2018 IEEE 59th Annual Symposium on Foundations of Computer Science

Testing Graph Clusterability: Algorithms and Lower Bounds

Ashish Chiplunkar*, Michael Kapralov*, Sanjeev Khanna†, Aida Mousavifar* and Yuval Peres‡
*EPFL, Lausanne, Switzerland
†University of Pennsylvania, Philadelphia, USA
‡Microsoft Research, Redmond, USA

***Abstract*—We consider the problem of testing graph cluster structure: given access to a graph $G = (V, E)$, can we quickly determine whether the graph can be partitioned into a few clusters with good inner conductance, or is far from any such graph? This is a generalization of the well-studied problem of testing graph expansion, where one wants to distinguish between the graph having good expansion (i.e. being a good single cluster) and the graph having a sparse cut (i.e. being a union of at least two clusters). A recent work of Czumaj, Peng, and Sohler (STOC'15) gave an ingenious sublinear time algorithm for testing k-clusterability in time $\tilde{O}(n^{1/2}\mathbf{poly}(k))$. Their algorithm implicitly embeds a random sample of vertices of the graph into Euclidean space, and then clusters the samples based on estimates of Euclidean distances between the points. This yields a very efficient testing algorithm, but only works if the cluster structure is very strong: it is necessary to assume that the gap between conductances of accepted and rejected graphs is at least logarithmic in the size of the graph G. In this paper we show how one can leverage more refined geometric information, namely angles as opposed to distances, to obtain a sublinear time tester that works even when the gap is a sufficiently large constant. Our tester is based on the singular value decomposition of a natural matrix derived from random walk transition probabilities from a small sample of seed nodes.**

We complement our algorithm with a matching lower bound on the query complexity of testing clusterability. Our lower bound is based on a novel property testing problem, which we analyze using Fourier analytic tools. As a byproduct of our techniques, we also achieve new lower bounds for the problem of approximating MAX-CUT value in sublinear time.

***Keywords*-clustering; property testing; sublinear algorithms; spectral graph theory**

I. Introduction

Graph clustering is the problem of partitioning vertices of a graph based on the connectivity structure of the graph. It is a fundamental problem in many application domains where one wishes to identify groups of closely related objects, for instance, communities in a social network. The clustering problem is, thus, to partition a graph into vertex-disjoint subgraphs, namely clusters, such that each cluster contains vertices that are more similar to each other than the rest of the graph. There are many natural measures that have been proposed to assess the quality of a cluster; one particularly well-studied and well-motivated measure for graph clustering is conductance of a cluster [21]. Roughly speaking, conductance of a graph measures the strength of connections across any partition of vertices relative to the strength of connections inside the smaller of the two parts. The higher the conductance inside a cluster, the harder it is to split it into non-trivial pieces. The conductance measure lends itself to a natural graph clustering objective, namely, partition the vertices of a graph into a small number of clusters such that each cluster has large conductance in the graph induced by it (the inner conductance of the cluster). Towards this objective, many efficient graph partitioning algorithms have been developed that partition vertices of a graph into a specified number of clusters with approximately high conductance (when possible). Any algorithm that outputs such a partition necessarily requires $\Omega(n)$ time – simply to output the solution, and usually $\Omega(m)$ time, where n and m respectively denote the number of vertices and edges in the input graph. On very large-scale graphs, even linear-time algorithms may prove to be computationally prohibitive, and consequently, there has been considerable recent interest in understanding the cluster structure of a graph in sublinear time. Specifically, given a target number of clusters, say k, and a measure ϕ of desired cluster quality, how much exploration of the input graph is needed to distinguish between graphs that can be partitioned into at most k clusters with inner conductance at least ϕ from graphs that are far from admitting such clustering? The focus of this paper is to understand the power of sublinear algorithms in discovering the cluster structure of a graph.

In our study, we use by now a standard model of graph exploration for sublinear algorithms, where at any step, the algorithm can either sample a uniformly at random vertex, ask the degree $d(u)$ of a vertex u, or specify a pair (u, i) and recover the i^{th} neighbor of u for any $i \in [1..d(u)]$. Each such operation is called a *query*. For any positive $\epsilon > 0$, we say a pair of graphs is ϵ-far if one needs to modify at least an ϵ-fraction of edges to convert one graph into another. The simplest form of the cluster structure problem is the case $k = 1$: how many queries to the graph are needed to distinguish between

2575-8454/18/$31.00 ©2018 IEEE
DOI 10.1109/FOCS.2018.00054

graphs that are expanders (YES case) from graphs that are $\Omega(1)$-far from being expanders (NO case)? A formal study of this basic question was initiated in the work of Goldreich and Ron [17] where they showed that even on bounded degree graphs, $\Omega(\sqrt{n})$ queries to the input graph are necessary to distinguish between expanders and graphs that are far from expanders. On the positive side, it is known that a bounded degree expander graph with conductance at least ϕ can be distinguished from a graph that is $\Omega(1)$-far from a graph with conductance $\gamma \times \phi^2$ for some positive constant γ, using only $n^{\frac{1}{2}+O(\gamma)}$ queries [20], [24]. Thus even the simplest setting of the graph clustering problem is not completely understood – the known algorithmic results require additional separation in the conductance requirements of YES and NO instances. Furthermore, even with this separation in conductance requirements, best algorithmic result require polynomially more queries than suggested by the lower bound.

Lifting algorithmic results above for the case $k = 1$ to larger values of k turned out to be a challenging task. A breakthrough was made by Czumaj, Peng, and Sohler [6], who designed an algorithm that differentiates between bounded degree graphs that can be clustered into k clusters with good inner conductance (YES case) from graphs that are far from such graphs (NO case), using only $\tilde{O}(n^{\frac{1}{2}}\text{poly}(k))$ queries. This striking progress, however, required an even stronger separation between YES and NO instances of the problem. In particular, the algorithm requires that in the YES case, the graph can be partitioned into k clusters with inner conductance at least ϕ, while in the NO case, the graph is ϵ-far from admitting k clusters with conductance $\phi^2/\log n$. Thus the cluster quality in the NO case needs to be weakened by a factor that now depends on the size of the input graph.

The current state of the art raises several natural questions on both algorithmic and lower bound fronts. On the algorithmic front, does sublinear testing of cluster structure of a graph fundamentally require such strong separation between the cluster structures of YES and NO cases? On the lower bound front, is there a stronger barrier than the current $\Omega(\sqrt{n})$ threshold for differentiating between the YES and NO cases? Even for the case of distinguishing an expander for a graph that is far from expander, the known algorithmic results require $n^{\frac{1}{2}+\Omega(1)}$ queries when the conductance guarantees of YES and NO cases are separated by only a constant factor.

In this work, we make progress on both questions above. On the algorithmic side, we present a new sublinear testing algorithm that considerably weakens the separation required between the conductance of YES and NO instances. In particular, for any fixed k, our algorithm can distinguish between instances that can be partitioned into k clusters with conductance at least ϕ from instances that are $\Omega(1)$-far from admitting k clusters with conductance $\gamma\phi^2$, using $n^{\frac{1}{2}+O(\gamma)}$ queries. This generalizes the results of [20], [24] for $k = 1$ to any fixed k and arbitrary graphs. Similar to [6] our algorithm is based on sampling a small number of vertices and gathering information about the transition probabilities of suitably long random walks from the sampled points. However, instead of classifying points as pairwise similar or dissimilar based on ℓ_2 similarity between the transition probability vectors, our approach is based on analyzing the structure of the Gram matrix of these transition probability vectors, which turns out to be a more robust mechanism for separating the YES and NO cases.

On the lower bound side, we show that arguably the simplest question in this setting, namely, differentiating a bounded degree expander graph with conductance $\Omega(1)$ from a graph that is $\Omega(1)$-far from a graph with conductance γ for some positive constant γ, already requires $n^{\frac{1}{2}+\Omega(\gamma)}$ queries. This improves upon the long-standing previous lower bound of $\Omega(n^{\frac{1}{2}})$. Going past the $n^{\frac{1}{2}}$ threshold requires us to introduce new ideas to handle non-trivial dependencies that manifest due to unavoidable emergence of cycles once an $\omega(n^{\frac{1}{2}})$-sized component is uncovered in an expander. We use a Fourier analytic approach to handle emergence of cycles and create a distribution where $n^{\frac{1}{2}+\Omega(\gamma)}$ queries are necessary to distinguish between YES and NO cases. We believe our lower bound techniques are of independent interest and will quite likely find applications to other problems. As one illustrative application, we show that our approach yields an $n^{\frac{1}{2}+\Omega(1)}$ query complexity lower bound for the problem of approximating the max-cut value in graph to within a factor better than 2, improving the previous best lower bound of $\Omega(n^{\frac{1}{2}})$.

