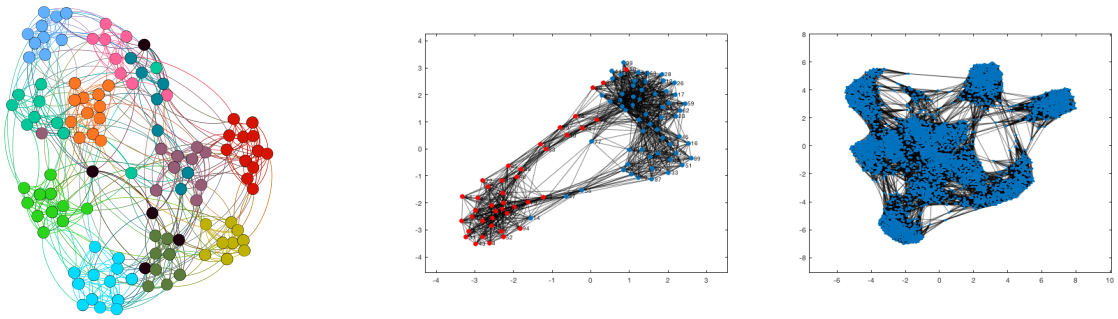

Figure 1: *Left:* the College Football graph of [24]. Vertices represents colleges fielding (American) football teams in the 2000 season. Vertices are connected if the respective teams played each other during the regular season. Clusters correspond to the various conferences in which teams play. Note that there are five schools, denoted in black, which are "independents" *ie* they are not affiliated with any conference. These can be thought of as background vertices. *Middle:* Senate co-voting for the 97th Congress, created using data from [32]. Vertices represent Senators and are connected if the respective Senators cast the same vote on a majority of bills. The two large clusters correspond to the two major American political parties. Notice how the blue cluster can be visually subdivided into two sub-clusters. *Right:* The OptDigits dataset consists of 5620 grayscale images of handwritten digits 0–9 of size 8×8. We discuss how to turn this into a graph in §10. Note that as there are ten digits, we expect this graph to have ten disjoint clusters.

semi-supervised algorithms) will perform better than ones which do not. With this in mind, it is convenient to appeal to the following taxonomy of clustering algorithms:

1. *Global clustering algorithms* assign every vertex to one of k clusters, where the clusters may or may not be disjoint. Algorithms for this problem may be unsupervised (for example `SpectralClustering` [43, 40] or `GenLouvain` [18]) or semisupervised (for example the auction dynamics approach of [30], or the regional force based methods of [54]). This is appropriate for graphs such as the OptDigits graph of Figure 1, where one expects a clear partition of the vertices into clusters.
2. *Local clustering algorithms*[1] take as input a small set of "seed vertices", $\Gamma \subset V$ and return a good cluster containing Γ. Algorithms for local clustering are not confounded by background vertices, as they are not required to assign them to a cluster. One can further sub-divide local clustering algorithms into strongly and weakly local clustering algorithms. Strongly local algorithms, for `Nibble` [45, 46], `PPR-Grow` [1] or `CapacityReleasingDiffusion` [52], are characterized by having run time proportional to the size of the cluster found. This is advantageous when the cluster in question has much fewer vertices than the graph as a whole. Weakly local algorithms are characterized as having run time proportional to the size of G. In practice they are frequently faster than strongly local algorithms when finding large or moderately large clusters.

[1] Also known as cluster extraction algorithms in the statistics literature

We note that both kinds of local clustering algorithms may take as input a scale parameter, which dictates the size of the output cluster returned. This is useful when the graph at hand contains clusters at multiple scales, such as the Senate co-voting graph of Figure 1.

3. *Cut improvement algorithms*(cf. [1], [41], [50]) take as input a cut, or subset $\Omega \subset V$, which one can think of as an approximation to a cluster C, and refine it to produce a better approximation. Often cut improvement algorithms are run on the output of a local clustering algorithm to improve the quality of the output.

The central contribution of this paper is a new cut improvement algorithm which we call `ClusterPursuit`, that phrases the cut improvement problem as a sparse recovery problem. We pair this with a simple local clustering algorithm which we call Random Walk Thresholding or `RWThresh` to obtain a two-stage weakly local clustering algorithm that we shall refer to as `CP+RWT`. One can iterate this algorithm to find all clusters in a graph; we call this procedure iterated `CP+RWT` or `ICP+RWT`. After presenting some mathematical preliminaries and outlining the assumptions we place on generative models of graphs in §2, we derive the `ClusterPursuit` algorithm in §3 and prove that, given a cut Ω satisfying $|C_1 \triangle \Omega|/|C_1| = O(1)$ `ClusterPursuit` returns $C_1^{\#}$ satisfying $|C_1 \triangle C_1^{\#}|/|C_1| = o(1)$. Here, C_1 denotes the smallest cluster in the graph. In §4 we discuss the `RWThresh` algorithm, and show that given a small set of seed vertices, $\Gamma \subset C_1$, it is capable of finding an Ω satisfying $|C_1 \triangle \Omega|/|C_1| = O(1)$. This leads naturally to guarantees of success for the two-stage local clustering algorithm `CP+RWT`, which we present in §5. In §6 we briefly discuss `ICP+RWT` while in §7 we show that `CP+RWT` and `ICP+RWT` enjoy a computational complexity of $O(nd_{\max}\log(n))$ where $d_{\max}$ is the largest vertex degree in the graph. In §8 we survey the literature and compare our work with relevant recent work in the area, while in §9 we show that a popular generative model of graphs with communities, namely the stochastic block model, satisfies the assumptions outlined in §2. Finally, we complement theoretical insight with experimental results in §10. In the interest of reproducibility, we make our code available at: danielmckenzie.github.io.

2. Preliminaries.

2.1. Graph Notation and Definitions. We restrict our attention to finite, simple, undirected graphs $G = (V, E)$, possibly with non-negative 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 any $S \subset V$ define $\mathrm{vol}(S) = \sum_{i \in S} d_i$. For quantities such as d_i (and later λ_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. By "cluster" we shall mean a subset of vertices, $C \subset V$, that is well-connected but sparsely connected to the rest of the graph. If a graph has clusters we shall refer to them as $C_1, \dots, C_k$. We define $n_a := |C_a|$ and assume that the clusters are ordered by size, so that $n_1 \leq n_2 \leq \dots, \leq n_k$. We reserve the letters a and b for indexing clusters, while i and j will index vertices.

Definition 2.1 (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 normalized, symmetric Laplacian is:* $L^{sym} := I - D^{-1/2}AD^{-1/2}$.

Recall the following elementary result in spectral graph theory (see [49], for example, for a proof):

Theorem 2.2. *Let* $C_1, \dots, C_k$ *denote the connected components of a graph* G. *Then the cluster indicator vectors* $\mathbf{1}_{C_1}, \dots, \mathbf{1}_{C_k}$ *form a basis for the kernel of* L.

Suppose that G has clusters $C_1, \dots, C_k$. 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 edges between vertices in the same cluster, while $G^{\text{out}} = (V, E^{\text{out}})$ consist only of edges between vertices in different clusters. We emphasize that this is a theoretical construction, as in practice we of course cannot ascertain whether two vertices are in the same cluster without first solving the clustering problem, which is precisely what we are trying to do. 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}). For future reference we define the random walk transition matrices $P = AD^{-1}$ and $N := D^{-1/2}AD^{-1/2}$. We note that the spectra of P, N, A, L are related:

Lemma 2.3. *For any matrix* B *with real eigenvalues let* $\lambda_i(B)$ *denote the* i*-th smallest eigenvalue, counted with multiplicity. Then* $\lambda_i(L) = \lambda_i(L^{sym})$ *while* $\lambda_{n-i}(N) = \lambda_{n-i}(P) = 1 - \lambda_i(L)$

Proof. Observe that $L = D^{-1/2}L^{\text{sym}}D^{1/2}$, hence L and L^{sym} have the same spectrum. Similarly $P = D^{1/2}\left(I - L^{\text{sym}}\right)D^{-1/2}$ hence P and $N = I - L^{\text{sym}}$ have the same spectrum. Thus if λ is the i-th smallest eigenvalue of L^{sym} it is the i-th largest (and hence the $(n-i)$-th smallest) eigenvalue of $I - L^{\text{sym}}$. ∎

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 $i, j \in S$. By A_{G_S} (resp. L_{G_S}) we mean the adjacency matrix (resp. Laplacian) of the graph G_S. Note that L_{G_S} is not a submatrix of L! 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 $i \in S$.

2.2. Compressive Sensing. Recall for any $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_0 := |\text{supp}(\mathbf{x})| = |\{i : \ x_i \neq 0\}|$ is the sparsity of $\mathbf{x}$. If $\|\mathbf{x}\|_0 \ll n$ we say that $\mathbf{x}$ is *sparse*. Candés, Donoho and their collaborators in [20, 9] pioneered the study of compressive sensing, which offers theoretical analysis and algorithmic tools for finding sparse solutions to linear systems $\Phi\mathbf{x} = \mathbf{b}$, for example by solving the minimization problem:

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

where $\Phi \in \mathbb{R}^{m \times n}$ is referred to as the *sensing matrix*. Typically, it is assumed that $m \leq n$ although this will not be the case in this paper. There are many algorithms available to solve

Algorithm 2.1 `SubspacePursuit`, as presented in [17]

Input variables: measurement matrix Φ, measurement vector $\mathbf{y}$, sparsity parameter s and number of iterations J.

Initialization:

(1) $S^{(0)} = \mathcal{L}_s(\Phi^\top \mathbf{y})$.

(2) $\mathbf{x}^{(0)} = \arg\min_{\mathbf{z}\in\mathbb{R}^N}\{\|\mathbf{y} - \Phi\mathbf{z}\|_2 : \ \mathrm{supp}(\mathbf{z}) \subset S^{(0)}\}$

(3) $\mathbf{r}^{(0)} = \mathbf{y} - \Phi\mathbf{x}^{(0)}$

for $j = 1 : J$ **do**

(1) $\hat{S}^{(j)} = S^{(j-1)} \cup \mathcal{L}_s\left(\Phi^\top \mathbf{r}^{(j-1)}\right)$

(2) $\mathbf{u} = \arg\min_{\mathbf{z}\in\mathbb{R}^N}\{\|\mathbf{y} - \Phi\mathbf{z}\|_2 : \ \mathrm{supp}(\mathbf{z}) \subset \hat{S}^{(j)}\}$

(3) $S^{(j)} = \mathcal{L}_s(\mathbf{u})$ and $\mathbf{x}^{(j)} = \mathcal{H}_s(\mathbf{u})$

(4) $\mathbf{r}^{(j)} = \mathbf{y} - \Phi\mathbf{x}^{(j)}$

end for

Problem (2.1), but the one we shall focus on is the `SubspacePursuit` algorithm introduced in [17]. 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 whether (2.1) has a unique solution, the following constant is often used (see [21])

Definition 2.4. *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).*

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 $\hat{\Phi}$ exactly. Instead, we have access only to $\Phi = \hat{\Phi} + M$, for some small perturbation M. Suppose further that there is some 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 $\mathbf{y}$, given only Φ? This question is answered in the affirmative way by several authors, starting with the work of [29]. For `SubspacePursuit`, we have the following result (cf. [33]):

Theorem 2.5. *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 [33]. 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. ∎

Next it is easy to obtain bounds on the quantity $\|B\|_2^{(s)} := \max_{\substack{S\subset[n]\\|S|=s}} \|B_S\|_2$:

Lemma 2.6. *For any matrix* B *and any* $2 \leq s \leq n$ *we have that* $\sigma_{s-1}(B) \leq \|B\|_2^{(s)} \leq \sigma_{\max}(B) = \|B\|_2$, *where* $\sigma_j(B)$ *denotes the* j*-th smallest singular value of* B.

Proof. Observe that, for any matrix B,

$$\|B\|_2^{(s)} = \max_{\substack{S\subset[n]\\|S|=s}} \|B_S\|_2 = \max_{\substack{S\subset[n]\\|S|=s}} \sigma_{\max}(B_S),$$

where $\sigma_{\max}(B_S)$ denotes the maximum singular value of B_S. Because $\sigma_{\max}(B_S) = \sigma_s(B_S)$, by the interlacing theorem for singular values (cf. [47]) $\sigma_{s-1}(B) \leq \sigma_{\max}(B_S) \leq \sigma_{\max}(B)$. ∎

2.3. The Data Model. For conceptual clarity, we shall take an asymptotic viewpoint, and consider graphs $G \in \mathcal{G}_n$ as $n \to \infty$. Note that the graphs under consideration may be weighted or unweighted. We say that a graph property P holds *almost surely* for $\mathcal{G}_n$ if the probability of a G drawn from $\mathcal{G}_n$ *not having* P is $o(1)$.