In what follows, we formally define our clustering problem, present our main results, and give an overview of our techniques.

A. Problem Statement

We start by introducing basic definitions, then proceed to define the problems that we design algorithms for (namely **PartitionTesting** and testing clusterability) in Section I-B, and finally discuss the communication game that we use to derive query complexity lower bounds (namely the **NoisyParities** game) and state our results on lower bounds in Section I-C.

Definition 1 (Internal and external conductance). Let $G = (V_G, E_G)$ be a graph. Let $\deg(v)$ be the degree of vertex v. For a set $S \subseteq V_G$, let $\text{vol}(S) = \sum_{v \in S} \deg(v)$ denote the *volume* of set S. For a set $S \subseteq C \subseteq V_G$, the *conductance of S within C*, denoted by $\phi_C^G(S)$, is the

number of edges with one endpoint in S and the other in $C \setminus S$ divided by $\mathrm{vol}(S)$. Equivalently, $\phi_C^G(S)$ is the probability that a uniformly random neighbor, of a vertex in S selected with probability proportional to degree, is in $C \setminus S$. The *internal conductance* of C, denoted by $\phi^G(C)$, is defined to be $\min_{S \subseteq C, 0 < \mathrm{vol}(S) \leq \frac{\mathrm{vol}(C)}{2}} \phi_C^G(S)$ if $|C| > 1$ and one otherwise. The *external conductance* of C is defined to be $\phi_{V_G}^G(C)$.

Based on the conductance parameters, clusterability and unclusterability of graphs is defined as follows.

Definition 2 (Graph clusterability). Graph $G = (V_G, E_G)$ is defined to be (k, φ)-*clusterable* if V_G can be partitioned into $C_1, \ldots, C_h$ for some $h \leq k$ such that for all $i = 1, \ldots, h$, $\phi^G(C_i) \geq \varphi$. Graph G is defined to be (k, φ, β)-*unclusterable* if V_G contains $k+1$ pairwise disjoint subsets $C_1, \ldots, C_{k+1}$ such that for all $i = 1, \ldots, k+1$, $\mathrm{vol}(C_i) \geq \beta \cdot \frac{\mathrm{vol}(V_G)}{k+1}$, and $\phi_{V_G}^G(C_i) \leq \varphi$.

The following algorithmic problem was implicitly defined in [6]:

Definition 3. PartitionTesting$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$ is the problem of distinguishing between the following two types of graphs.

1) The YES case: graphs which are (k, φ_{in})-clusterable
2) The NO case: graphs which are $(k, \varphi_{\text{out}}, \beta)$-unclusterable

The ultimate problem that we would like to solve is the **Clusterability** problem, defined below:

Definition 4. Clusterability$(k, \varphi, k', \varphi', \varepsilon)$ is the problem of distinguishing between the following two types of graphs.

1) The YES case: graphs which are (k, φ)-clusterable
2) The NO case: graphs which are ε-far from (k', φ')-clusterable.

Here, a graph $G = (V, E)$ is ε-far from (k', φ')-clusterable if there does not exist a (k', φ')-clusterable graph $G' = (V, E')$ such that $|E \oplus E'| \leq \varepsilon \cdot |E|$ (here $\oplus$ denotes the symmetric difference between sets).

Note that in the clusterability problem considered by Czumaj et al. [6], the YES instances were required to have clusters with small outer conductance, whereas we have no such requirement.

Queries and Complexity. We assume that the algorithm has access to the input graph G via the following queries.

1) Vertex query: returns a uniformly random vertex $v \in V_G$
2) Degree query: outputs degree $\deg(v)$ of a given $v \in V_G$.
3) Neighbor query: given a vertex $v \in V_G$, and $i \in [n]$, returns the i-th neighbor of v if $i \leq \deg(v)$, and returns *fail* otherwise.

The complexity of the algorithm is measured by number of access queries.

B. Algorithmic Results

Theorem 1. *Suppose $\varphi_{out} \leq \frac{1}{480}\varphi_{in}^2$. Then there exists a randomized algorithm for* **PartitionTesting**$(k, \varphi_{in}, \varphi_{out}, \beta)$ *which gives the correct answer with probability at least $2/3$, and which makes $\mathrm{poly}(1/\varphi_{in}) \cdot \mathrm{poly}(k) \cdot \mathrm{poly}(1/\beta) \cdot \mathrm{poly}\log(m) \cdot m^{1/2+O(\varphi_{out}/\varphi_{in}^2)}$ queries on graphs with m edges.*

Observe that even when the *average* degree of the vertices of the graph is constant, the dependence of query complexity on n, the number of vertices, is $\tilde{O}(n^{1/2+O(\varphi_{\text{out}}/\varphi_{\text{in}}^2)})$. We also note that our current analysis of the tester is probably somewhat loose: the tester likely requires no more than $\tilde{O}(n^{1/2+O(\varphi_{\text{out}}/\varphi_{\text{in}}^2)})$ for graphs of arbitrary volume.

Theorem 1 allows us to obtain the following result on testing clusterability, which removes the logarithmic gap assumption required for the results in [7] in the property testing framework.

Theorem 2. *Suppose $\varphi' \leq \alpha_{4.5}\varepsilon$, (for the constant $\alpha_{4.5} = \Theta(\min(d^{-1}, k^{-1}))$ from Lemma 4.5 of [7], where d denotes the maximum degree), and $\varphi' \leq c'\varepsilon^2\varphi^2/k^2$ for some small constant c'. Then there exists a randomized algorithm for* **Clusterability**$(k, \varphi, k, \varphi', \varepsilon)$ *problem on degree d-bounded graphs that gives the correct answer with probability at least $2/3$, and which makes $\mathrm{poly}(1/\varphi) \cdot \mathrm{poly}(k) \cdot \mathrm{poly}(1/\varepsilon) \cdot \mathrm{poly}(d) \cdot \mathrm{poly}\log(n) \cdot n^{1/2+O(\varepsilon^{-2}k^2 \cdot \varphi'/\varphi^2)}$ queries on graphs with n vertices.*

The proof of the theorem follows by combining Theorem 1 and Lemma 4.5 of [7].

Furthermore, we strengthen Lemma 4.5 of [7] to reduce the dependence of the gap between inner and outer conductance to logarithmic in k, albeit at the expense of a bicriteria approximation. This gives us the following theorem.

Theorem 3. *Let $0 \leq \varepsilon \leq \frac{1}{2}$. Suppose $\varphi' \leq \alpha$, (for $\alpha = \min\{\frac{c_{exp}}{150d}, \frac{c_{exp} \cdot \varepsilon}{1400 \log(\frac{16k}{\varepsilon})}\}$, where d denotes the maximum degree), and $\varphi' \leq c \cdot \varepsilon^2\varphi^2/\log(\frac{32k}{\varepsilon})$ for some small constant c. Then there exists a randomized algorithm for* **Clusterability**$(k, \varphi, 2k, \varphi', \varepsilon)$ *problem on degree d-bounded graphs that gives the correct answer with probability at least $2/3$, and which makes $\mathrm{poly}(1/\varphi) \cdot \mathrm{poly}(k) \cdot \mathrm{poly}(1/\varepsilon) \cdot \mathrm{poly}(d) \cdot \mathrm{poly}\log(n) \cdot$*

$n^{1/2+O(\varepsilon^{-2}\log(\frac{32k}{\varepsilon})\cdot\varphi'/\varphi^2)}$ *queries on graphs with* n *vertices.*

C. Lower bound Results

Our lower bounds are based on the following communication problem that we refer to as the **NoisyParities**(d, ε):

Definition 5. NoisyParities(d, ε) is the problem with parameters $d \geq 3$ and $\varepsilon \leq 1/2$ defined as follows. An adversary samples a random d-regular graph $G = (V, E)$ from the distribution induced by the *configuration model* of Bollobás [2]. The adversary chooses to be in the YES case or the NO case with probability $1/2$, and generates a vector of binary edge labels $Y \in \{0,1\}^E$ as follows:

1) The YES case: The vector Y is chosen uniformly at random from $\{0,1\}^E$, that is, the labels $Y(e)$ for all edges $e \in E$ are independently 0 or 1 with probability $1/2$;
2) The NO case: A vector $X \in \{0,1\}^V$ is sampled uniformly at random. Independently, a "noise" vector $Z \in \{0,1\}^E$ is sampled such that all the $Z(e)$'s are independent Bernoulli random variables which are 1 with probability ε and 0 with probability $1-\varepsilon$. The label of an edge $e = (u, v) \in E$ is given by $Y(e) = X(u) + X(v) + Z(e)$.

The algorithm can query vertices $q \in V$ in an adaptive manner deterministically. Upon querying a vertex $q \in V$, the algorithm gets the edges incident on q together with their labels as a response to the query, and must ultimately determine whether the adversary was in the YES or the NO case.

Our main result is the following lower bound on the query complexity of **NoisyParities**, whose proof is deferred to the full version of the paper [4].

Theorem 4. *Any deterministic algorithm that solves the* **NoisyParities** *problem correctly with probability at least* $2/3$ *must make at least* $n^{1/2+\Omega(\varepsilon)}$ *queries on* n*-vertex graphs, for constant* d*.*

We remark that this lower bound is tight up to constant factors multiplying ε in the exponent. As a consequence of Theorem 4 and an appropriate reductions, we derive the following lower bound on the query complexity of **PartitionTesting**.

Theorem 5. *Any algorithm that distinguishes between a* $(1, \varphi_{in})$*-clusterable graph (that is, a* φ_{in}*-expander) and a* $(2, \varphi_{out}, 1)$*-unclusterable graph on* n *vertices (in other words, solves* **PartitionTesting**$(1, \varphi_{in}, \varphi_{out}, 1)$*) correctly with probability at least* $2/3$ *must make at least* $n^{1/2+\Omega(\varphi_{out})}$ *queries, even when the input is restricted to regular graphs, for constant* φ_{in}*.*

Moreover, as a by-product of Theorem 4, we also get the following result, whose proof is deferred to the full version [4].

Theorem 6. *Any algorithm that approximates the MAX-CUT of* n*-vertex graphs within a factor* $2-\varepsilon'$ *with probability at least* $2/3$ *must make at least* $n^{1/2+\Omega(\varepsilon'/\log(1/\varepsilon'))}$ *queries.*

D. Our techniques

In this section we give an overview of the new techniques involved in our algorithm and lower bounds.

1) Algorithms: We start by giving an outline the approach of [6], outline the major challenges in designing robust tester of graph cluster structure, and then describe our approach.

As [6] show, the task of distinguishing between (k, φ)-clusterable graphs and graphs that are ϵ-far from (k, φ')-clusterable reduces to **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$, where $\beta = \text{poly}(\epsilon)$. In this problem we are given query access to a graph G, and would like to distinguish between two cases: either the graph can be partitioned into at most k clusters with inner conductance at least φ_{in} (the **YES** case, or 'clusterable' graphs) or there exists at least $k+1$ subsets $C_1, \ldots, C_{k+1}$ with outer conductance at most φ_{out} and containing nontrivial (i.e. no smaller than $\beta n/(k+1)$) number of nodes (the **NO** case, or 'non-clusterable' graphs). Here $\varphi_{\text{in}} = \varphi$, and φ_{out} is a function of the conductance φ', the number of nodes k, and the precision parameter ϵ.

A very natural approach to **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$ is to sample $10k$ nodes, say, run random walks of appropriate length from the sampled nodes, and compare the resulting distributions: if a pair of nodes is in the same cluster, then the distributions of random walks should be 'close', and if the nodes are in different clusters, the distributions of random walks should be 'far'. The work of [6] shows that this high level approach can indeed be made to work: if one compares distributions in ℓ_2 norm, then for an appropriate separation between φ_{in} and φ_{out} random walks whose distributions are closer than a threshold θ in ℓ_2 sense will indicate that the starting nodes are in the same cluster, and if the distributions are further than 2θ apart in ℓ_2, say, then the starting points must have been in different clusters. Using an ingenious analysis [6] show that one can construct a graph on the sampled nodes where 'close' nodes are connected by an edge, and the original graph is clusterable if and only if the graph on the sampled nodes is a union of at most k connected components. The question of estimating ℓ_2 norm distance between distributions remains, but this can be done in about $\sqrt{n}$ time by estimating collision probabilities (by

the birthday paradox), or by using existing results in the literature. The right threshold θ turns out to be $\approx 1/\sqrt{n}$.

The main challenge: While very beautiful, the above approach unfortunately does not work unless the cluster structure in our instances is very pronounced. Specifically, the analysis of [6] is based on arguing that random walks of $O(\log n)$ length from sampled nodes that come from the same cluster mostly don't leave the cluster, and this is true only if the outer conductance of the cluster is no larger than $1/\log n$. This makes the approach unsuitable for handling gaps between conductances that are smaller than $\log n$ (it is not hard to see that the walk length must be at least logarithmic in the size of the input graph, so shortening the walk will not help).

One could think that this is a question of designing of a more refined analysis of the algorithm of [6], but the problem is deeper: it is, in general, not possible to choose a threshold θ that will work even if the gap between conductances is constant, and even if we want to distinguish between 2-clusterable and far from 2-clusterable graphs (such a choice is, in fact, possible for $k = 1$). To summarize, euclidean distance between distributions is no longer a reliable metric if one would like to operate in a regime close to theoretical optimum, and a new proxy for clusterability is needed. (See the full version [4] for a more elaborate discussion.)

Our main algorithmic ideas: Our main algorithmic contribution is a more geometric approach to analyzing the proximity of the sampled points: instead of comparing ℓ_2 distances between points, our tester considers the Gram matrix of the random walk transition probabilities of the points, estimates this matrix entry-wise to a precision that depends on the gap between φ_{in} and φ_{out} in the instance of **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$ that we would like to solve, and computes the $(k+1)$-st largest eigenvalue of the matrix. This quantity turns out to be a more robust metric, yielding a tester that operates close to the theoretical optimum, i.e. able to solve **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$ as long as the gap $\varphi_{\text{out}}/\varphi_{\text{in}}^2$ is smaller than an absolute constant.[1] Specifically, our tester (see Algorithms 1 and 2 in Section II-A for the most basic version) samples a multiset S of $s \approx \text{poly}(k)\log n$ vertices of the graph G independently and with probability proportional to the degree distribution (this can be achieved in $\approx \sqrt{n}$ time per sample using the result of Eden and Rosenbaum [13]), and computes the matrix

$$A := (D^{-\frac{1}{2}} M^t S)^\top (D^{-\frac{1}{2}} M^t S), \tag{1}$$

[1]Note that our runtime depends on $\varphi_{\text{out}}/\varphi_{\text{in}}^2$ as opposed to $\varphi_{\text{out}}/\varphi_{\text{in}}$ due to a loss in parameters incurred through Cheeger's inequality. This loss is quite common for spectral algorithms.

where M is the random walk transition matrix of the graph G, and D is the diagonal matrix of degrees. Note that this is the Gram matrix of the distributions of the endpoints of random walks from the sampled nodes in G, where each walk has a logarithmic number of steps. Intuitively, the matrix A captures pairwise collision probabilities of random walks from sampled nodes, weighted by inverse degree. The algorithm accepts the graph if the $(k+1)$-st largest eigenvalue of the matrix A is below a threshold, and rejects otherwise. Specifically, the algorithm accepts if $\mu_{k+1}(A) \lesssim \text{vol}(V_G)^{-1-\Theta(\varphi_{\text{out}}/\varphi_{\text{in}}^2)}$ and rejects otherwise. Before outlining the proof of correctness for the tester, we note that, of course, the tester above cannot be directly implemented in sublinear time, as computing the matrix A exactly is expensive. The actual sublinear time tester approximately computes the entries of the matrix A to additive precision about $\frac{1}{\text{poly}(k)}\text{vol}(V_G)^{-1-\Theta(\varphi_{\text{out}}/\varphi_{\text{in}}^2)}$ and uses the eigenvalues of the approximately computed matrix to decide whether to accept or reject. Such an approximation can be computed in about $\text{vol}(V_G)^{\frac{1}{2}+\Theta(\varphi_{\text{out}}/\varphi_{\text{in}}^2)}$ queries by rather standard techniques (see Section II-B).