Assumptions 2.7. *Suppose that there exist* $\epsilon_i = o(1)$ *as* $n \to \infty$ *for* $i = 1, 2, 3$ *such that for all* $G \in \mathcal{G}_n$:

(A1) $V = C_1 \cup \ldots \cup C_k$ where the C_a are disjoint, planted clusters and k is $O(1)$ as $n \to \infty$.
(A2) For all $a \in [k]$ we have that $\lambda_2(L_{G_{C_a}}) \geq 1 - \epsilon_1$ and $\lambda_{n_a}(L_{G_{C_a}}) \leq 1 + \epsilon_1$ almost surely.
(A3) Letting $r_i := d_i^{\text{out}}/d_i^{\text{in}}$, $r_i \leq \epsilon_2$ for all $i \in [n]$ almost surely.
(A4) If $d_{\text{av}}^{\text{in}} := \mathbb{E}[d_i^{\text{in}}]$ then $d_{\max}^{\text{in}} \leq (1+\epsilon_3)d_{\text{av}}^{\text{in}}$ and $d_{\min}^{\text{in}} \geq (1-\epsilon_3)d_{\text{av}}^{\text{in}}$ almost surely.

Note that we can think of (A1)–(A4) as "regularity" requirements for graphs; as they insist that degrees do not vary too wildly, and that the eigenvalues are well behaved. In §9 we verify that a common model of unweighted graphs with clusters—the stochastic block model—satisfies these assumptions, so they are certainly not too restrictive. It seems probable (and indeed supported by the numerical evidence of §10) that reasonable models of random weighted graphs satisfy these properties too, although we leave this for future work.

3. The `ClusterPursuit` Algorithm. The motivation for our algorithm is the following observation. Suppose for a moment that one had access to L^{in}. Suppose further that one is given a cut Ω "near" a cluster of interest, C_a, which we shall take quantitatively to mean that $|C_a \triangle \Omega| = \epsilon |C_a|$, where $\triangle$ denotes the symmetric difference, i.e $C \triangle \Omega = (C \backslash \Omega) \cup (\Omega \backslash C)$ and $\epsilon \in (0,1)$. Letting $U = C_a \setminus \Omega$ and $W = \Omega \setminus C_a$ one observes that:

$$\begin{aligned} & \mathbf{1}_\Omega = \mathbf{1}_{C_a} + \mathbf{1}_W - \mathbf{1}_U \\ \implies & L^{\text{in}} \mathbf{1}_\Omega = L^{\text{in}} \mathbf{1}_{C_a} + L^{\text{in}} (\mathbf{1}_W - \mathbf{1}_U) \\ \implies & L^{\text{in}} \mathbf{1}_\Omega = 0 + L^{\text{in}} (\mathbf{1}_W - \mathbf{1}_U) \quad \text{(by Theorem 2.2)} \\ \implies & \mathbf{y}^{\text{in}} = L^{\text{in}} (\mathbf{1}_W - \mathbf{1}_U) \quad (\text{if } \mathbf{y}^{\text{in}} := L^{\text{in}} \mathbf{1}_\Omega) \end{aligned}$$

Solving the linear system $\mathbf{y}^{\text{in}} = L^{\text{in}} \mathbf{x}$ is unlikely to yield $\mathbf{x} = \mathbf{1}_W - \mathbf{1}_U$, as L^{in} has a large kernel (Theorem 2.2). However, Theorem 3.2 will show that one may recover $\mathbf{1}_W - \mathbf{1}_U$ as the solution to the sparse recovery problem:

$$\arg\min_{\mathbf{x} \in \mathbb{R}^n} \left\{ \|L^{\text{in}} \mathbf{x} - \mathbf{y}^{\text{in}}\|_2 : \ \|\mathbf{x}\|_0 \leq s \right\} \tag{3.1}$$

where $s \approx |C_a \triangle \Omega|$. Of course, one will not in practice have access to L^{in}, only L. Thus one needs to consider a perturbed version of (3.1):

$$\arg\min_{\mathbf{x} \in \mathbb{R}^n} \{ \|L\mathbf{x} - \mathbf{y}\|_2 : \ \|\mathbf{x}\|_0 \leq s \} \tag{3.2}$$

where $\mathbf{y} = L\mathbf{1}_\Omega$. Theorem 3.4 will show that the solution $\mathbf{x}^\#$ to the minimization problem (3.2) found by `SubspacePursuit` is a good enough approximation to $\mathbf{1}_U - \mathbf{1}_W$, hence one may infer U and W from the signed support of $\mathbf{x}^\#$. Clearly, if one knows Ω, U and W one may reconstruct C_a as $C_a = (\Omega \setminus W) \cup U$. This is the essence of `ClusterPursuit`, which we present as Algorithm 3.1.

Algorithm 3.1 `ClusterPursuit`

Input: Adjacency matrix A, initial cut Ω, estimate $s \approx |\Omega \triangle C_a|$ and $R \in [0,1)$.
(1) Compute $L = I - D^{-1}A$ and $\mathbf{y} = L\mathbf{1}_\Omega$.
(2) Let $\mathbf{x}^\#$ be the solution to

$$\text{argmin}\{ \|L\mathbf{x} - \mathbf{y}\|_2 : \ \|\mathbf{x}\|_0 \leq s \} \tag{3.3}$$

obtained after $m = O(\log(n))$ iterations of `SubspacePursuit`.
(3) Let $U^\# = \{i : \ x_i^\# < -R\}$ and $W^\# = \{i : \ x_i^\# > R\}$.
Output: $C_a^\# = \left(\Omega \setminus W^\#\right) \cup U^\#$.

Remark 3.1. `ClusterPursuit` requires as an input an estimate of $|\Omega \triangle C_a|$, which might not always be available. This is less of an issue than it might first appear as:

1. Theorem 3.4 will show that as long as $|\Omega \triangle C_a| \leq s \leq 0.13|C_a|$ `ClusterPursuit` works well.
2. If no knowledge of $|\Omega \triangle C_a|$ is available, one may run `ClusterPursuit` for various values of s and keep the returned cluster with lowest conductance.
3. Alternatively, one could consider the Lasso form of problem (3.3):

$$\operatorname{argmin}\{\|L\mathbf{x}-\mathbf{y}\|_2+\lambda\|\mathbf{x}\|_1\} = \operatorname{argmin}\{\|L\mathbf{x}-\mathbf{y}\|_2+\lambda\|\mathbf{x}\|_0\} \tag{3.4}$$

as the sparse solution is the cluster indicator $\mathbf{1}_U - \mathbf{1}_W$ which satisfies $\|\mathbf{x}\|_0 = \|\mathbf{x}\|_1$. We do not analyze this further here.

Theorem 3.2. *$\mathbf{1}_W - \mathbf{1}_U$ is the unique solution to Problem (3.1), for any graph G with clusters $C_1, \ldots, C_k$, as long as $|C_a \triangle \Omega| \leq s < n_1/2$.*

Proof. One can easily verify that $\mathbf{1}_W - \mathbf{1}_U$ is a solution to (3.1), thus it remains to show that it is the unique one. So, suppose that $\mathbf{v}$ satisfies $L^{\text{in}}\mathbf{v} = \mathbf{y}^{\text{in}}$ and that $\mathbf{v} \neq \mathbf{1}_W - \mathbf{1}_U$. Because $\mathbf{y}^{\text{in}} = L^{\text{in}}\mathbf{1}_\Omega$:

$$L^{\text{in}}\mathbf{v} - L^{\text{in}}\mathbf{1}_\Omega = 0 \implies \mathbf{v} - \mathbf{1}_\Omega \in \ker(L^{\text{in}}) \implies \mathbf{v} - \mathbf{1}_\Omega = \sum_{b=1}^{k} \alpha_b \mathbf{1}_{C_b} \quad (\text{ by Theorem 2.2})$$

$$\implies \mathbf{v} = \sum_{b=1}^{k} \alpha_b \mathbf{1}_{C_b \setminus \Omega} + \sum_{b=1}^{k} (\alpha_b + 1)\mathbf{1}_{C_b \cap \Omega}$$

Now if $\alpha_a = -1$ and $\alpha_b = 0$ for all $b \neq a$ then $\mathbf{v} = \mathbf{1}_W - \mathbf{1}_U$, which we are assuming is not the case. Hence either $\alpha_a \neq -1$, in which case $\|\mathbf{v}\|_0 \geq |C_a \cap \Omega| \geq |C_a| - |C_a \triangle \Omega| > n_a/2$, or $\alpha_b \neq 0$ for $b \neq a$ in which case $\|\mathbf{v}\|_0 \geq |C_b \setminus \Omega| \geq |C_b| - |C_a \triangle \Omega| > n_b/2$ as we are assuming that $|C_a \triangle \Omega| < n_1/2$ and $n_1 = \min_b n_b$. By assumption, $s < n_1/2$ hence in either case $\mathbf{v}$ is infeasible for Problem (3.1), as it does not satisfy the constraint $\|\mathbf{v}\|_0 \leq s$. ∎

Henceforth, we shall focus on recovering the smallest cluster, C_1. We do this to avoid a technical complication in the estimation of $\delta_{\gamma n_a}(L)$ for $a > 1$ (see Theorem 3.3 and Remark A.3). We note that as long as $n_a \approx n_1$ this is not really an issue, and the proof of Theorem 3.4 will extend to this case, albeit with a tighter bound on ϵ.

Let us now quantify the size of the perturbation in moving from (3.1) to (3.2). Define $M := L - L^{\text{in}}$ and $\mathbf{e} := \mathbf{y} - \mathbf{y}^{\text{in}}$. Recall from Theorem 2.5, that the three key parameters in perturbed compressive sensing are the restricted isometry constant of L and:

$$\epsilon_{\mathbf{y}} = \frac{\|\mathbf{e}\|_2}{\|\mathbf{y}^{\text{in}}\|_2} \quad \text{and} \quad \epsilon_L^s = \frac{\|M\|_2^{(s)}}{\|L^{\text{in}}\|_2^{(s)}} \tag{3.5}$$

as well as two secondary quantities, ρ and τ. We prove the following:

Theorem 3.3. *Suppose that $\mathcal{G}_n$ satisfies (A1)–(A4) and that $|\Omega \triangle C_1| \leq 0.13 n_1$. Then for any $\gamma \in (0,1)$ the following hold almost surely:*

1. $\epsilon_{\mathbf{y}} = o(1)$ and $\epsilon_L^{\gamma n_1} = o(1)$.

2. $\delta_{\gamma n_1}(L) \leq \gamma + o(1)$.
3. If $\delta_{3s}(L) \leq 0.45$ then $\rho \leq 0.8751$ and $\tau \leq 55.8490$ for any $s \in (0, n_1/3)$.

Proof. Part (3) follows by direct computation. For parts (1) and (2) see Appendix A. ∎

We now prove the main result of this section:

Theorem 3.4. *Suppose that A is the adjacency matrix of $G \sim \mathcal{G}_n$ satisfying assumptions (A1)–(A4), and that Ω satisfies $|C_1 \triangle \Omega| = \epsilon n_1$ with $\epsilon \leq 0.13$. If $C^{\#}$ is the output of* `ClusterPursuit` *when given inputs A, Ω, $\epsilon n_1 \leq s \leq 0.13 n_1$ and $R = 0.5$ then:*

$$\frac{\left|C_1 \triangle C_1^{\#}\right|}{|C_1|} = o(1) \qquad \textit{almost surely.}$$

Remark 3.5. $s \leq 0.13 n_1$ is a conservative upper bound on s for which the guarantees from §2.2 will hold. If one has no further information on $|C_1 \triangle \Omega|$ we recommend using this as the default value of s. Empirically (see §10.1) we still observe excellent performance when $s > 0.13 n_1$.