We now outline the proof of correctness of the tester above (the detailed proof is presented in Section II-A). It turns out to be not too hard to show that the tester accepts graphs that are (k, φ_{in})-clusterable. One first observes that Cheeger's inequality together with the assumption that each of the k clusters is a φ_{in}-expander implies that the $(k+1)$-st eigenvalue of the normalized Laplacian of G is at least $\varphi_{\text{in}}^2/2$. It follows that the matrix M^t of t-step random walk transition probabilities, for our choice of $t = (c/\varphi_{\text{in}}^2)\log n$, is very close to a matrix of rank at most k, and thus the $(k+1)$-st eigenvalue of the matrix A above (see (1)) is smaller than $1/n^2$, say. The challenging part is to show that the tester rejects graphs that are $(k, \varphi_{\text{out}}, \beta)$-unclusterable, since in this case we do not have any assumptions on the inner structure of the clusters $C_1, \ldots, C_{k+1}$. The clusters $C_1, \ldots, C_{k+1}$ could either be good expanders, or, for instance, unions of small disconnected components. The random walks from nodes in those clusters behave very differently in these two cases, but the analysis needs to handle both. Our main idea is to consider a carefully defined $k+1$-dimensional subspace of the eigenspace of the normalized Laplacian of G that corresponds to small (smaller than $O(\varphi_{\text{out}})$) eigenvalues, and show that our random sample of points is likely to have a well-concentrated projection onto this subspace. We then show that this fact implies that the matrix A in (1) has a large $(k+1)$-st eigenvalue with high probability. The assumption that vertices in S are sampled with probabilities proportional to their degrees is crucial to making the proof work for general (sparse)

graphs.

One consequence of the fact that our algorithm for **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$ estimates the entries of the Gram matrix referred to above to additive precision $\approx n^{-1-\Theta(\varphi_{\text{out}}/\varphi_{\text{in}}^2)}$ is that the runtime $\approx n^{1/2+\Theta(\varphi_{\text{out}}/\varphi_{\text{in}}^2)}$. If $\varphi_{\text{out}} \ll \varphi_{\text{in}}^2/\log n$, then we recover the $\approx \sqrt{n}$ runtime of [6], but for any constant gap between φ_{out} and φ_{in}^2 our runtime is polynomially larger than $\sqrt{n}$. Our main contribution on the lower bound side is to show that this dependence is necessary. We outline our main ideas in that part of the paper now.

2) The lower bound: We show that the $n^{1+\Omega(\varphi_{\text{out}}/\varphi_{\text{in}}^2)}$ runtime is necessary for **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$ problem, thereby proving that our runtime is essentially best possible for constant k. More precisely, we show that even distinguishing between an expander and a graph that contains a cut of sparsity ϵ for $\epsilon \in (0, 1/2)$ requires $n^{1+\Omega(\epsilon)}$ adaptive queries, giving a lower bound for the query complexity (and hence runtime) of **PartitionTesting**$(1, \Omega(1), \epsilon, 1)$ that matches our algorithm's performance.

The **NoisyParities** *problem:* Our main tool in proving the lower bound is a new communication complexity problem (the **NoisyParities** problem) that we define and analyze: an adversary chooses a regular graph $G = (V, E)$ and a hidden binary string $X \in \{0,1\}^V$, which can be thought of as encoding a hidden bipartition of G. The algorithm can repeatedly (and adaptively) query vertices of G. Upon querying a vertex v, the algorithm receives the edges incident on v and a binary label $Y(e)$ on each edge e. In the **NO** case the labels $Y(e)$ satisfy $Y(e) = X(u) + X(v) + Z(e)$, where $Z(e)$ is an independent Bernoulli random variable with expectation ϵ (i.e. the algorithm is told whether the edge crosses the hidden bipartition, but the answer is noisy). In the **YES** case each label $Y(e)$ is uniformly random in $\{0,1\}$. The task of the algorithm is to distinguish between the two cases using the smallest possible number of queries to the graph G.

It is easy to see that if $\epsilon = 0$, then the algorithm can get a constant advantage over random guessing as long as it can query all edges along a cycle in G. If G is a random d-regular graph unknown to the algorithm, one can show that this will take at least $\Omega(\sqrt{n})$ queries, recovering the lower bound for expansion testing due to Goldreich and Ron [17]. In the noisy setting, however, detecting a single cycle is not enough, as cycles that the algorithm can locate in a random regular graph using few queries are generally of logarithmic length, and the noise added to each edge compounds over the length of the cycle, leading to only advantage of about $n^{-O(\epsilon)}$ over random guessing that one can obtain from a single cycle. Intuitively. this suggests that the algorithm should find at least $n^{\Omega(\epsilon)}$ cycles in order to get a constant advantage. Detecting a single cycle in an unknown sparse random graphs requires about $\sqrt{n}$ queries, which leads to the $n^{1/2+\Omega(\epsilon)}$ lower bound. Turning this intuition into a proof is challenging, however, as **(a)** the algorithm may base its decisions on labels that it observes on its adaptively queried subgraph of G and **(b)** the algorithm does not have to base its decision on observed parities over cycles. We circumvent these difficulties by analyzing the distribution of labels on the edges of the subgraph that the algorithm queries in the **NO** case and proving that this distribution is close to uniformly random in total variation distance, with high probability over the queries of the algorithm. We analyze this distribution using a Fourier analytic approach, which we outline now.

Suppose that we are in the **NO** case, i.e. the edge labels presented to the algorithm are an ϵ-noisy version of parities of the hidden boolean vector $X \in \{0,1\}^n$, and suppose that the algorithm has discovered a subset $E_{\text{query}} \subseteq E_G$ of edges of the graph G (recall that the graph G, crucially, is not known to the algorithm) together with their labels. The central question that our analysis needs to answer in this situation turns out to be the following: given the observed labels on edges in E_{query} and an edge $e = (a, b) \in E_G$ what is the posterior distribution of $X(a) + X(b)$ given the information that the algorithm observed so far? For example, if E_{query} does not contain any cycles (i.e. is a forest), then $X(a) + X(b)$ is a uniformly random Bernoulli variable with expectation $1/2$ if the edge (a, b) does not close a cycle when added to E_{query}. If it does close a cycle but E_{query} is still a forest, then one can show that if the distance in E_{query} from a to b is large (at least $\Omega(\log n)$), then the posterior distribution of $X(a) + X(b)$ is still $n^{-\Omega(\epsilon)}$ close, in total variation distance, to a Bernoulli random variable with expectation $1/2$. Our analysis needs to upper bound this distance to uniformity for a 'typical' subset E_{query} that arises throughout the interaction process of the algorithm with the adversary, and contains two main ideas. First, we show using Fourier analytic tools that for a 'typical' subset of queried edges E_{query} and any setting of observed labels, one has that the bias of $X(a)+X(b)$, i.e. the absolute deviation of the expectation of this Bernoulli random variable from $1/2$, satisfies

$$\text{bias}(X(a)+X(b)) \lesssim \sum_{\substack{E' \subseteq E_{\text{query}} \text{ s.t.} \\ E' \cup \{a,b\} \text{ is Eulerian}}} (1-2\epsilon)^{|E'|}. \quad (2)$$

Note that for the special case of E_{query} being a tree, the right hand side is exactly the $(1-2\epsilon)^{\text{dist}(a,b)}$, where $\text{dist}(a, b)$ stands for the shortest path distance from a to b in T. Since 'typical' cycles that the algorithm will

discover will be of $\Omega(\log n)$ length due to the fact that G is a constant degree random regular graph, this is $n^{-\Omega(\epsilon)}$, as required. Of course, the main challenge in proving our lower bound is to analyze settings where the set of queried edges E_{query} is quite far from being a tree, and generally contains many cycles, and control the sum in (2). In other words, we need to bound the weight distribution of Eulerian subgraphs of E_{query}. The main insight here is the following structural claim about 'typical' sets of queried edges E_{query}: we show that for typical interaction scenarios between the algorithm and the adversary one can decompose E_{query} as $E_{\text{query}} = F \cup R$, where F is a forest and R is a small (about $n^{O(\epsilon)}$ size) set of 'off-forest' edges that further satisfies the property that the endpoints of edges in R are $\Omega(\log n)$-far from each other in the shortest path metric induced by F. This analysis relies on basic properties of random graphs with constant degrees. Once such a decomposition of $E_{\text{query}} = T \cup F$ is established, we get a convenient basis for the cycle space of E_{query}, which lets us control the right hand side in (2) as required.