Proof. Recall $\mathbf{x}^{\#}$ is the solution obtained by $m = O(\log(n))$ iterations of `SubspacePursuit` on Problem 3.2, which we are regarding as a perturbation of Problem 3.1. Clearly, $0.13 n_1 < n_1/2$, hence by Theorem 3.2 $\mathbf{1}_W - \mathbf{1}_U$ is the unique solution to (3.1). By Theorem 3.3 part (2) we get $\delta_s(L) \leq 0.13 + o(1) < 0.15$ almost surely, for large enough n_1. Similarly $\delta_{3s}(L) \leq 0.45$, again almost surely for n_1 large enough. It follows from Theorem 3.3 part (3) that $\rho \leq 0.8751$ and $\tau \leq 55.8490$. We now appeal to Theorem 2.5 to obtain:

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

The second term on the right-hand side is $o(1)$ by Theorem 3.3. As long as $m \geq \log_\rho(1/n) = O(\log(n))$, we obtain that $\rho^m = 1/n = o(1)$ too. Thus:

$$\frac{\|(\mathbf{1}_U - \mathbf{1}_W) - \mathbf{x}^{\#}\|_2}{\|\mathbf{1}_U - \mathbf{1}_W\|_2} \leq o(1) \implies \|(\mathbf{1}_U - \mathbf{1}_W) - \mathbf{x}^{\#}\|_2 \leq o\left(\|\mathbf{1}_U - \mathbf{1}_W\|_2\right) = o(\sqrt{n_1}) \tag{3.6}$$

As $\|\mathbf{1}_U - \mathbf{1}_W\|_2 = \sqrt{|U| + |W|} = \sqrt{\epsilon n_1}$. In Lemma 3.6 below we show that, because $\mathbf{1}_U - \mathbf{1}_W$ is a difference of binary vectors, equation (3.6) implies that $|U \triangle U^{\#}| = o(n_1)$ and $|W \triangle W^{\#}| = o(n_1)$, and hence $|C_1 \triangle C_1^{\#}| = o(n_1)$, as required. ∎

Lemma 3.6. *Consider disjoint $T_1, T_2 \subset [n]$ and any $\mathbf{v} \in \mathbb{R}^n$. Define $T_1^{\#} = \{i : \ v_i > 0.5\}$ and $T_2^{\#} = \{i : \ v_i < -0.5\}$. If $\| (\mathbf{1}_{T_1} - \mathbf{1}_{T_2}) - \mathbf{v}\|_2 \leq D$ then:*

$$|T_1 \triangle T_1^{\#}| + |T_2 \triangle T_2^{\#}| \leq 4D^2.$$

Proof. Let $T_3 := [n] \setminus (T_1 \cup T_2)$ and write $\mathbf{v} = \mathbf{v}^{(1)} + \mathbf{v}^{(2)} + \mathbf{v}^{(3)}$ where $\mathbf{v}^{(i)}$ denotes the part of $\mathbf{v}$ supported on T_i. Observe that:

$$D^2 \geq \|(\mathbf{1}_{T_1} - \mathbf{1}_{T_2}) - \mathbf{v}\|_2^2 = \|\mathbf{1}_{T_1} - \mathbf{v}^{(1)}\|_2^2 + \| - \mathbf{1}_{T_2} + \mathbf{v}^{(2)}\|_2^2 + \|\mathbf{v}^{(3)}\|^2$$

One can easily verify that:

$$\|\mathbf{1}_{T_1} - \mathbf{v}^{(1)}\|_2^2 = \|\mathbf{1}_{T_1 \cap T_1^\#} - \mathbf{v}^{(1)}|_{T_1 \cap T_1^\#}\|^2 + \|\mathbf{1}_{T_1 \setminus T_1^\#} - \mathbf{v}^{(1)}_{T_1 \setminus T_1^\#}\|_2^2 \geq (0.5)^2 |T_1 \setminus T_1^\#|$$

Similarly, $\| - \mathbf{1}_{T_2} + \mathbf{v}^{(2)}\|_2^2 \geq (0.5)^2 |T_2 \setminus T_2^\#|$, and:

$$\|\mathbf{v}^{(3)}\|_2^2 \geq \|\mathbf{v}^{(3)}|_{T_1^\# \setminus T_1}\|^2 + \|\mathbf{v}^{(3)}|_{T_2^\# \setminus T_2}\|^2 \geq (0.5)^2 |T_1^\# \setminus T_1| + (0.5)^2 |T_2^\# \setminus T_2|$$

Putting this all together we get that:

$$D^2 \geq (0.5)^2 \left(|T_1 \setminus T_1^\#| + |T_2 \setminus T_2^\#| + |T_1^\# \setminus T_1| + |T_2^\# \setminus T_2| \right) = 0.25 \left(|T_1 \triangle T_1^\#| + |T_2 \triangle T_2^\#| \right)$$

4. The `RWThresh` algorithm. Here, we introduce a simple, diffusion-based local clustering algorithm which we call `RWThresh` (see Algorithm 4.1). We note that `RWThresh` is somewhat similar to other more sophisticated diffusion-based local clustering algorithms, such as `PPR-Grow`, `HK-Grow` and `CapacityReleasingDiffusion`. We do not claim that `RWThresh` outperforms similar existing algorithms; its main utility lies in the fact that it reliably (and provably) produces approximate cuts, Ω, that are of a high enough quality to be used as an initialization for `ClusterPursuit`.

Algorithm 4.1 `RWThresh`

Input: Adjacency matrix A, a thresholding parameter $\epsilon \in (0,1)$, seed vertices $\Gamma \subset C_1$,$\hat{n}_1 \approx n_1$ and depth of random walk t.
(1) Compute $P = AD^{-1}$ and let $\mathbf{v}^{(0)} = D\mathbf{1}_\Gamma$.
(2) Compute $\mathbf{v}^{(t)} = P^t \mathbf{v}^{(0)}$
(3) Define $\Omega = \tilde{\mathcal{L}}_{(1+\epsilon)\hat{n}_1}(\mathbf{v}^{(t)})$.
Output: $\Omega = \Omega \cup \Gamma$.

Here $\tilde{\mathcal{L}}_t(\cdot)$ is a thresholding operator, similar to $\mathcal{L}_t(\cdot)$, but that returns the indices of the t largest, not largest-in-magnitude, components of a vector. To motivate `RWThresh` we observe the following. If $\mathcal{G}_n$ satisfies assumptions (A1)–(A4) then:

1. G_{C_1} is sufficiently densely connected that after t steps the random walk has a fairly large probability of visiting every $i \in C_1$.
2. C_1 is sufficiently weakly connected to $V \setminus C_1$ that the probability of the random walk leaving C_1 after t steps is fairly small.

Hence Algorithm, 4.1 which runs a short random walk starting on Γ and takes Ω to be the set of vertices most likely to be visited, should produce an Ω which is close to our intuitive notion of a good cluster. Let us quantify this as Theorem 4.1.

Theorem 4.1. *Let $G \sim \mathcal{G}_n$ satisfy Assumptions (A1)–(A4) and let A denote the adjacency matrix of G. Let Ω denote the output of* `RWThresh` *with inputs A, any $\epsilon \in (0,1)$, any $t = O(1)$, $\hat{n}_1 = n_1$ and $\Gamma \subset C_1$ with $|\Gamma| = g\epsilon_3^{2t-1} n_1$ for any constant $g \in (0,1)$, where ϵ_3 is as in Assumption (A4)). Then $|\Omega \triangle C_1| \leq (\epsilon + o(1))n_1$ almost surely.*

Proof. The proof is left to Appendix B. ∎

We note that there are many local clustering algorithms, for example the `PPR-Grow` and `CapacityReleasingDiffusion` algorithms discussed in §8, that require only $|\Gamma| = O(1)$. However, these algorithms tend to return small clusters, typically of size $|C| = O(1)$. If $\epsilon_3 = O(1/\log(n))$, as it is in the numerical experiments of §10.1, then Theorem 4.1 requires that $|\Gamma| = O(n_1/\text{polylog}(n_1))$, which seems to be a reasonable assumption when finding a cluster of size $O(n)$. In practice, we find it suffices to take $|\Gamma| = 0.01n_1$ or $|\Gamma| = 0.02n_1$.

5. Using `ClusterPursuit` for local clustering. As mentioned earlier, using `RWThresh` to quickly generate a rough approximation to C_1, namely Ω, and then using `ClusterPursuit` to then refine this cut leads to a (weakly) local clustering algorithm. Here we verify that this approach, presented below as algorithm 5.1, works well for our model of graph.

Algorithm 5.1 `CP+RWT`

Input: Adjacency matrix A and seed vertices $\Gamma \subset C_1$. Parameters $\epsilon \in (0, 0.13)$, $s \approx \epsilon n_1$, $R \in [0, 1)$, $\hat{n}_1 \approx n_1$, $t \in \mathbb{Z}_+$

(1) Let $\Omega = \texttt{RWThresh}(A, \epsilon, \Gamma, \hat{n}_1, t)$

(2) Let $C_1^{\#} = \texttt{ClusterPursuit}(A, s, R)$

Output: $C_1^{\#}$

Theorem 5.1. *Let $G \sim \mathcal{G}_n$ satisfy Assumptions (A1)–(A4) and let A denote the adjacency matrix of G. Let $C_1^{\#}$ denote the output of* `CP+RWT` *with inputs A, $\epsilon \in (0, 0.13)$, $R = 0.5$, $\hat{n}_1 = n_1$, any $t = O(1)$, any s satisfying $\epsilon < s \leq 0.13n_1$ and $\Gamma \subset C_1$ with $|\Gamma| = g\epsilon_3^{2t-1}n_1$ for any constant $g \in (0, 1)$, where ϵ_3 is as in Assumption (A4). Then:*

$$\frac{\left|C_1 \triangle C_1^{\#}\right|}{|C_1|} = o(1)$$

almost surely, for large enough n_1.

Proof. By Theorem 4.1, the call to `RWThresh` in Step (1) of `CP+RWT` almost surely returns an Ω satisfying $|\Omega \triangle C_1| \leq (\epsilon + o(1))n_1$ for input parameters with the given values. For large enough n_1, we have that $(\epsilon + o(1))n_1 \leq s \leq 0.13n_1$, hence the call to `ClusterPursuit` in Step (2) of `CP+RWT` returns $C_1^{\#}$ with $\left|C_1 \triangle C_1^{\#}\right| / |C_1| = o(1)$ by Theorem 3.4, again almost surely. ∎

Remark 5.2. In practice (see §10) we find it generally suffices to take $t = 3$. If C_1 is densely connected, one might consider a smaller value of t, and conversely one might choose a larger value (say $t = 5$) if C_1 is sparsely connected.

6. Using `ClusterPursuit` for semi-supervised clustering. In the (global) semi-supervised clustering problem, one is given a small set of seed vertices $\Gamma_a \subset C_a$ in each cluster, usually referred to in this context as "labeled data". The goal here is to find a partition into disjoint sets: $V = C_1^{\#} \cup C_2^{\#} \cup \ldots \cup C_k^{\#}$ that closely resembles the ground truth partition $V = C_1 \cup C_2 \cup \ldots \cup C_k$. An iterated version of `CP+RWT`, which we call `ICP+RWT`, can be used to solve this problem. `ICP+RWT` is presented as algorithm 6.1. Note that in the second line of the for loop we use the shorthand $G^{(a+1)} = G^{(a)} \setminus C_a^{\#}$ to denote the graph formed from

$G^{(a)}$ by removing the vertices $C_a^{\#}$. We do not analyze the theoretical performance of `ICP+RWT` here[2] but we provide numerical evidence that `ICP+RWT` is competitive with state-of-the-art semi-supervised graph clustering algorithms in §10.3.

Algorithm 6.1 `ICP+RWT`

Input: Adjacency matrix A, labeled data $\Gamma_a \subset C_a$ for $a = 1, \ldots, k$. Parameters $\epsilon \in (0,1)$, $R \in [0,1)$, $\hat{n}_a \approx n_a$ and $s_a \approx \epsilon n_a$ for $a = 1, \ldots, k$, and $t \in \mathbb{Z}_+$
Initialize: $G^{(1)} = G$ and $A^{(1)} = A$.
for $a = 1, \ldots k$ **do**
  Let $C_a^{\#} = \texttt{CP+RWT}(A^{(a)}, \Gamma_a, \epsilon, R, s_a, \hat{n}_a, t)$
  Let $G^{(a+1)} = G^{(a)} \setminus C_a^{\#}$ and let $A^{(a+1)}$ be the adjacency matrix of $G^{(a+1)}$.
end for
Output: $C_1^{\#}, \ldots, C_k^{\#}$

7. Computational Complexity.

In this section we discuss the run times of the algorithms introduced in this paper. Let $\mathcal{T}_m$ denote the cost of a matrix-vector multiply with A, L or P (they are all of the same magnitude).

Theorem 7.1. `RWThresh` *requires* $O(n\log(n) + t\mathcal{T}_m)$ *operations, where* t *is the depth of the random walk.*

Proof. Computing $\mathbf{v}^{(t)}$ requires t matrix-vector multiplies and hence requires $O(t\mathcal{T}_m)$ operations. Sorting $\mathbf{v}^{(t)}$ in order to find Ω requires $O(n\log(n))$ operations. ∎

Let us now analyze the complexity of `ClusterPursuit`

Theorem 7.2. `ClusterPursuit` *requires* $O\left(\mathcal{T}_m \log(n)\right)$ *operations.*

Remark 7.3. Note that if A is stored as a sparse matrix then $\mathcal{T}_m = O(nd_{\max})$ in which case the run time of `ClusterPursuit` becomes $O(nd_{\max}\log(n))$.