Finally, our lower bound on the query complexity of **NoisyParities** yields a lower bound for **PartitionTesting**$(1, \Omega(1), \epsilon, 1)$ (Theorem 5), as well as a lower bound for better than factor 2 approximation to MAX-CUT value in sublinear time (Theorem 6). The reduction to MAX-CUT follows using rather standard techniques (e.g. is very similar to [22]). The reduction to **PartitionTesting**$(1, \Omega(1), \epsilon, 1)$ is more delicate and novel: the difficulty is that we need to ensure that the introduction of random noise Z_e on the edge labels produces graphs that have the expansion property (in contrast, the MAX-CUT reduction produces graphs with a linear fraction of isolated nodes).

E. Related Work

Goldreich and Ron [17] initiated the framework of testing graph properties via neighborhood queries. In this framework, the goal is to separate graphs having a certain property from graphs which are "far" from having that property, in the sense that they need many edge additions and deletions to satisfy the property. The line of work closest to this paper is the one on testing expansion of graphs [16], [24], [8], [20] which proves that expansion testing can be done in about $\tilde{O}(\sqrt{n})$ queries, and $\Omega(\sqrt{n})$ queries are indeed necessary. Going beyond expansion (that is, 1-clusterability), Kannan et al. [21] introduced (internal) conductance as a measure of how well a set of vertices form a cluster. In order to measure the quality of a clustering, that is, a partition of vertices into clusters, Zhu et al. [1] and Oveis Gharan and Trevisan [14] proposed bi-criteria measures which take into account the (minimum) internal conductance and the (maximum) external conductance of the clusters. Considering this measure, Czumaj et al. [6] defined the notion of clusterable graphs parameterized by requirements on the minimum internal expansion and the maximum external expansion, and gave an algorithm for testing clusterability.

There has been an extensive work on testing many other graph properties in the framework of Goldreich and Ron. For instance, Czumaj et al. [5] give algorithms for testing several properties including cycle-freeness, whereas Eden et al. [10] design algorithms to test arboricity. Estimation of graph parameters such as degree distribution moments [12], number of triangles [9], and more generally, number of k-cliques [11] has also received attention recently.

A closely related model of property testing is the one where the graph arrives as a random order stream and the property testing algorithm is required to use sublinear space. Although this appears to be a less powerful model because the algorithm no longer has the ability to execute whatever queries it wants, interestingly, Peng and Sohler [25] show that sublinear property testing algorithms give rise to sublinear space algorithms for random order streams.

Other graph property testing models include extension to dense graphs [19], [18] where the algorithm queries the entries of the adjacency matrix of the graph, and the non-deterministic property testing model [23], [15], where the algorithm queries the graph and a certificate, and must decide whether the graph satisfies the property. We refer the reader to [6] for a more comprehensive survey of the related work.

II. Algorithm for **PartitionTesting**

The goal of this section is to present an algorithm for the **PartitionTesting** problem, analyze it, and hence, prove Theorem 1. We start by setting up some notation. Let $G = (V_G, E_G)$ be a graph and let A be its adjacency matrix.

Definition 6. The *normalized adjacency matrix* $\overline{A}$ of G is $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where D is the diagonal matrix of the degrees. The *normalized Laplacian* of G is $L = I - \overline{A}$.

Definition 7. The *random walk associated with* G is defined to be the random walk with transition matrix $M = \frac{I + AD^{-1}}{2}$. Equivalently, from any vertex v, this random walk takes every edge of G incident on v with probability $\frac{1}{2 \cdot \deg(v)}$, and stays on v with probability $\frac{1}{2}$.

We will use the following notation.

- For a vertex $a \in V_G$, $\mathbb{1}_a \in \mathbb{R}^{V_G}$ denotes the indicator of a, that is, the vector which is 1 at a and 0 elsewhere. Fix some total order on V_G. For a (multi) set $S = \{a_1, \ldots, a_s\}$ of vertices from V_G where $a_1, \ldots, a_s$ are sorted, we abuse notation

and also denote by S the $V_G \times s$ matrix whose i^{th} column is $\mathbb{1}_{a_i}$.

- For a symmetric matrix B, $\mu_h(B)$ (resp. $\mu_{\max}(B)$ $\mu_{\min}(B)$) denotes the h^{th} largest (resp. maximum, minimum) eigenvalue of B.

Towards designing our algorithm for **PartitionTesting**, we first make the following simplifying assumption. We assume that we have the following oracle at our disposal: the oracle takes a vertex a as input, and returns $D^{-\frac{1}{2}}M^t\mathbb{1}_a$, where D is the diagonal matrix of the vertex degrees, M is the transition matrix of the lazy random walk associated with the input graph, and $\mathbb{1}_a$ is the indicator vector of a. We first present in Section II-A an algorithm for **PartitionTesting** which makes use of this oracle. Following this, in Section II-B, we describe how we can (approximately) simulate the oracle, and thereby, get an algorithm for **PartitionTesting**.

We remark that our algorithms use the value of $\text{vol}(V_G)$, which is not available directly through the access model described in Section I-A. However, by the result of [26], it is possible to approximate the value of $\text{vol}(V_G)$ with an arbitrarily small multiplicative error using $\tilde{O}(\sqrt{|V_G|})$ queries.

A. The algorithm under an oracle assumption

Fix t, the length of the random walk, and assume that we have access to the oracle which returns $D^{-\frac{1}{2}}M^t\mathbb{1}_a$ on input $a \in V_G$. Our algorithm for **PartitionTesting** is presented as a main procedure PARTITIONTEST (Algorithm 2), that calls the subroutine ESTIMATE (Algorithm 1). The goal of this section is to prove guarantees about this algorithm, as stated in Theorem 7.

Algorithm 1 ESTIMATE(G, k, s, t, η)

1: Sample s vertices from V_G independently and with probability proportional to the degree of the vertices at random with replacement using **sampler**(G, η) (See Lemma 1). Let S be the multiset of sampled vertices.
2: Compute $D^{-\frac{1}{2}}M^tS$ using the oracle.
3: **return** $\mu_{k+1}((D^{-\frac{1}{2}}M^tS)^\top(D^{-\frac{1}{2}}M^tS))$.

Algorithm 2 PARTITIONTEST$(G, k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$

1: $\eta := 0.5$
2: $s := \frac{1600(k+1)^2 \cdot \ln(12(k+1)) \cdot \ln(\text{vol}(V_G))}{\beta(1-\eta)}$
3: $c := \frac{20}{\varphi_{\text{in}}^2}$, $t := c\ln(\text{vol}(V_G))$
4: $\mu_{\text{thres}} := \frac{4(k+1)\ln(12(k+1))}{\beta\cdot(1-\eta)}\text{vol}(V_G)^{-1-120c\varphi_{\text{out}}}$
5: **if** ESTIMATE$(G, k, s, t, \eta) \le \mu_{\text{thres}}$ **then**
6: Accept G.
7: **else**
8: Reject G.

Theorem 7. *Suppose $\varphi_{in}^2 > 480\varphi_{out}$. For every graph G, integer $k \ge 1$, and $\beta \in (0,1)$,*

1) *If G is (k, φ_{in})-clusterable (YES case), then* PARTITIONTEST*$(G, k, \varphi_{in}, \varphi_{out}, \beta)$ accepts.*
2) *If G is $(k, \varphi_{out}, \beta)$-unclusterable (NO case), then* PARTITIONTEST*$(G, k, \varphi_{in}, \varphi_{out}, \beta)$ rejects with probability at least $\frac{2}{3}$.*

Algorithm PARTITIONTEST calls the procedure ESTIMATE, compares the value returned with a threshold, and then decides whether to accept or reject. Procedure ESTIMATE needs to draw several samples of vertices, where each vertex of the input graph is sampled with probability proportional to its degree. This, by itself, is not allowed in the query model under consideration defined in Section I-A. Therefore, procedure ESTIMATE makes use of the following result by Eden and Rosenbaum to (approximately) sample vertices with probabilities proportional to degree.

Lemma 1 (Corollary 1.5 of [13])**.** *Let $G = (V_G, E_G)$ be an arbitrary graph, and $\eta > 0$. Let $\mathcal{D}$ denote the degree distribution of G (i.e. $\mathcal{D}(v) = \frac{\deg(v)}{vol(G)}$). Then there exists an algorithm, denoted by* ***sampler****(G, η), that with probability at least $\frac{2}{3}$ produces a vertex v sampled from a distribution $\mathcal{P}$ over V_G, and outputs "Fail" otherwise. The distribution $\mathcal{P}$ is such that for all $v \in V_G$, $|\mathcal{P}(v) - \mathcal{D}(v)| \le \eta \cdot \mathcal{D}(v)$. The algorithm uses $\tilde{O}\left(\frac{|V_G|}{\sqrt{\eta \cdot vol(V_G)}}\right)$ vertex, degree and neighbor queries.*

The proof of Theorem 7 relies on the following guarantees about the behavior of the algorithm in the YES case, and the NO case respectively, whose proofs are deferred to the full version of the paper [4].

Theorem 8. *Let $\varphi_{in} > 0$ and integer $k \ge 1$. Then for every (k, φ_{in})-clusterable graph $G = (V_G, E_G)$ (see definition 2), with $\min_{v \in V_G} deg(v) \ge 1$ the following holds:*

$$\text{ESTIMATE}(G, k, s, t, \eta) \le s \cdot \left(1 - \frac{\varphi_{in}^2}{4}\right)^{2t}.$$

Theorem 9. *Let $\varphi_{out} > 0$, $\beta \in (0,1)$, and integer $k \ge 1$. Let*

$$s = 1600(k+1)^2 \cdot \ln(12(k+1)) \cdot \ln(vol(V_G))/(\beta \cdot (1-\eta)).$$

Then for every $(k, \varphi_{out}, \beta)$-unclusterable graph $G = (V_G, E_G)$ (see definition 2), with $\min_{v \in V_G} deg(v) \ge 1$, the following holds with probability at least $\frac{2}{3}$.

$$\text{ESTIMATE}(G, k, s, t, \eta) \ge \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta) \cdot vol(V_G)} \times (1 - 30\varphi_{out})^{2t}.$$

Proof of Theorem 7: Let $t = c\ln(\mathrm{vol}(V_G))$ for $c = \frac{20}{\varphi_{\text{in}}^2}$. We call the procedure ESTIMATE with

$$s = \frac{1600(k+1)^2 \cdot \ln(12(k+1)) \cdot \ln(\mathrm{vol}(V_G))}{\beta \cdot (1-\eta)},$$

and $t = c\ln(\mathrm{vol}(V_G))$. In the YES case, by Theorem 8, ESTIMATE returns a value at most

$$\begin{aligned} & s \cdot \left(1 - \frac{\varphi_{\text{in}}^2}{4}\right)^{2t} \leq s \cdot \exp\left(-\frac{\varphi_{\text{in}}^2 t}{2}\right) \\ &= s \cdot \exp\left(-\frac{\varphi_{\text{in}}^2 c \ln(\mathrm{vol}(V_G))}{2}\right) \\ &= \frac{1600(k+1)^2 \ln(12(k+1)) \ln(\mathrm{vol}(V_G))}{\beta \cdot (1-\eta) \cdot \mathrm{vol}(V_G)^{c\varphi_{\text{in}}^2/2}} \\ &\leq \frac{1}{2} \cdot \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta)} \mathrm{vol}(V_G)^{2 - c\frac{\varphi_{\text{in}}^2}{2}}. \end{aligned}$$

In the last inequality we use the fact that $k+1 \leq \mathrm{vol}(V_G)$, and $\mathrm{vol}(V_G)$ is large enough to ensure that $200\ln(\mathrm{vol}(V_G)) \leq \mathrm{vol}(V_G)$. In the NO case, by Theorem 9, with probability at least $\frac{2}{3}$, ESTIMATE returns a value at least

$$\begin{aligned} & \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta) \cdot \mathrm{vol}(V_G)} \times (1 - 30\varphi_{\text{out}})^{2t} \\ &\geq \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta) \cdot \mathrm{vol}(V_G)} \times \exp\left(-120\varphi_{\text{out}} t\right) \\ &\geq \frac{1}{2} \cdot \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta)} \times \mathrm{vol}(V_G)^{-1-120c\varphi_{\text{out}}}. \end{aligned}$$

Since $\varphi_{\text{in}}^2 > 480\varphi_{\text{out}}$, the value of $c = \frac{20}{\varphi_{\text{in}}^2}$, chosen in PARTITIONTEST is such that $2 - c\frac{\varphi_{\text{in}}^2}{2} < -1 - 120c\varphi_{\text{out}}$. Therefore, for $\mathrm{vol}(V_G)$ large enough, the upper bound on the value returned by ESTIMATE in the YES case is less than $\mu_{\text{thres}} = \frac{1}{2} \cdot \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta)} \times \mathrm{vol}(V_G)^{-1-120c\varphi_{\text{out}}}$, which is less than the lower bound on the value returned by ESTIMATE in the NO case. ■

B. Lifting the Oracle Assumption

The goal of this section is to show how we can remove the oracle assumption that we made in Section II-A, and get an algorithm for the **PartitionTesting** problem that fits into the query complexity model, defined in Section I-A, that only allows (uniformly random) vertex, degree, and neighbor queries. This will then establish Theorem 7. The algorithm is presented as a main procedure PARTITIONTESTWITHOUTORACLE (Algorithm 4) that calls the subroutine ESTIMATEWITHOUTORACLE (Algorithm 3). These two procedures can be seen as a analogs of the procedures PARTITIONTEST (Algorithm 2) and ESTIMATE (Algorithm 1) respectively, from Section II-A.

Recall from Definition 7 that with the graph G we associated a random walk, and let M be the transition matrix of that random walk. For a vertex a of G, denote by $\mathbf{p}_a^t = M^t \mathbb{1}_a$ the probability distribution of of a t step random walk starting from a. Recall that ESTIMATE assumed the existence of an oracle that takes a vertex a of G as input, and returns $D^{-\frac{1}{2}} M^t \mathbb{1}_a$. ESTIMATEWITHOUTORACLE simulates the behavior of the oracle by running several t-step random walks from a. For any vertex b, the fraction of the random walks ending in b is taken as an estimate of $\mathbf{p}_a^t(b) = \mathbb{1}_b^\top M^t \mathbb{1}_a$, the probability that the t-step random walk started from a ends in b. However, for this estimate to have sufficiently small variance, the quantity $\|D^{-\frac{1}{2}} \mathbf{p}_a^t\|_2^2$ needs to be small enough. To check this, ESTIMATEWITHOUTORACLE uses the procedure **ℓ_2^2-norm tester**, whose guarantees are formally specified in the following lemma. The details of our **ℓ_2^2-norm tester**, which is a modification of the **ℓ_2^2-norm tester** from [7], are deferred to the full

Algorithm 3
ESTIMATEWITHOUTORACLE$(G, k, s, t, \sigma, R, \eta)$

Sample s vertices from N independently and with probability proportional to the degree of the vertices at random with replacement using **sampler**(G, η). Let S be the multiset of sampled vertices.
$r = 192s\sqrt{\mathrm{vol}(V_G)}$.
for each sample $a \in S$ **do**
    if **ℓ_2^2-norm tester**(G, a, σ, r) rejects **then**
        return ∞.
for Each sample $a \in S$ **do**
    Run R random walks starting from a, and let $\mathbf{q}_a$ be the distribution of the t-step random walk started at a.
Let Q be the matrix whose columns are $\{D^{-\frac{1}{2}}\mathbf{q}_a : a \in S\}$.
return $\mu_{k+1}(Q^\top Q)$.