Proof. The run time of `ClusterPursuit` is dominated by the cost of the call to `SubspacePursuit` (see Algorithm 2.1) in step (3) which costs m times the cost of each iteration. We now bound the cost of each iteration. The cost of the j-th iteration is dominated by the cost of solving the least squares problem:

$$\arg\min_{\mathbf{z}\in\mathbb{R}^n} \left\{ \|L\mathbf{z} - \mathbf{y}\|_2 : \ \mathrm{supp}(\mathbf{x}) \subset \hat{S}^j \right\}.$$

(step (4) in the "for" loop of Algorithm 2.1). Because of the support condition, and because $|\hat{S}^j| = 2s \leq 0.26n_1$, this is equivalent to the least squares problem:

$$\arg\min_{\mathbf{z}\in\mathbb{R}^{2s}} \left\{ \left\| L_{\hat{S}^j}\mathbf{z} - \mathbf{y} \right\|_2 \right\} \tag{7.1}$$

We recommend using an iterative method, such as conjugate gradient (in our implementation we use MATLAB's `lsqr` operation). Fortunately, as pointed out in [39], the matrix in question,

[2]There is a minor technical difficulty: one needs to show that if G is drawn from a model satisfying assumptions (A1)–(A4) then each $G^{(a)}$ is also drawn from a model satisfying assumptions (A1)–(A4).

$L_{\hat{S}^j}$ is extremely well conditioned. This is because $\delta_{2s}(L) \leq \delta_{3s}(L) \leq 0.45$, as shown in the proof of Theorem 3.4. By [39], specifically Proposition 3.1 and the discussion of §5, this implies that the condition number is small:

$$\kappa(L_{\hat{S}^j}^\top L_{\hat{S}_j}) := \frac{\lambda_{\max}(L_{\hat{S}^j}^\top L_{\hat{S}_j})}{\lambda_{\min}(L_{\hat{S}^j}^\top L_{\hat{S}_j})} \leq \frac{1+\delta_{2s}}{1-\delta_{2s}} \leq 2.64$$

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 (7.1) to within an acceptable tolerance. Indeed, Corollary 5.3 of [39] argues that three iterations suffices. We play it safe by performing ten iterations. The cost of each iteration of conjugate gradient is equal to (a constant times) the cost of a matrix vector multiply by $L_{\hat{S}_j}$ or $L_{\hat{S}_j}^\top$, which is $\mathcal{T}_m$. Hence the total cost of step (3) of `ClusterPursuit` is $O(m\mathcal{T}_m) = O(\log(n)\mathcal{T}_m)$ because we are taking $m = O(\log(n))$. ∎

As a direct consequence of Theorems 7.1 and 7.2, we get that `CP+RWT` runs in time $O((nd_{\max}\log(n))$. If the number of clusters, k, is $O(1)$, we get that `ICP+RWT` also runs in time $O((nd_{\max}\log(n))$.

8. Comparison with Existing Literature. `ClusterPursuit` can naturally be compared with other cut improvement algorithms such as `FlowImprove` [2], `LocalFlow` [41] and `SimpleLocal` [50]. We note that the performance guarantees for these three algorithms are of a different flavor to ours. Specifically, and translating into the notation of this paper, they bound the conductance of the improved cut, $C_1^\#$, by some function of the original cut, Ω. In contrast, our performance guarantees for `ClusterPursuit` are of a more statistical nature. In terms of run-time, `LocalFlow` and `SimpleLocal` are strongly local, so have run times $O(\mathrm{vol}(\Omega)^\alpha)$ for $\alpha \geq 1$. While this is certainly better than `ClusterPursuit` for finding small clusters, *ie* when $|\Omega| = O(1)$, these run times become less attractive for even moderate sized clusters, *eg* $|C_1| = O(\sqrt{n})$. In §10 we demonstrate that `ClusterPursuit` is several orders of magnitude faster than `FlowImprove` and `SimpleLocal` in the regime $|C_1| = O(n)$.

The idea of combining a fast, diffusion based clustering algorithm with a refinement procedure to create a local clustering algorithm is not new. See, for example, the algorithms `LEMON` [27, 35], `LOSP` and `LOSP++` [34], `LBSA` [44], and `FlowSeed` [51]. We compare `CP+RWT` to a selection of these algorithm in §10. We note that there exist many diffusion-based local clustering algorithms that may find better approximations to C_1 than `RWThresh`. See for example, `PPR-Grow` [6], `HK-Grow` [31] or `CapacityReleasingDiffusion` [52]. We emphasize that the main advantage of `RWThresh` is that it rapidly and provably finds good enough initial cuts, Ω, to be fed into `ClusterPursuit`. We show in §10 that the combination `CP+RWT` typically outperforms these diffusion-only approaches, particularly for large, sparsely connected clusters.

The analysis of `CP+RWT` contained in §3–5 can be compared to the recent works [52] and [26]. In both the performance of a local clustering algorithm on graphs drawn from a certain probabilistic model is studied. In both papers, the model is more general in one sense: there is no restriction on the structure of $V \setminus C_1$, but more restrictive in other senses: the ratio

$d^{\text{out}}/d^{\text{in}}$ must be at most $O(1/\log^2(n_1))$ in [52] while the results in [26] are most meaningful when $n_1 = O(1)$ and $d^{\text{out}} = O(1)$. In contrast, our results tackle the regime where $n_1 = O(n)$ and $d^{\text{out}}/d^{\text{in}}$ can be bounded by an arbitrarily slowly decaying function of n_1.

Finally, we mention several recent works that combine notions of sparsity and local clustering. In particular, we mention the works of Fountoulakis, Gleich, Mahoney *et al* [25, 22, 26] which introduce and study the ℓ_1 regularized page rank problem. The algorithms `LOSP` and `LOSP++` also set up and solve a sparse recovery problem, although with an additional non-negativity requirement. However, to the best of the authors' knowledge, `ClusterPursuit` is the first algorithm that explicitly phrases the problem of improving a cut, Ω, as the problem of finding a sparse change to the indicator vector $\mathbf{1}_\Omega$.

9. Which Probabilistic Models Satisfy our Assumptions?. First, we verify that a well-studied model of graphs with clusters, namely the stochastic block model, satisfies Assumptions (A1)–(A4) of §2.3. We first remind the reader of the simpler Erdős - Rènyi model:

Definition 9.1. *We say $G = (V, E)$ is drawn from the Erdős - Rènyi model on n vertices with parameter p (and write $G \sim ER(n, p)$) if $V = [n]$ and $\mathbb{P}[\{i, j\} \in E] = p$ for $i, j \in V$, with all such probabilities being independent.*

Definition 9.2 ([28, 3]). *Let $\mathbf{n} = (n_1, \ldots, n_k)$ be a vector of positive integers, and let P be a $k \times k$ symmetric matrix with entries $P_{ab} \in [0, 1]$ for all a, b. We say a graph $G = (V, E)$ is drawn from the Stochastic Block Model (written $G \sim SBM(\mathbf{n}, P)$) if there exists a partition $V = C_1 \cup C_2 \ldots \cup C_k$ with $|C_a| = n_a$ 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.*

Note that if $G \sim \text{SBM}(\mathbf{n}, P)$ then each $G_{C_a} \sim \text{ER}(n_a, P_{aa})$. Without loss of generality, we shall assume that $n_1 \leq n_2 \leq \ldots \leq n_k$. In an appendix, we shall prove the following:

Theorem 9.3. *Suppose that $n_1 = O(n) \to \infty$, $P_{aa} = \omega \log(n)/n_a$ for any $\omega \to \infty$ and $P_{ab} = (\beta + o(1)) \log(n)/n$ for any $a \neq b$ where $\beta \geq 0$ is a constant. Then $SBM(\mathbf{n}, P)$ satisfies assumptions (A1)–(A4).*

Proof. See Appendix C. ∎

As a consequence of this theorem we have that, given a small fraction of vertices in C_1, `CP+RWT` will reliably return a $C_1^{\#}$ with $|C_1 \triangle C_1| = o(n_1)$. We experimentally confirm this in §10 for $\omega \sim \log(n)$. In this regime we have that $d_{\max} = O(\log^2(n))$ with high probability, hence the run time of `CP+RWT` is $O(n \log^3(n))$ by Theorem 7.2.

It is interesting to contrast this result with what is known for the global clustering problem for the stochastic block model. There are several unsupervised algorithms, see for example[4] and [38], that return a partition $V = C_1^{\#} \cup C_2^{\#} \cup \ldots \cup C_k^{\#}$ such that $C_a^{\#} = C_a$ with high probability. However these approaches either have impractically high run times [38] or are tricky to implement in practice [4]. In contrast, `CP+RWT` has a low run time, in theory and in practice, and can be implemented in a few lines of code. In addition, the "one cluster at a time" nature of `CP+RWT` affords an additional flexibility that may be useful in certain circumstances.

On the other hand, we have had less success with using `CP+RWT` for certain random geometric graphs arising as K-NN graphs of point clouds in $\mathbb{R}^d$. We note that `CP+RWT` is most effective when the adjacency matrix of the K-NN graph is sparse but has its non-zero entries uniformly distributed. In contrast, for certain artificial data sets, for example points drawn from a thickened line or sphere embedded in a high dimensional space, this adjacency matrix tends to exhibit a banded structure—at least when nearest neighbors are determined using the Euclidean metric. Experimentally, we have observed that `CP+RWT` performs poorly on these data sets. However, this problem is to a large extent particular to the use of the Euclidean metric. In particular, when a data-driven metric such as those detailed in [36] is used to construct the K-NN graph, `CP+RWT` performs much better. Moreover, even when using the Euclidean metric `CP+RWT` still performs extremely well on real data sets, such as MNIST, COIL and Optdigits, which are frequently thought of as consisting of data points drawn from a low-dimensional manifold embedded in a high dimensional space (see §10.3).

10. Numerical Experiments.

We compare the algorithms `ClusterPursuit`, `CP+RWT` and `ICP+RWT` to the state of the art on the various problems they are designed to solve. Specifically, in §10.1 we compare the performance of `ClusterPursuit` on the cut improvement task to two baseline algorithms, namely `FlowImprove` and `SimpleLocal`, for graphs drawn from the stochastic block model. We also compare `CP+RWT` to the local clustering algorithms `HK-Grow`, `PPR-Grow` and `LBSA` for the same data.[3]. In §10.2 we repeat this experiment for social networks. We take care to choose our data sets and performance measures to allow for easy comparison with similar work in [52]. In §10.3 we test the performance of `ICP+RWT` on two data sets commonly studied in the machine learning community—MNIST and OptDigits. We provide a detailed description of the implementation of all algorithms considered in the supplementary material.

10.1. Synthetic Data Sets.

We consider graphs drawn from $\mathrm{SBM}(\mathbf{n}^{(i)}, P^{(i)})$ for two different sets of parameters. The first set: $\mathbf{n}^{(1)} = (n_1, 1.5n_1, 2.5n_1, 5n_1)$ and $P^{(1)}$ with $P_{aa} = \log^2(n)/2$ and $P_{ab} = 5\log(n)/n$ for all $a \neq b$ is designed to satisfy the conditions of Theorem 9.3 while presenting a challenge to existing clustering algorithms. The second set: $\mathbf{n}^{(2)} = (n_1, 10n_1)$ and $P^{(2)} = \begin{bmatrix} 2\log^2(n)/n & \log(n)/n \\ \log(n)/n & \log(n)/n \end{bmatrix}$ goes beyond the assumptions of Theorem 9.3 and is essentially the planted cluster model studied in [26] and elsewhere. For both sets of parameters we perform two experiments. In the first we test the performance of the three cut improvement algorithms when initialized with an Ω "close" to C_1. This Ω is found using `RWThresh`. In the second we compare the performance of `CP+RWT` with the performance of the local clustering algorithms mentioned above. For both experiments we report both run time and accuracy, as measured by the Jaccard Index in Figure 2 and in Figure 3, respectively.

10.2. Social Networks.

The well-known `facebook100` dataset consists of anonymized Facebook friendship networks at 100 American universities, and was first introduced and studied in [48]. Certain demographic markers (year of entry, residence etc.) were also collected

[3]While there are certainly other worthy local clustering algorithms that deserve to be included, such as `CapacityReleasingDiffusion` [52] and `FlowSeed` [51], we stick to algorithms with a freely available MATLAB implementation

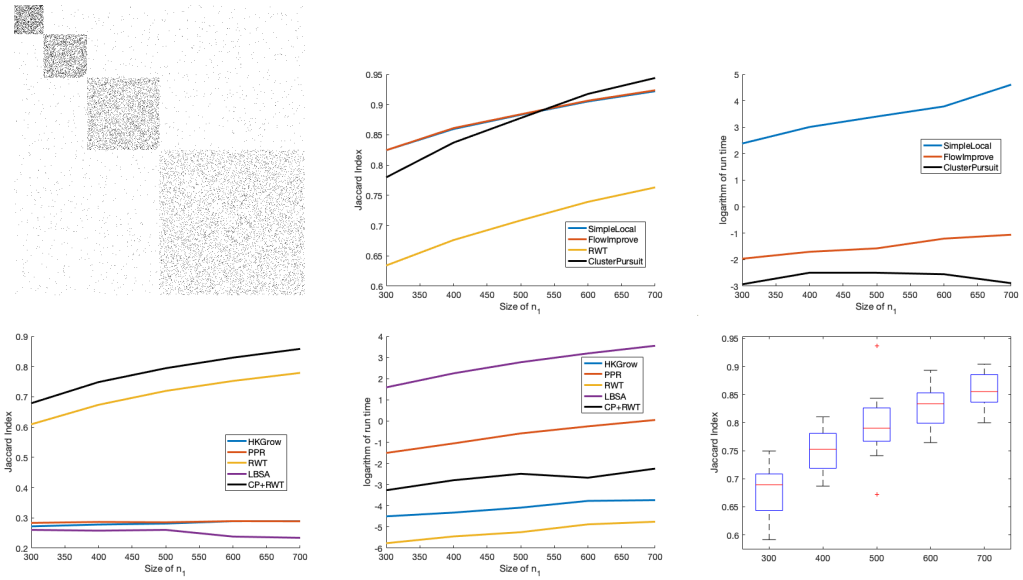

Figure 2: *Top row, left to right:* Stylized representation of the adjacency matrix of graphs drawn from SBM($\mathbf{n}^{(1)}, P^{(1)}$), Jaccard index for results of cut improvement (`SimpleLocal` and `FlowImprove` always have the same Jaccard index) and(log. of) run time for the three cut improvement algorithms. Note that `ClusterPursuit` is at least an order of magnitude faster than the other two, even though `FlowImprove` is implemented in C. *Bottom row, left to right:* Jaccard index for local clustering (The poor performance of the other methods is not an implementation issue. Rather, it is a consequence of the small gap between $P^{(1)}_{aa}$ and $P^{(1)}_{ab}$). (Log. of) run time for local clustering. Box plot of Jaccard index for `CP+RWT`.

in an anonymized format. One can think of vertices sharing the same marker as defining a ground truth cluster, although some of these clusters are extremely noisy. We focus on four clusters identified in [52] as having good (*ie* low) or moderately good conductance scores, namely Johns Hopkins class of 2009, Rice University dorm 203, Simmons College class of 2009 and Colgate University class of 2006. The details of these clusters are displayed in Table 1. For ease of comparison with the results of [52] we report accuracy using precision and recall scores. We remind the reader that, in the notation of this paper, precision $= |C_1 \cap C_1^{\#}|/|C_1^{\#}|$ and recall $= |C_1 \cap C_1^{\#}|/|C_1|$. It is desirable to have both of these values as close to 1 as possible. For all four experiments we take Γ to be selected uniformly and at random from C_1, with $|\Gamma| = 0.02n_1$. We average over fifty independent trials. There results are shown in Figure 4.

10.3. Machine Learning Benchmarks. We consider two venerable benchmark data sets:

OptDigits. This data set consists of grayscale images of handwritten digits 0–9 of size 8×8. There are $n = 5620$ images and the clusters are fairly well balanced with approximately 560 images of each digit.

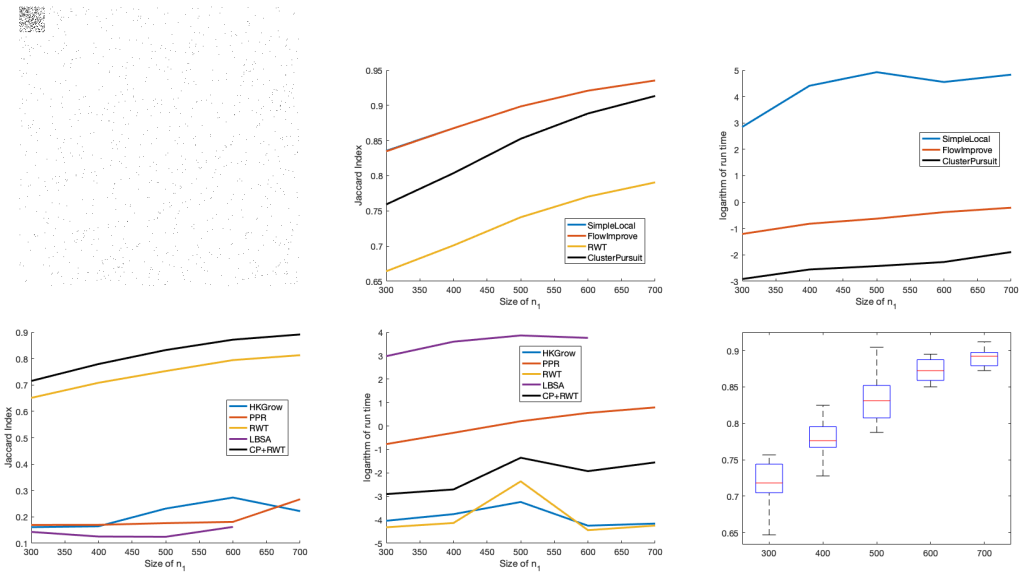

Figure 3: *Top row, left to right:* Stylized representation of the adjacency matrix of graphs drawn from SBM($\mathbf{n}^{(2)}, P^{(2)}$), Jaccard index for results of cut improvement (again, `SimpleLocal` and `FlowImprove` always have the same Jaccard index). (log. of) Run time for the three cut improvement algorithms. *Bottom row, left to right:* Jaccard index for local clustering (Again, the poor performance of the benchmark methods is a consequence of the challenging SBM parameters chosen). (Log. of) run time for local clustering. Box plot of Jaccard index for `CP+RWT`.

School	Cluster	Size of graph	Size of Cluster	Conductance
Johns Hopkins	Class of 2009	5180	910	0.21
Rice	Dorm. 203	4087	406	0.47
Simmons	Class of 2009	1518	289	0.11
Colgate	Class of 2006	3482	557	0.49

Table 1: Basic properties of the four social networks studied.

MNIST. This data set also consists of grayscale images of the handwritten digits 0–9 although here there are $n = 70\,000$ images, all of size 28×28. There are approximately 7 000 images of each digit.

For each data set we form a k-NN graph using the procedure presented in [30] and described in detail in the supplementary material. The labeled data, Γ_a, was sampled uniformly at random from C_a, and each is of size $g|C_a|$. The accuracy of the classification given by `ICP+RWT`, for increasing g, is presented in Table 2. All results are averaged over twenty independent trials.

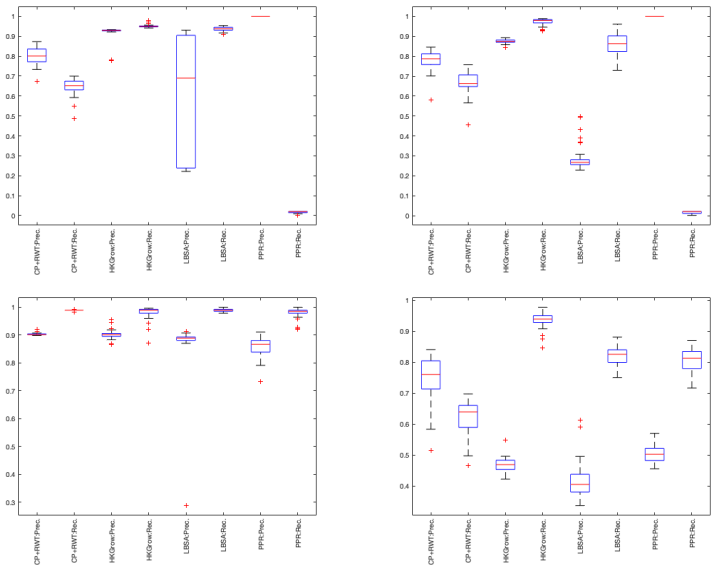

Figure 4: Precision and Recall for various local clustering algorithms on the social networks described in Table 1. Clockwise from top left: Johns Hopkins, Rice, Colgate and Simmons. Note that `CP+RWT` consistently achieves high precision without sacrificing recall.

% Labeled Data / Data Set	0.5	1	1.5	2	2.5
MNIST	96.41%	97.32%	97.44%	97.52%	97.50%
OptDigits	91.88%	95.47%	97.16%	98.06%	98.08%

Table 2: Classification accuracy, as a function of amount of labeled data, for `ICP+RWT` on two well-studied benchmark data sets.

Method	Labeled	Accuracy
TVRF [54]	600	96.8%
`ICP+RWT`	700	97.32%
Multi-Class MBO with Auction Dynamics [30]	700	97.43%
`ICP+RWT`	1050	97.44%
Ladder Networks [42]	1000	99.16%

Table 3: Comparing `ICP+RWT` to other, state-of-the-art, semi-supervised methods on MNIST. TVRF and Multi-Class MBO are graph-based, and have similar run times to `ICP+RWT`. The Ladder Network approach uses a deep neural network and hence requires training ($\sim$ 2 hours on a GPU) before it can be used for classification.

REFERENCES

[1] Reid Andersen, Kevin J. Lang, Communities from seed sets, Proceedings of the 15th International Conference on World Wide Web, 223–232, 2006.

[2] Reid Andersen, Kevin J. Lang, An algorithm for improving graph partitions, Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, 2008.

[3] Emmanuel Abbe. Community detection and stochastic block models: Recent developments. The Journal of Machine Learning Research, 18(1):6446–6531, 2017.

[4] Emmanuel Abbe and Colin Sandon. Recovering communities in the general stochastic block model without knowing the parameters. In Advances in Neural Information Processing Systems, 676–684, 2015.

[5] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 US election: Divided they blog. In Proceedings of the 3rd International Workshop on Link Discovery, 36–43, 2005.

[6] Reid Andersen, Fan Chung, and Kevin Lang. Using pagerank to locally partition a graph. Internet Mathematics, 4(1):35–64, 2007.

[7] Béla Bollobás. Vertices of given degree in a random graph. Journal of Graph Theory, 6(2):147–155, 1982.

[8] Béla Bollobás. Random graphs, Cambridge University Press, 2001.

[9] Emmanuel J. Candes, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on Information Theory, 52(2):489–509, 2006.

[10] Olivier Chappelle, Bernard Schölkopf and Alexander Zien. Semi-Supervised Learning. MIT Press. 2006.

[11] Fan Chung. Spectral graph theory (CBMS Regional Conference Series in Mathematics, no. 92). 1996.

[12] Fan Chung. The heat kernel as the pagerank of a graph. Proceedings of the National Academy of Sciences, 104(50):19735–19740, 2007.

[13] Fan Chung. Random walks and local cuts in graphs. Linear Algebra and its Applications, 423(1):22–32, 2007.

[14] Fan Chung and Ron Graham. Quasi-random graphs with given degree sequences. Random Structures & Algorithms, 32(1):1–19, 2008.

[15] Fan Chung and Mary Radcliffe, On the spectra of general random graphs. The Electronic Journal of Combinatorics, 18(1):215, 2011.

[16] Fan Chung and Olivia Simpson. Computing heat kernel pagerank and a local clustering algorithm. European Journal of Combinatorics, 68:96–119, 2018.

[17] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. IEEE Transactions on Information Theory, 55(5):2230–2249, 2009.

[18] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. Generalized Louvain method for community detection in large networks. In 2011 11th International Conference on Intelligent Systems Design and Applications: 88–93, 2011.

[19] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining:269–274, 2001.

[20] David L. Donoho. Compressed sensing. IEEE Transactions on Information Theory, 52(4):1289–1306, 2006.

[21] Simon Foucart and Holger Rauhut. A Mathematical Introduction to Compressive Sensing. Springer Science & Business Media, 2013.

[22] Kimon Fountoulakis, David Gleich and Michael Mahoney. An optimization approach to locally-biased graph algorithms. Proceedings of the IEEE 105.2: 256-272, 2017.

[23] Alan Frieze and Michal Karonński. Introduction to random graphs. Cambridge University Press, 2016.

[24] Michelle Girvan and Mark E.J. Newman. Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12):7821–7826, 2002.