Algorithm 4
PARTITIONTESTWITHOUTORACLE$(G, k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$

$\eta := 0.5$
$s := \frac{1600(k+1)^2 \cdot \ln(12(k+1)) \cdot \ln(\mathrm{vol}(V_G))}{(\beta(1-\eta))}$
$c := \frac{20}{\varphi_{\text{in}}^2}$, $t := c\ln(\mathrm{vol}(V_G))$
$\sigma := \frac{192sk(1+\eta)}{\mathrm{vol}(V_G)}$
$\mu_{\text{thres}} := \frac{1}{2} \cdot \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta)} \mathrm{vol}(V_G)^{-1-120c\varphi_{\text{out}}}$
$\mu_{\text{err}} = \frac{1}{3} \cdot \frac{8(k+1)\ln(12(k+1))}{\beta \cdot (1-\eta)} \mathrm{vol}(V_G)^{-1-120c\varphi_{\text{out}}}$
$R := \max\left(\frac{100s^2\sigma^{1/2}}{\mu_{\text{err}}}, \frac{200s^4\sigma^{3/2}}{\mu_{\text{err}}^2}\right)$
if ESTIMATEWITHOUTORACLE$(G, k, s, t, \sigma, R, \eta)$ $\leq \mu_{\text{thres}}$ **then**
    Accept G.
else
    Reject G.

version of the paper [4].

Lemma 2. *Let $G = (V_G, E_G)$. Let $a \in V_G$, $\sigma > 0$, $0 < \delta < 1$, and $R \geq \frac{16\sqrt{vol(G)}}{\delta}$. Let $t \geq 1$, and $\mathbf{p}_a^t$ be the probability distribution of the endpoints of a t-step random walk starting from a. There exists an algorithm, denoted by ℓ_2^2-**norm tester**(G, a, σ, R), that outputs accept if $\|D^{-\frac{1}{2}}\mathbf{p}_a^t\|_2^2 \leq \frac{\sigma}{4}$, and outputs reject if $\|D^{-\frac{1}{2}}\mathbf{p}_a^t\|_2^2 > \sigma$, with probability at least $1 - \delta$. The running time of the tester is $O(R \cdot t)$.*

The next two theorems formalize the guarantees of Algorithm 4 in the YES and the NO cases respectively, and their proofs are deferred to the full version of the paper [4].

Theorem 10. *Let $\varphi_{in} > 0$, and integer $k \geq 1$. Then for every (k, φ_{in})-clusterable graph $G = (V_G, E_G)$ (see definition 2), with $\min_{v \in V_G} deg(v) \geq 1$, Algorithm 4 accepts G with probability at least $\frac{5}{6}$.*

Theorem 11. *Let $\varphi_{out} > 0$, $\beta \in (0, 1)$, and integer $k \geq 1$. Then for every $(k, \varphi_{out}, \beta)$-unclusterable graph $G = (V_G, E_G)$ (see definition 2), with $\min_{v \in V_G} deg(v) \geq 1$, Algorithm 4 rejects G with probability at least $\frac{4}{7}$.*

Now we are set to prove Theorem 1.

Theorem 1 (restated). Suppose $\varphi_{\text{out}} \leq \frac{1}{480}\varphi_{\text{in}}^2$. Then there exists a randomized algorithm for **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$ which gives the correct answer with probability at least $2/3$, and which makes $\text{poly}(1/\varphi_{\text{in}}) \cdot \text{poly}(k) \cdot \text{poly}(1/\beta) \cdot \text{poly}\log(m) \cdot m^{1/2+O(\varphi_{\text{out}}/\varphi_{\text{in}}^2)}$ queries on graphs with m edges.

Proof: The correctness of the algorithm is guaranteed by Theorems 10 and 11. Since these theorems give correctness probability that is a constant larger than $1/2$, it can be boosted up to $2/3$ using standard techniques (majority of the answers of a sufficiently large constant number of independent runs). It remains to analyze the query complexity. The running time of the **sampler** algorithm to sample each vertex is $\tilde{O}(\frac{|V_G|}{\text{vol}(G)})$. Hence in total the query complexity of sampling is $\tilde{O}(s \cdot \sqrt{\text{vol}(G)})$. For each of the s sampled vertices, we run ℓ_2^2**-norm tester** once, followed by R random walks of t steps each. Each call to the ℓ_2^2**-norm tester** takes $O(rt) = O(st\sqrt{\text{vol}(V_G)}) = O(st\sqrt{m})$ queries, as guaranteed by Lemma 2. The random walks from each vertex take $O(Rt)$ time. Thus, the overall query complexity is $O(srt + sRt + s\sqrt{m})$. Substituting the values of s, r, R, and t as defined in Algorithm 4, and noting that $m = \text{vol}(V_G)/2$, we get the required bound. ∎

III. Query Lower Bound for **PartitionTesting**

In this section, we prove Theorem 5, which gives a lower bound on the query complexity of **PartitionTesting**. Our starting point is the following lower bound on **NoisyParities** (Definition 5), whose proof is deferred to the full version of the paper [4].

Theorem 4 (restated). Consider a deterministic algorithm ALG for the **NoisyParities** problem with parameters d and ε. Let $b = 1/(8 \ln d)$. Suppose ALG makes at most $n^{1/2+\delta}$ queries on n vertex graphs, where $\delta < \min(1/16, b\varepsilon)$. Then ALG gives the correct answer with probability at most $1/2 + o(1)$.

Recall that the **NoisyParities** problem has a random d-regular graph generated according to the configuration model as its underlying graph. The configuration model of Bollobás generates a random d regular graph $G = (V, E)$ over a set V of n vertices (provided dn is even) as follows. It first puts d half-edges on each vertex and identifies the set of half-edges with $V \times [d]$. Then in each round, an arbitrary unpaired half-edge (u, i) of some arbitrary vertex u is picked, and it is paired up with a uniformly random unpaired half-edge (v, j). This results in the addition of an edge (u, v) to E. This continues until all the half-edges are paired up. (This might result in self-loops and parallel edges, so G is not necessarily simple.) The following is known about the expansion of random d-regular graphs generated by the configuration model [3].

Fact 1. *For $d \geq 3$ let $\eta(d) \in (0, 1)$ be such that $(1-\eta(d))\log_2(1-\eta(d)) + (1+\eta(d))\log_2(1+\eta(d)) > 4/d$. Then with probability $1 - o(1)$, a random d-regular graph on n vertices generated from the configuration model has expansion at least $(1 - \eta(d))/2$.*

We now show how the problem **NoisyParities** reduces to testing **PartitionTesting**$(k, \varphi_{\text{in}}, \varphi_{\text{out}}, \beta)$, even for $k = 1$ and any $\beta \leq 1$. By this reduction, we establish a lower bound of $n^{1/2+\Omega(\varphi_{\text{out}})}$ on the number of queries required to test whether a graph is $(1, \varphi_{\text{in}})$-clusterable for some constant φ_{in} (the YES case), or it is $(2, \varphi_{\text{out}}, \beta)$-unclusterable for any constant $\beta \leq 1$ (the NO case). The reduction is given by Algorithm 5.

Algorithm 5 ReductionToPartitionTesting $(G = (V, E), y : E \longrightarrow \{0, 1\})$

1: Input: $G = (V, E)$, labeling $y : E \longrightarrow \{0, 1\}$
2: $V' := V \times \{0, 1\}$.
3: ▷ We denote the vertex $(v, b) \in V \times \{0, 1\}$ by v^b for readability.
4: $E'_0 := \bigcup_{e=(u,v)\in E:\ y(e)=0} \{(u^0, v^0), (u^1, v^1)\}$.
5: $E'_1 := \bigcup_{e=(u,v)\in E:\ y(e)=1} \{(u^0, v^1), (u^1, v^0)\}$.
6: $E' = E'_0 \cup E'_1$.
7: **return** $G' = (V', E')$.

Observe that the reduction is "query complexity preserving" in the sense that any query from a **PartitionTesting** algorithm asking the neighbors of a vertex

$v^b \in V'$ can be answered by making (at most) one query, asking the yet undisclosed edges incident on v in G and their labels. To establish the correctness of the reduction, it is sufficient to prove:

1) The YES case: If the edges of G are labeled independently and uniformly at random, then G' is an expander with high probability.
2) The NO case: If each edge $e = (u, v)$ of G is labeled $X(u) + X(v) + Z(u, v)$, where $Z(u, v)$ is 1 with probability ε, then with high probability G' contains a cut with n vertices on each side whose expansion is $O(\varepsilon)$.

The next two lemmas formalize the above two claims, and their proofs are deferred to the full version [4].