[25] David Gleich and Michael Mahoney. Anti-differentiating approximation algorithms: A case study with min-cuts, spectral and flow. International Conference on Machine Learning, 2014.
[26] Wooseok Ha, Kimon Fountoulakis, and Michael W. Mahoney. Statistical guarantees for local graph clustering. arXiv preprint arXiv:1906.04863 (2019).
[27] Kun He, Yiwei Sun, David Bindel, John Hopcroft, and Yixuan Li. Detecting overlapping communities from local spectral subspaces. In 2015 IEEE International Conference on Data Mining: 769–774, 2015.
[28] Paul Holland, Kathryn Blackmond Laskey and Samuel Leinhardt. Stochastic blockmodels: First steps. Social Networks, 5(2): 109–137, 1983.
[29] Matthew A. Herman and Thomas Strohmer. General deviants: An analysis of perturbations in compressed sensing. IEEE Journal of Selected Topics in Signal Processing, 4(2):342–349, 2010.
[30] Matt Jacobs, Ekaterina Merkurjev, and Selim Esedoǵlu. Auction dynamics: A volume constrained MBO scheme. Journal of Computational Physics, 354:288–310, 2018.
[31] Kyle Kloster and David F. Gleich. Heat kernel based community detection. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 1386–1395, 2014.
[32] Jeffrey Lewis, Keith Poole, Howard Rosenthal, Adam Boche, Aaron Rudkin, and Luke Sonnet. Voteview: Congressional Roll-Call Votes Database. https://voteview.com/
[33] Haifeng Li. Improved analysis of SP and CoSaMP under total perturbations. EURASIP Journal on Advances in Signal Processing, 2016(1):112, 2016.
[34] Yixuan Li, Kun He, David Bindel, and John E. Hopcroft. Uncovering the small community structure in large networks: A local spectral approach. In Proceedings of the 24th International Conference on World Wide Web: 658–668, 2015.
[35] Yixuan Li, Kun He, Kyle Kloster, David Bindel, and John E. Hopcroft. Local spectral clustering for overlapping community detection. ACM Transactions on Knowledge Discovery from Data (TKDD), 12(2):17, 2018.
[36] Daniel McKenzie and Steven Damelin. Power weighted shortest paths for clustering Euclidean data. Foundations of Data Science, 1.3 (2019).
[37] Michael W. Mahoney, Lorenzo Orecchia, and Nisheeth K. Vishnoi. A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally. Journal of Machine Learning Research, 13(Aug):2339–2365, 2012.
[38] Elchanan Mossel, Joe Neeman and Allan Sly. A proof of the block model threshold conjecture. Combinatorica 38.3: 665–708, 2018.
[39] Deanna Needell and Joel A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. Applied and Computational Harmonic Analysis, 26(3):301–321, 2009.
[40] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In Advances in Neural Information Processing Systems: 849–856, 2002.
[41] Lorenzo Orecchia, Zeyuan Allen Zhu. Flow-based algorithms for local graph clustering. Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms: 1267–1286, 2014.
[42] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semisupervised learning with ladder networks. In Advances in Neural Information Processing Systems: 3546–3554, 2015.
[43] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888–905, 2000.
[44] Pan Shi, Kun He, David Bindel, and John E. Hopcroft. Locally-biased spectral approximation for community detection. Knowledge-Based Systems, 164:459–472, 2019.
[45] Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, In Proceedings of the STOC (4), 2004.
[46] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. SIAM Journal on Computing, 42(1):1–26, 2013.

[47] Robert C. Thompson. Principal submatrices IX: Interlacing inequalities for singular values of submatrices. Linear Algebra and its Applications, 5(1):1–12, 1972.
[48] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of Facebook networks. Physica A: Statistical Mechanics and its Applications, 391(16):4165–4180, 2012.
[49] Ulrike Von Luxburg. A tutorial on spectral clustering. Statistics and Computing, 17(4):395–416, 2007.
[50] Nate Veldt, David F. Gleich, Micheal W. Mahoney, A simple and strongly-local flow-based method for cut improvement, International Conference on Machine Learning (2016).
[51] Nate Veldt, Christine Klymko, David F. Gleich, Flow-based local graph clustering with better seed set inclusion, Proceedings of the 2019 SIAM International Conference on Data Mining, 378–386 (2019).
[52] Di Wang, Kimon Fountoulakis, Monika Henziger, Michael W. Mahoney, Satish Rao, Capacity releasing diffusion for speed and locality, Proceedings of the 34th International Conference on Machine Learning, 70:3598–3607, 2017.
[53] James D. Wilson, Simi Wang, Peter J. Mucha, Shankar Bhamidi, Andrew B. Nobel, A testing based extraction algorithm for identifying significant communities in networks, The Annals of Applied Statistics, 8(3):1853–1891, 2014.
[54] Ke Yin and Xue-Cheng Tai. An effective region force for some variational models for learning and clustering, Journal of Scientific Computing, 74 (2018), 175–196.
[55] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In Advances in Neural Information Processing Systems, pages 1601–1608, 2005.

A. Restricted Isometry Property for Laplacians.

In this section, we prove parts (1) and (2) of Theorem 3.3. We proceed via a series of lemmas.

A.1. Restricted Isometry Property for L^{in}.

Lemma A.1. *Let G be any connected graph on n_0 vertices, and let $s < n_0$. Let $\lambda_i := \lambda_i(L)$ denote the i-th smallest eigenvalue of L. Then:*

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

Proof. Recall that the s-Restricted Isometry Constant $\delta_s(L)$ is the smallest δ such that, for any $\mathbf{v}$ with $\|\mathbf{v}\|_0 \leq s$ and $\|\mathbf{v}\|_2 = 1$: $(1 - \delta) \leq \|L\mathbf{v}\|_2^2 \leq (1 + \delta)$. The RHS bound is straightforward since

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

The LHS bound requires some work. Recall that $L = I - D^{-1}A$. This matrix is not symmetric, but $L^{\text{sym}} = I - D^{-1/2}AD^{-1/2}$ is. By Lemma 2.3 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, [11]) and in particular $\mathbf{w}_1 = \frac{1}{\sqrt{\text{vol}(G)}} D^{1/2}\mathbf{1}$ where $\mathbf{1}$ is the all-ones vector. Observe that:

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

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

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

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:

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

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:

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

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

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

Returning to equation (A.1):

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

These yield the desired estimate. ∎

Theorem A.2. *Let $G \sim \mathcal{G}_n$ with $\mathcal{G}_n$ satisfying (A2) and (A4). Then for any $\gamma \in (0,1)$, we have that $\delta_{\gamma n_a}(L^{in}) \leq \frac{n_a}{n_1}\gamma + o(1)$.*

Proof. Firstly, observe that L^{in} is block diagonal with blocks $L_{G_{C_b}}$. For any block diagonal matrix we have that $\delta_s(L^{\text{in}}) = \max_b \delta_s(L_{G_{C_b}})$. By Lemma A.1 we have that:

$$\delta_s(L_{G_{C_b}}) \leq \max_b \{1 - \lambda_2(L_{G_{C_b}})^2 \left(\frac{d_{\min}^{\text{in}}}{d_{\max}^{\text{in}}} - \frac{d_{\max}^{\text{in}}}{d_{\min}^{\text{in}}} \frac{s}{n_b}\right), \lambda_{\max}(L_{G_{C_b}})^2 - 1\}. \tag{A.2}$$

From assumption (A4) we get that:

$$\frac{d_{\min}^{\text{in}}}{d_{\max}^{\text{in}}} = \frac{1-\epsilon_3}{1+\epsilon_3} = 1 - o(1) \quad \text{and} \quad \frac{d_{\max}^{\text{in}}}{d_{\min}^{\text{in}}} = \frac{1+\epsilon_3}{1-\epsilon_3} = 1 + o(1).$$

From assumption (A2) we get that:

$$\lambda_2(L_{G_{C_b}})^2 \geq (1-\epsilon_1)^2 = 1 - 2\epsilon_1 + \epsilon_1^2 = 1 - o(1)$$

and similarly $\lambda_{\max}(L_{G_{C_b}})^2 - 1 = o(1)$. Plugging this in to (A.2) with $s = \gamma n_a$ gives:

$$\delta_{\gamma n_a}(L_{G_{C_b}}) \leq \max\left\{\frac{\gamma n_a}{n_b} + o(1), o(1)\right\} \leq \gamma \frac{n_a}{n_1} + o(1) \implies \delta_{\gamma n_a}(L^{\text{in}}) \leq \gamma \frac{n_a}{n_1} + o(1).$$

∎

Remark A.3. We note that the RIP is only meaningful for $\delta_{\gamma n_a} < 1$. Hence the above theorem is only meaningful for $\gamma < \frac{n_1}{n_a} - o(1)$. To avoid this complicating technicality, we henceforth assume that $a = 1$, *i.e.* that the target cluster is C_1.

A.2. Bounding the size of the Perturbation.

Theorem A.4. *Suppose that $G \sim \mathcal{G}_n$ with $\mathcal{G}_n$ satisfying (A3). If L denotes the Laplacian of G and $M := L - L^{in}$ then $\|M\|_2 \leq o(1)$.*

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

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

Earlier we defined $r_i = d_i^{\text{out}}/d_i^{\text{in}}$. We now use the following easily verifiable identity:

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

Thus:

$$\begin{aligned}
L_{ij} &= \delta_{ij} - \left(\frac{1}{d_i^{\text{in}}} - \frac{1}{d_i^{\text{in}}} \left(\frac{r_i}{r_i + 1} \right) \right) \left(A_{ij}^{\text{in}} + A_{ij}^{\text{out}} \right) \\
&= \left(\delta_{ij} - \frac{1}{d_i^{\text{in}}} A_{ij}^{\text{in}} \right) - \frac{1}{d_i^{\text{in}}} A_{ij}^{\text{out}} + \frac{1}{d_i^{\text{in}}} \left(\frac{r_i}{r_i + 1} \right) \left(A_{ij}^{\text{in}} + A_{ij}^{\text{out}} \right) \\
&= L_{ij}^{\text{in}} - \frac{1}{d_i^{\text{in}}} \left(1 - \frac{r_i}{r_i + 1} \right) A_{ij}^{\text{out}} + \frac{1}{d_i^{\text{in}}} \left(\frac{r_i}{r_i + 1} \right) A_{ij}^{\text{in}} \\
&= L_{ij}^{\text{in}} - \frac{1}{d_i^{\text{in}}} \left(\frac{1}{r_i + 1} \right) A_{ij}^{\text{out}} + \frac{1}{d_i^{\text{in}}} \left(\frac{r_i}{r_i + 1} \right) A_{ij}^{\text{in}}.
\end{aligned}$$

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

$$\begin{aligned}
\|M\|_2 &= \max_i \{ |\mu_i| : \ \mu_i \text{ eigenvalue of } M \} \leq \max_i \sum_j |M_{ij}| \\
&= \max_i \frac{1}{d_i^{\text{in}}} \left(\frac{1}{r_i + 1} \right) \sum_j A_{ij}^{\text{out}} + \frac{1}{d_i^{\text{in}}} \left(\frac{r_i}{r_i + 1} \right) \sum_j A_{ij}^{\text{in}} \\
&= \max_i \left\{ \frac{1}{d_i^{\text{in}}} \left(\frac{1}{r_i + 1} \right) (d_i^{\text{out}}) + \frac{1}{d_i^{\text{in}}} \left(\frac{r_i}{r_i + 1} \right) (d_i^{\text{in}}) \right\} \\
&= \max_i \left\{ \left(\frac{r_i}{r_i + 1} \right) + \left(\frac{r_i}{r_i + 1} \right) \right\} \leq 2 \max_i r_i \leq 2\epsilon_2 = o(1)
\end{aligned}$$

by (A3). ∎

Theorem A.5. *Suppose that $G \sim \mathcal{G}_n$ with $\mathcal{G}_n$ satisfying (A1)–(A4). If L denotes the Laplacian of G and $|C_1 \triangle \Omega| = \epsilon n_1$ with $\epsilon \leq 0.13$ then $\epsilon_{\mathbf{y}} = o(1)$ and $\epsilon_L^{\gamma n_1} = o(1)$ for any $\gamma \in (0, 1)$.*

Proof. Recall that $\epsilon_{\mathbf{y}} = \dfrac{\|\mathbf{e}\|_2}{\|\mathbf{y}^{\text{in}}\|_2}$ and $\epsilon_L^{\gamma n_1} = \dfrac{\|M\|_2^{(\gamma n_1)}}{\|L^{\text{in}}\|_2^{(\gamma n_1)}}$. Using the bound on the restricted isometry constant of L^{in} from Theorem A.2 we have:

$$\begin{aligned}\|\mathbf{y}^{\text{in}}\|_2^2 = \|L^{\text{in}}(\mathbf{1}_W - \mathbf{1}_U)\|_2^2 &\geq \left(1 - \delta_{\epsilon n_1}(L^{\text{in}})\right)\|\mathbf{1}_W - \mathbf{1}_U\|_2^2 \\ &\geq (\epsilon - o(1))\,|C_1 \triangle \Omega| = (\epsilon^2 - o(1))n_1\end{aligned}$$

Thus $\|\mathbf{y}^{\text{in}} + \mathbf{z}^{\text{in}}\|_2 \geq \sqrt{\epsilon^2 - o(1)}\sqrt{n_1}$. On the other hand:

$$\|\mathbf{e}\|_2 = \|\mathbf{y} - \mathbf{y}^{\text{in}}\|_2 = \|L\mathbf{1}_\Omega - L^{\text{in}}\mathbf{1}_\Omega\|_2 = \|M\mathbf{1}_\Omega\|_2 \leq \|M\|_2\|\mathbf{1}_\Omega\|_2 \leq o(1)\sqrt{(1+\epsilon)n_1}$$

Thus:

$$\epsilon_{\mathbf{y}} = \frac{\|\mathbf{e}\|_2}{\|\mathbf{y}^{\text{in}}\|_2} \leq \frac{o(1)\sqrt{(1+\epsilon)}\sqrt{n_1}}{\sqrt{(\epsilon^2 - o(1))}\sqrt{n_1}} = o(1)$$

as ϵ is a constant, *i.e.* independent of n_1. The bound on $\epsilon_L^{\gamma n_1}$ is easier. By Lemma 2.6 and Property 3:

$$\|L^{\text{in}}\|_2^{(\gamma n_1)} \geq \sigma_{\gamma n_1 - 1}(L^{\text{in}}) = \lambda_{\gamma n_1 - 1}(L^{\text{in}}) \geq \lambda_{k+1}(L^{\text{in}})$$

as long as $\gamma n_1 \geq k + 3$, which is certainly the case for large enough n_1. Because $\lambda_1(L_{G_{C_1}}) = \ldots = \lambda_1(L_{G_{C_k}}) = 0$ and the spectrum of L^{in} is the union of the spectra of the $L_{G_{C_a}}$, it follows that:

$$\lambda_{k+1}\left(L^{\text{in}}\right) = \min_{a=1}^{k} \lambda_2(L_{G_{C_a}}) \geq 1 - \epsilon_1 = 1 - o(1)$$

by (A1). By Theorem A.4 and Lemma 2.6 $\|M\|_2^{(\gamma n_1)} \leq \|M\|_2 = o(1)$. It follows that:

$$\epsilon_L^{\gamma n_1} = \frac{\|M\|_2^{(\gamma n_1)}}{\|L^{\text{in}}\|_2^{(\gamma n_1)}} = \frac{o(1)}{1 - o(1)} = o(1).$$

A.3. Restricted Isometry Property for L. Finally, we extend from $\delta_s(L^{\text{in}})$ to $\delta_s(L)$ using the following result of Herman and Strohmer (cf. [29]):

Theorem A.6. *Suppose that $\Phi = \hat{\Phi} + M$. Let $\hat{\delta}_s$ and δ_s denote the s restricted isometry constants of $\hat{\Phi}$ and Φ respectively. Then:*

$$\delta_s \leq (1 + \hat{\delta}_s)\,(1 + \epsilon_\Phi^s)^2 - 1.$$

Corollary A.7. *Let L denote the Laplacian of $G \sim \mathcal{G}_n$ satisfying (A1)–(A4). Then we have $\delta_{\gamma n_1}(L) \leq \gamma + o(1)$ for any $\gamma \in (0, 1)$.*

Proof. By Theorem A.6 we have that:

$$\delta_{\gamma n_1}(L) \leq (1 + \delta_{\gamma n_1}(L^{\text{in}}))(1 + \epsilon_L^{\gamma n_1})^2 - 1.$$

Substituting the values of $\delta_{\gamma n_1}(L^{\text{in}})$ and $\epsilon_L^{\gamma n_1}$ from Theorems A.2 and A.5 yields the claim. ∎

B. Proof of Theorem 4.1. Before proving this theorem we prove the a series of lemmas. We first note that Assumptions (A3) and (A4) easily allow us to bound $\mathrm{vol}(S)$, which will be required in the proof of Theorem 4.1:

Lemma B.1. *Suppose that $\mathcal{G}_n$ satisfies (A3) and (A4). For any $S \subset V$ define $vol^{in}(S) = \sum_i d_i^{in}$. Then for any $G \in \mathcal{G}_n$ we have that:*

(1) $(1-\epsilon_3)|S|d_{av}^{in} \leq vol^{in}(S) \leq (1+\epsilon_3)|S|d_{av}^{in}$; *and* (2) $vol^{in}(S) \leq vol(S) \leq (1+\epsilon_2)vol^{in}(S)$.

Proof. For part (1), observe that:

$$\mathrm{vol}^{\mathrm{in}}(S) = \sum_{i\in S} d_i^{\mathrm{in}} \geq |S| d_{\min}^{\mathrm{in}} \geq |S|(1-\epsilon_3)d_{\mathrm{av}}^{\mathrm{in}},$$

where the final inequality is from (A4). The bound $\mathrm{vol}^{\mathrm{in}}(S) \leq (1+\epsilon_3)|S|d_{\mathrm{av}}^{\mathrm{in}}$ follows similarly. For part (2) we note that by assumption (A3) $d_i = d_i^{\mathrm{in}} + d_i^{\mathrm{out}} \leq d_i^{\mathrm{in}} + \epsilon_2 d_i^{\mathrm{in}} = (1+\epsilon_2)d_i^{\mathrm{in}}$. Hence:

$$\mathrm{vol}(S) = \sum_{i\in S} d_i \leq \sum_{i\in S}(1+\epsilon_2)d_i^{\mathrm{in}} = (1+\epsilon_2)\mathrm{vol}^{\mathrm{in}}(S)$$

while the lower bound follows simply from the fact that $d_i \geq d_i^{\mathrm{in}}$. ∎

Lemma B.2. *Let $G \in \mathcal{G}_n$ satisfies Assumptions (A1)–(A4). If $N_{G_{C_1}} := D_{G_{C_1}}^{-1/2} A_{G_{C_1}} D_{G_{C_1}}^{-1/2}$ and $U, \Gamma \subset C_1$ then:*

$$\left| \langle D_{G_{C_1}}^{1/2} \mathbf{1}_U, N_{G_{C_1}}^t D_{G_{C_1}}^{1/2} \mathbf{1}_\Gamma \rangle - \frac{vol^{in}(U)vol^{in}(\Gamma)}{vol^{in}(G_{C_1})} \right| \leq \epsilon_1^t \sqrt{vol^{in}(U)vol^{in}(\Gamma)}$$

Proof. From the proof of Lemma 2 in [14] (note that they use $M_{G_{C_1}}$ instead of $N_{G_{C_1}}$) we get that:

$$\left| \langle D_{G_{C_1}}^{1/2} \mathbf{1}_U, N_{G_{C_1}}^t D_{G_{C_1}}^{1/2} \mathbf{1}_\Gamma \rangle - \frac{\mathrm{vol}^{\mathrm{in}}(U)\mathrm{vol}^{\mathrm{in}}(\Gamma)}{\mathrm{vol}^{\mathrm{in}}(G_{C_1})} \right| \leq \lambda_{n_1-1}(N_{G_{C_1}})^t \sqrt{\mathrm{vol}^{\mathrm{in}}(U)\mathrm{vol}^{\mathrm{in}}(\Gamma)}$$

By Lemma 2.3 and (A2) we get that $\lambda_{n_1-1}(N_{G_{C_1}}) = 1 - \lambda_2(L_{G_{C_1}}) \leq \epsilon_1$. ∎

Proof of Theorem 4.1. As in §3, let $U = C_1 \setminus \Omega$ and $W = \Omega \setminus C_1$. Let $|U| = un_1$, in which case $|W| = (\epsilon+u)n_1$. We shall prove that $u = o(1)$. By definition, Ω is the set of the $(1+\epsilon)n_1$ largest entries in $\mathbf{v}^{(t)} := P^t D \mathbf{1}_\Gamma$. Because U is not in Ω, but W is, we must have $v_i^{(t)} \leq v_j^{(t)}$ for every $i \in U$ and $j \in W$. We sum first over $j \in W$ and then sum over $i \in U$ to obtain:

$$v_i^{(t)} \leq v_j^{(t)} \implies (\epsilon+u)n_1 v_i^{(t)} \leq \sum_{j\in W} v_j^{(t)} \implies (\epsilon+u)n_1 \sum_{i\in U} v_i^{(t)} \leq un_1 \sum_{j\in W} v_j^{(t)}.$$

It follows that:

$$\sum_{i\in U} v_i^{(t)} \leq \frac{u}{\epsilon+u} \sum_{j\in W} v_j^{(t)} \leq \sum_{j\in W} v_j^{(t)}. \tag{B.1}$$

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

We first show that the term on the left-hand side of inequality B.1, *i.e.* the sum over the vertices in C_1 that were missed by Ω, is necessarily quite large. We do this by relating P to P^{in}, the random walk transition matrix for the graph G^{in}. Note that G^{in} is a disjoint union of the graphs G_{C_a}. For every $i \in [n]$, define $q_i := d_i^{\text{in}}/d_i$. Observe that $1/d_i = q_i/d_i^{\text{in}}$ and thus $D^{-1} = D_{\text{in}}^{-1}Q$ where Q is the diagonal matrix with (i,i)-th entry q_i. Now:

$$P = AD^{-1} = \left(A^{\text{in}} + A^{\text{out}}\right) D^{-1} = A^{\text{in}}\left(D_{\text{in}}^{-1}Q\right) + A^{\text{out}}D^{-1} = P^{\text{in}}Q + A^{\text{out}}D^{-1}.$$

Observe that P, $P^{\text{in}}Q$ and $A^{\text{out}}D^{-1}$ all have non-negative entries. It follows that for any non-negative vector $\mathbf{x}$: $P\mathbf{x}$ and $P^{\text{in}}Q\mathbf{x}$ are also non-negative and $P\mathbf{x} \geq P^{\text{in}}Q\mathbf{x}$, where the inequality should be interpreted componentwise. One can the extend the inequality by iterated multiplication:

$$P^t\mathbf{x} \geq \left(P^{\text{in}}Q\right)^t\mathbf{x} \geq q_{\min}^t\left(P^{\text{in}}\right)^t\mathbf{x}$$

and again the inequality should be interpreted componentwise. Now:

$$\begin{aligned}\sum_{i\in U} v_i^{(t)} &= \langle \mathbf{1}_U, \mathbf{v}^{(t)}\rangle = \langle \mathbf{1}_U, P^tD\mathbf{1}_\Gamma\rangle \geq \langle \mathbf{1}_U, q_{\min}^t\left(P^{\text{in}}\right)^t D\mathbf{1}_\Gamma\rangle \\ &= q_{\min}^t\langle \mathbf{1}_U, \left(P^{\text{in}}\right)^t D_{\text{in}}\mathbf{1}_\Gamma\rangle = q_{\min}^t\langle \mathbf{1}_U, \left(P_{G_{C_1}}\right)^t D_{G_{C_1}}\mathbf{1}_\Gamma\rangle,\end{aligned}$$

where the final line follows as $U, \Gamma \subset C_1$.