Lemma 3. *Let $G = (V, E)$ be a d-regular φ-expander with $|V| = n$. Suppose each edge $(u, v) \in E$ independently and uniformly given label $Y(u, v) \in \{0, 1\}$. Suppose* ReductionToPartitionTesting *on input (G, Y) returns the graph $G' = (V', E')$. Then G' is a $\min(\varphi/4, 1/32)$-expander with probability at least $1 - 2^{2n} \cdot \exp(-dn/256)$.*

Lemma 4. *Let $G = (V, E)$ be a d-regular graph with $|V| = n$. For each $v \in V$, let $X(v)$ be an independent uniformly random bit. For each edge $(u, v) \in E$, let $Z(u, v)$ be an independent random bit which is 1 with probability ε and 0 otherwise. Suppose each edge $(u, v) \in E$ is labeled $Y(u, v) = X(u)+X(v)+Z(u, v)$. Suppose* ReductionToPartitionTesting *on input (G, Y) returns the graph $G' = (V', E')$. Then with probability at least $1 - \exp(-\varepsilon nd/6)$, there exists a set $V^* \subseteq V'$ with $|V^*| = n = |V'|/2$ whose expansion is at most 2ε.*

We are now able to prove the following lower bound on the query complexity of **PartitionTesting**.

Theorem 5 (restated). There exist positive constants φ_{in} and b such that for all $\varphi_{\text{out}} \leq 1$, any algorithm that distinguishes between a $(1, \varphi_{\text{in}})$-clusterable graph (that is, a φ_{in}-expander) and a $(2, \varphi_{\text{out}}, 1)$-unclusterable graph on n vertices with success probability at least $2/3$ must make at least $(n/2)^{1/2+b\varphi_{\text{out}}/2}$ queries, even when the input is restricted to d-regular graphs for a large enough constant d.

Proof: Let $d = 512$, $\varphi = (1-\eta(d))/2 \approx 0.45$, and $\varphi_{\text{in}} = \min(\varphi/4, 1/32)$. As before, let $b = 1/(8 \ln d)$ (with $d = 512$ now). Given $\varphi_{\text{out}} \leq 1$, let $\varepsilon = \varphi_{\text{out}}/2 \leq 1/2$. Suppose there is an algorithm for **PartitionTesting** which makes $(n/2)^{1/2+\delta}$ queries on n vertex graphs and outputs the correct answer with probability at least $2/3$, for some $\delta < b\varphi_{\text{out}}/2 = b\varepsilon = \min(1/16, b\varepsilon)$ (note that $b\varepsilon \leq 1/(16 \ln 512) \leq 1/16$). Then for any probability distribution $\mathcal{D}$ over n-vertex **PartitionTesting** instances, there exists a deterministic algorithm ALG($\mathcal{D}$) making $O(n^{1/2+\delta})$ queries which outputs the correct answer with probability at least $2/3$, on a random instance of **PartitionTesting** drawn from the distribution.

Let (G, y) be a random instance of the **NoisyParities** problem with parameters d and ε, where $G = (V, E)$ is a graph and $y : E \longrightarrow \{0, 1\}$ is an edge labeling. Apply ReductionToPartitionTesting to (G, y), and thus, get a random instance G' of **PartitionTesting** from the appropriate probability distribution $\mathcal{D}$. Run ALG($\mathcal{D}$) on this instance and return the answer. Note that to answer one query of ALG($\mathcal{D}$), we make at most one query into G. Thus, this reduction gives an algorithm ALG$'$ for **NoisyParities** making at most $n^{1/2+\delta}$ queries.

The underlying graph G is a random d-regular graph on n vertices. By Fact 1, with high probability, G is a φ-expander. Hence, by Lemma 3, if (G, y) is a YES instance, then with high probability the reduced graph G' is a φ_{in}-expander for $\varphi_{\text{in}} = \min(\varphi/4, 1/32)$ (we chose d large enough so that the failure probability $2^{2n} \cdot \exp(-dn/256)$ in Lemma 3 becomes $o(1)$). On the other hand, if (G, y) is a NO instance, then by Lemma 4, the reduced graph G' is a graph on $2n$ vertices containing with high probability a subset of n vertices whose expansion is at most $2\varepsilon = \varphi_{\text{out}}$. Thus, G' is $(2, \varphi_{\text{out}}, 1)$-unclusterable. Hence, the reduction succeeds with probability $1 - o(1)$. Since ALG answers correctly with probability at least $2/3$, ALG$'$ answers correctly with probability at least $2/3 - o(1)$.

However, by Theorem 4, since ALG$'$ makes at most $n^{1/2+\delta}$ queries, ALG$'$ can be correct with probability at most $1/2 + o(1)$. This is a contradiction. ■

ACKNOWLEDGMENT

Ashish Chiplunkar is supported by the Joint Research Program ICES II–MRL Contract No. 2017-050. Michael Kapralov is supported in part by ERC Starting Grant 759471-Sublinear. Sanjeev Khanna is supported in part by National Science Foundation grant CCF-1617851.

References

[1] Zeyuan Allen Zhu, Silvio Lattanzi, and Vahab S. Mirrokni. A local algorithm for finding well-connected clusters. In *ICML*, pages 396–404, 2013.

[2] Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *Eur. J. Comb.*, 1(4):311–316, 1980.

[3] Béla Bollobás. The isoperimetric number of random regular graphs. *Eur. J. Comb.*, 9(3):241–244, 1988.

[4] Ashish Chiplunkar, Michael Kapralov, Sanjeev Khanna, Aida Mousavifar, and Yuval Peres. Testing graph clusterability: Algorithms and lower bounds. *CoRR*, abs/1808.04807, 2018. Available at `https://arxiv.org/abs/1808.04807`.

[5] Artur Czumaj, Oded Goldreich, Dana Ron, C. Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Struct. Algorithms*, 45(2):139–184, 2014.

[6] Artur Czumaj, Pan Peng, and Christian Sohler. Testing cluster structure of graphs. In *STOC*, pages 723–732, 2015.

[7] Artur Czumaj, Pan Peng, and Christian Sohler. Testing cluster structure of graphs. *CoRR*, abs/1504.03294, 2015.

[8] Artur Czumaj and Christian Sohler. Testing expansion in bounded-degree graphs. *Combinatorics, Probability & Computing*, 19(5-6):693–709, 2010.

[9] Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017.

[10] Talya Eden, Reut Levi, and Dana Ron. Testing bounded arboricity. In *SODA*, pages 2081–2092, 2018.

[11] Talya Eden, Dana Ron, and C. Seshadhri. On approximating the number of k-cliques in sublinear time. *CoRR*, abs/1707.04858, 2017.

[12] Talya Eden, Dana Ron, and C. Seshadhri. Sublinear time estimation of degree distribution moments: The degeneracy connection. In *ICALP*, pages 7:1–7:13, 2017.

[13] Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In *SOSA*, pages 7:1–7:9, 2018.

[14] Shayan Oveis Gharan and Luca Trevisan. Partitioning into expanders. In *SODA*, pages 1256–1266, 2014.

[15] Lior Gishboliner and Asaf Shapira. Deterministic vs non-deterministic graph property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:59, 2013.

[16] Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(20), 2000.

[17] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[18] Oded Goldreich and Dana Ron. Algorithmic aspects of property testing in the dense graphs model. *SIAM J. Comput.*, 40(2):376–445, 2011.

[19] Mira Gonen and Dana Ron. On the benefits of adaptivity in property testing of dense graphs. *Algorithmica*, 58(4):811–830, 2010.

[20] Satyen Kale and C. Seshadhri. An expansion tester for bounded degree graphs. *SIAM J. Comput.*, 40(3):709–720, 2011.

[21] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

[22] Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. $1 + \Omega(1)$-approximation to MAX-CUT requires linear space. In *SODA*, pages 1703–1722, 2017.

[23] László Lovász and Katalin Vesztergombi. Non-deterministic graph property testing. *Combinatorics, Probability & Computing*, 22(5):749–762, 2013.

[24] Asaf Nachmias and Asaf Shapira. Testing the expansion of a graph. *Inf. Comput.*, 208(4):309–314, 2010.

[25] Pan Peng and Christian Sohler. Estimating graph parameters from random order streams. In *SODA*, pages 2449–2466, 2018.

[26] C Seshadhri. A simpler sublinear algorithm for approximating the triangle count. *arXiv preprint arXiv:1505.01927*, 2015.