Our goal now is to bound the quantity $\langle \mathbf{1}_U, \left(P_{G_{C_1}}\right)^t D_{G_{C_1}}\mathbf{1}_\Gamma\rangle$. One can rearrange the iterated matrix product slightly:

$$\begin{aligned}\left(P_{G_{C_1}}\right)^t &= \left(A_{G_{C_1}}D_{G_{C_1}}^{-1}\right)^t = A_{G_{C_1}}D_{G_{C_1}}^{-1}A_{G_{C_1}}D_{G_{C_1}}^{-1}\ldots A_{G_{C_1}}D_{G_{C_1}}^{-1} \\ &= D_{G_{C_1}}^{1/2}\left(D_{G_{C_1}}^{-1/2}A_{G_{C_1}}D_{G_{C_1}}^{-1/2}\right)\left(D_{G_{C_1}}^{-1/2}A_{G_{C_1}}D_{G_{C_1}}^{-1/2}\right)\ldots\left(D_{G_{C_1}}^{-1/2}A_{G_{C_1}}D_{G_{C_1}}^{-1/2}\right)D_{G_{C_1}}^{-1/2} \\ &= D_{G_{C_1}}^{1/2}N_{G_{C_1}}^tD_{G_{C_1}}^{-1/2},\end{aligned}$$

Hence, we have

$$\begin{aligned}\langle \mathbf{1}_U, \left(P_{G_{C_1}}\right)^t D_{G_{C_1}}\mathbf{1}_\Gamma\rangle &= \langle \mathbf{1}_U, \left(D_{G_{C_1}}^{1/2}N_{G_{C_1}}^tD_{G_{C_1}}^{-1/2}\right)D_{G_{C_1}}\mathbf{1}_\Gamma\rangle \\ &= \langle D_{G_{C_1}}^{1/2}\mathbf{1}_U, N_{G_{C_1}}^tD_{G_{C_1}}^{1/2}\mathbf{1}_\Gamma\rangle \geq \frac{\text{vol}^{\text{in}}(U)\text{vol}^{\text{in}}(\Gamma)}{\text{vol}^{\text{in}}(G_{C_1})} - \epsilon_1^t\sqrt{\text{vol}^{\text{in}}(U)\text{vol}^{\text{in}}(\Gamma)},\end{aligned}$$

where the final inequality follows from Lemma B.2. Returning to (20):

$$\sum_{i\in U} v_i^{(t)} \geq q_{\min}^t\left(\frac{\text{vol}^{\text{in}}(U)\text{vol}^{\text{in}}(\Gamma)}{\text{vol}^{\text{in}}(G_{C_1})} - \epsilon_1^t\sqrt{\text{vol}^{\text{in}}(U)\text{vol}^{\text{in}}(\Gamma)}\right). \tag{B.2}$$

We now consider the right hand side of (B.1), *i.e.* the sum over W. Because $W \subset V \setminus C_1$ we have that:

$$\sum_{j\in W} v_j^{(t)} \leq \sum_{j\in V\setminus C_1} |v_j^{(t)}| = \|\mathbf{v}_{V\setminus C_1}^{(t)}\|_1$$

Thus it remains to bound $\|\mathbf{v}^{(t)}_{V\setminus C_1}\|_1$. Observe that:

$$\mathbf{v}^{(t)}_{V\setminus C_1} = A^{\text{in}}D^{-1}\mathbf{v}^{(t-1)}_{V\setminus C_1} + \left(A^{\text{out}}D^{-1}\mathbf{v}^{(t-1)}\right)_{V\setminus C_1}.$$

Clearly

$$\left\|\left(A^{\text{out}}D^{-1}\mathbf{v}^{(t-1)}\right)_{V\setminus C_1}\right\|_1 \leq \left\|A^{\text{out}}D^{-1}\mathbf{v}^{(t-1)}\right\|_1$$

and so $\|\mathbf{v}^{(t)}_{V\setminus C_1}\|_1 \leq$

$$\|A^{\text{in}}D^{-1}\mathbf{v}^{(t-1)}_{V\setminus C_1}\|_1 + \|A^{\text{out}}D^{-1}\mathbf{v}^{(t-1)}\|_1 \leq \|A^{\text{in}}D^{-1}\|_1\|\mathbf{v}^{(t-1)}_{V\setminus C_1}\|_1 + \|A^{\text{out}}D^{-1}\|_1\|\mathbf{v}^{(t-1)}\|_1$$

Moreover: $\|A^{\text{in}}D^{-1}\|_1 = \max_j \sum_i \frac{A^{\text{in}}_{ij}}{d_j} = \max_j \frac{d^{\text{in}}_j}{d_j} \leq 1$ and similarly $\|A^{\text{out}}D^{-1}\|_1 = \max_j \frac{d^{\text{out}}_j}{d_j} \leq \max_j r_j \leq \epsilon_2$ by assumption (A2). Thus $\|\mathbf{v}^{(t)}_{V\setminus C_1}\|_1 \leq 1\|\mathbf{v}^{(t-1)}_{V\setminus C_1}\|_1 + \epsilon_2\|\mathbf{v}^{(t-1)}\|_1$. Solving this recursion relation we obtain:

$$\|\mathbf{v}^{(t)}_{V\setminus C_1}\|_1 \leq \epsilon_2 \sum_{s=0}^{t-1} \|\mathbf{v}^{(s)}\|_1 + \|\mathbf{v}^{(0)}_{V\setminus C_1}\|_1$$

Because $\mathbf{v}^{(0)} = D\mathbf{1}_\Gamma$ and $\Gamma \subset C_1$, it follows that $\|\mathbf{v}^{(0)}_{V\setminus C_1}\|_1 = 0$ and $\|\mathbf{v}^{(0)}\|_1 = \text{vol}(\Gamma)$. Because $\|P\|_1 = 1$ it follows that $\|\mathbf{v}^{(s)}\|_1 = \|\mathbf{v}^{(0)}\|_1 = \text{vol}(\Gamma)$ for all s. Thus:

$$\sum_{j\in W} v^{(t)}_j \leq \|\mathbf{v}^{(t)}_{V\setminus C_1}\|_1 \leq t\epsilon_2\text{vol}(\Gamma) \leq t\epsilon_2(1+\epsilon_2)\text{vol}^{\text{in}}(\Gamma), \tag{B.3}$$

where the final inequality follows from Lemma B.1. Now let us put this all together. Returning to (B.1) with (B.2) and (B.3) in hand:

$$q^t_{\min}\left(\frac{\text{vol}^{\text{in}}(U)\text{vol}^{\text{in}}(\Gamma)}{\text{vol}^{\text{in}}(G_{C_1})} - \epsilon_1^t\sqrt{\text{vol}^{\text{in}}(U)\text{vol}^{\text{in}}(\Gamma)}\right) \leq t\epsilon_2(1+\epsilon_2)\text{vol}^{\text{in}}(\Gamma) \tag{B.4}$$

$$\Longrightarrow q^t_{\min}\left(\frac{\text{vol}^{\text{in}}(U)}{\text{vol}^{\text{in}}(G_{C_1})} - \epsilon_1^t\sqrt{\frac{\text{vol}^{\text{in}}(U)}{\text{vol}^{\text{in}}(\Gamma)}}\right) \leq t\epsilon_2(1+\epsilon_2). \tag{B.5}$$

From Lemma B.1 and the assumptions on $|U|$ and $|\Gamma|$:

$$\frac{\text{vol}^{\text{in}}(U)}{\text{vol}^{\text{in}}(G_{C_1})} \geq \frac{(1-\epsilon_3)d^{\text{in}}_{\text{av}}|U|}{(1+\epsilon_3)d^{\text{in}}_{\text{av}}|C_1|} = \frac{(1-\epsilon_3)un_1}{(1+\epsilon_3)n_1} = \frac{1-\epsilon_3}{1+\epsilon_3}u$$

$$\frac{\text{vol}^{\text{in}}(U)}{\text{vol}^{\text{in}}(\Gamma)} \leq \frac{(1+\epsilon_3)d^{\text{in}}_{\text{av}}|U|}{(1-\epsilon_3)d^{\text{in}}_{\text{av}}|\Gamma|} \leq \frac{(1+\epsilon_3)}{(1-\epsilon_3)}\frac{u}{g\epsilon_1^{2t-1}}$$

Finally because $q_i = 1 - r_i$ it follows that $q_{\min} \geq 1 - \epsilon_2$. Putting this all into equation (B.5):

$$(1-\epsilon_2)^t\left(\frac{1-\epsilon_3}{1+\epsilon_3}u - \epsilon_1^{1/2}\sqrt{\frac{(1+\epsilon_3)}{(1-\epsilon_3)}\frac{u}{g}}\right) \leq t\epsilon_2(1+\epsilon_2)$$

At this stage it is illuminating to use the assumption that $\epsilon_1, \epsilon_2, \epsilon_3 = o(1)$. Observe that:

$$\frac{1-o(1)}{1+o(1)} = 1-o(1), \quad \frac{1+o(1)}{1-o(1)} = 1+o(1), \quad \text{and } (1-o(1))^t = 1-o(1)$$

where the final equality follows as t is constant with respect to n. Hence:

$$(1-o(1))u - o(\sqrt{u}) \leq o(1) \implies u \leq o(1) + o(u).$$

This is only possible if $u = o(1)$. It follows that $|C_1 \triangle \Omega| = |U| + |W| = (\epsilon + 2u)n_1 = (\epsilon + o(1))n_1$ as stated. ∎

C. Showing the SBM satisfies our assumptions. Let us verify that SBM$(\mathbf{n}, P)$ satisfies the assumptions (A1)–(A4), under the hypotheses of Theorem 9.3. Recall that our assumption is $P_{ab} = (\beta + o(1)) \log(n)/n$ for $a \neq b$, and that $P_{aa} = \omega \log(n)/n_a$ for $a = 1, \ldots, k$. As we also assume that $n_1 = O(n) \to \infty$, and n_1 is the size of the smallest cluster, we get that $k = O(1)$, *i.e.* (A1) holds.

Theorem C.1 (see [7, 8]). *Let $G \sim ER(n, q)$ with $q = (\beta + o(1)) \log(n)/n$. There exist a function $\eta(\beta)$ satisfying $0 < \eta(\beta) < 1$ and $\lim_{\beta \to \infty} \eta(\beta) = 0$ such that*

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

Theorem C.2 (see [23], Theorem 3.4 (ii)). *If $G \sim ER(n_a, p)$ with $p_a = \omega \log(n)/n_a$ where $\omega \to \infty$, then $d_{\min}(G) = (1 - o(1))\omega \log(n)$ and $d_{\max}(G) = (1 + o(1))\omega \log(n)$ a.s.*

Theorem C.3. *Suppose that $G \sim ER(n_a, p)$ with $p = \omega \log(n)/n_a$ where $\omega \to \infty$. Then we have almost surely $|\lambda_i(L) - 1| = O(\omega^{-1/2}) = o(1)$ for all $i > 1$.*

Proof. Theorem 4 in [15] shows that

$$|\lambda_i(L^{\text{sym}}) - 1| \leq \sqrt{\frac{6 \log(2n_a)}{\omega \log(n)}}.$$

By Lemma 2.3 L^{sym} and L have the same spectrum. The result follows as $\log(n) \geq \log(n_a)$ ∎

As each $G_{C_a} \sim \text{ER}(n_a, p)$, it follows from Theorem C.3 that:

Corollary C.4. *SBM$(\mathbf{n}, P)$ with parameters as in Theorem 9.3 satisfies assumption (A2) with $\epsilon_1 = O(\omega^{-1/2})$.*

We now discuss the remaining two assumptions. Let G^{in} and G^{out} be as in §2. If $G \sim \text{SBM}(\mathbf{n}, P)$ then G^{in} consists of k disjoint Erdős - Rènyi graphs, $G_{C_a} \sim \text{ER}(n_a, p)$. The graph G^{out} is not an Erdős - Rènyi graph, as there is zero 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 C.2 and C.1:

Corollary C.5. *If $G \sim$ SBM$(\mathbf{n}, P)$ with parameters as in Theorem 9.3 then $d^{out}_{\max}(G) \leq 2\beta \log n + o(1)$ a.s.*

Proof. Consider G^{out} as a subgraph of $\widetilde{G^{\text{out}}} \sim \text{ER}(n, q)$ and apply Theorem C.1 ∎

Corollary C.6. *If $G \sim SBM(\mathbf{n}, P)$ with parameters as in Theorem 9.3, then $d^{in}_{\min}(G) \geq (1 - o(1))\omega \log(n)$ and $d^{in}_{\max}(G) \leq (1 + o(1))\omega \log(n)$ a.s.*

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

$$d^{\text{in}}_{\max}(G) = \max_i d_i^{\text{in}} = \max_a d_{\max}(G_{C_a})$$

By Theorem C.2, $d_{\max}(G_a) = (1+o(1))\omega \log(n)$ a.s. Note that the $d_{\max}(G_{C_a})$ are independent random variables, and since we are taking a maximum over $k = \mathcal{O}(1)$ of them, it follows that $\max_a d_{\max}(G_{C_a}) \leq (1 + o(1))\omega \log(n)$ a.s. too. The proof for $d^{\text{in}}_{\min}(G)$ is similar. ∎

Corollary C.7. *$SBM(\mathbf{n}, P)$ with parameters as in Theorem 9.3 satisfies assumption (A3) with $\epsilon_2 = O(\omega^{-1})$.*

Proof. First of all, it is clear that for any i, $d_i^{\text{out}}/d_i^{\text{in}} \leq d^{\text{out}}_{\max}/d^{\text{in}}_{\min}$. From Corollaries C.5 and C.6 we have:

$$\frac{d^{\text{out}}_{\max}}{d^{\text{in}}_{\min}} \leq \frac{2\beta \log n + o(1)}{(1 - o(1))\omega \log(n)} = \frac{2\beta + o(1)}{(1 - o(1))\omega} = O(\omega^{-1}).$$ ∎

Corollary C.8. *$SBM(\mathbf{n}, P)$ with parameters as in Theorem 9.3 satisfies assumption (A4).*

Proof. Observe that $d^{\text{in}}_{\text{av}} = \omega \log(n)$. The result then follows from Corollary C.6. ∎