Random walks and forbidden minors III: $\mathrm{poly}(d\varepsilon^{-1})$-time partition oracles for minor-free graph classes

Akash Kumar C. Seshadhri Andrew Stolman

***Abstract*—Consider the family of bounded degree graphs in any minor-closed family (such as planar graphs). Let d be the degree bound and n be the number of vertices of such a graph. Graphs in these classes have hyperfinite decompositions, where, one removes a small fraction of edges of the graph controlled by a proximity parameter to get connected components of size independent of n. An important tool for sublinear algorithms and property testing for such classes is the partition oracle, introduced by the seminal work of Hassidim-Kelner-Nguyen-Onak (FOCS 2009). A partition oracle is a local procedure that gives consistent access to a hyperfinite decomposition, without any preprocessing. Given a query vertex v, the partition oracle outputs the component containing v in time independent of n. All the answers are consistent with a single hyperfinite decomposition.**

The partition oracle of Hassidim et al. runs in time exponential in the proximity parameter per query. They pose the open problem of whether partition oracles which run in time polynomial in reciprocal of proximity parameter can be built. Levi-Ron (ICALP 2013) give a refinement of the previous approach, to get a partition oracle that runs in quasipolynomial time per query.

In this paper, we resolve this open problem and give polynomial time partition oracles (in reciprocal of proximity parameter) for bounded degree graphs in any minor-closed family. Unlike the previous line of work based on combinatorial methods, we employ techniques from spectral graph theory. We build on a recent spectral graph theoretical toolkit for minor-closed graph families, introduced by the authors to develop efficient property testers. A consequence of our result is an efficient property tester for any monotone and additive with running time property of minor-closed families (such as bipartite planar graphs). Our result also gives query efficient algorithms for additive approximations for problems such as maximum matching, minimum vertex cover, maximum independent set, and minimum dominating set for these graph families.

I. Introduction

The algorithmic study of planar graphs is a fundamental direction in theoretical computer science and graph theory. Classic results like the Kuratowski-Wagner characterization [18], [29], linear time planarity algorithms [14], and the Lipton-Tarjan separator theorem underscore the significance of planar graphs [22]. The celebrated theory of Robertson-Seymour give a grand generalization of planar graphs through minor-closed families [24], [25], [26]. This has led to many deep results in graph algorithms, and an important toolkit is provided by separator theorems and associated decompositions [2].

Over the past decade, there have been many advances in *sublinear* algorithms for planar graphs and minor-closed families. We focus on the model of random access to bounded degree adjacency lists, introduced by Goldreich-Ron [12]. Let $G = (V, E)$ be a graph with vertex set $V = [n]$ and degree bound d. The graph is accessed through *neighbor queries*: there is an oracle that on input $v \in V$ and $i \in [d]$, returns the ith neighbor of v. (If none exist, it returns $\perp$.)

One of the key properties of bounded-degree graphs in minor-closed families is that they exhibit hyperfinite decompositions. A graph G is hyperfinite if $\forall\ 0 < \varepsilon < 1$, one can remove εdn edges from G and obtain connected components of size independent of n (we refer to these as pieces). For minor-closed families, one can remove εdn edges and get pieces of size $O(\varepsilon^{-2})$.

The seminal result of Hassidim-Kelner-Nguyen-Onak (HKNO) [13] introduced the notion of *partition oracles*. This is a local procedure that provides "constant-time" access to a hyperfinite decomposition. The oracle takes a query vertex v and outputs the piece containing v. Each piece is of size independent of n, and at most εdn edges go between pieces. Furthermore, all the answers are consistent with a single hyperfinite decomposition, despite there being no preprocessing or explicit coordination. (All queries uses the same random seed, to ensure consistency.) Partition oracles are extremely powerful as they allow a constant time procedure to directly access a hyperfinite decomposition.

As observed in previous work, partition oracles lead to a plethora of property testing results and sublinear time approximation algorithms for minor-closed graph families [13], [23]. In some sense, one can think of partition oracles as a moral analogue of Szémeredi's regularity lemma for dense graph property testing: it is a decomposition tool that immediately yields a litany of constant time (or constant query) algorithms.

We give a formal definition of partition oracles. (We deviate somewhat from the definition in Chap. 9.5 of Goldreich's book [10] by including the running time as a parameter, instead of the set size.)

Definition I.1. *Let $\mathcal{P}$ be a family of graphs with degree bound d and $T : (0,1) \to \mathbb{N}$ be a function. A procedure $\boldsymbol{A}$ is an $(\varepsilon, T(\varepsilon))$-*partition oracle *for $\mathcal{P}$ if it satisfies the following properties. The deterministic procedure takes as input random access to $G = (V, E)$ in $\mathcal{P}$, random access to a random seed r (of length polynomial in graph size), a proximity parameter $\varepsilon > 0$, and a vertex v of G. (We will think of fixing G, r, ε, so we use the notation $\boldsymbol{A}_{G,r,\varepsilon}$. All probabilities are with respect to r.) The procedure $\boldsymbol{A}_{G,r,\varepsilon}(v)$ outputs a set of vertices and satisfies the following properties.*

1) *(Consistency) The sets $\{\boldsymbol{A}_{G,r,\varepsilon}(v)\}$, over all v, form a partition of V. Also, these sets $\boldsymbol{A}_{G,r,\varepsilon}(v)$ induce connected graphs for all $v \in V$.*
2) *(Cut bound) With probability (over r) at least $2/3$, the number of edges between the sets $\boldsymbol{A}_{G,r,\varepsilon}(v)$ is at most εdn.*
3) *(Running time) For every v, $\boldsymbol{A}_{G,r,\varepsilon}(v)$ runs in time $T(\varepsilon)$.*

We stress that there is no explicit "coordination" or sharing of state between calls to $\boldsymbol{A}_{G,r,\varepsilon}(v)$ and $\boldsymbol{A}_{G,r,\varepsilon}(v')$ (for $v \neq v'$). There is no global pre-processing step once the random seed is fixed. The consistency guarantee holds with probability 1. Note that the running time $T(\varepsilon)$ is clearly an upper bound on the size of the sets $\boldsymbol{A}_{G,r,\varepsilon}(v)$. For minor-closed families, one can convert any partition oracle to one that output sets of size $O(\varepsilon^{-2})$ with a constant factor increase in the cut bound. (refer to the end of Sec. 9.5 in [10]).

The challenge in partition oracles is to bound the running time $T(\varepsilon)$. HKNO gave a partition oracle with running time $(d\varepsilon^{-1})^{\mathrm{poly}(d\varepsilon^{-1})}$. Levi-Ron [19] built on the ideas from HKNO and dramatically improved the bound to $(d\varepsilon^{-1})^{\log(d\varepsilon^{-1})}$. Yet, for all minor-closed families, one can (in linear time) remove εdn edges to get connected components of size $O(\varepsilon^{-2})$. HKNO raise the natural open question as to whether $(\varepsilon, \mathrm{poly}(d\varepsilon^{-1}))$-partition oracles exist.

In this paper, we resolve this open problem.

Theorem I.2. *Let $\mathcal{P}$ be the set of d-bounded degree graphs in a minor-closed family. There is an $(\varepsilon, \mathrm{poly}(d\varepsilon^{-1}))$-partition oracle for $\mathcal{P}$.*

A. Consequences

As observed by HKNO and Newman-Sohler [23], partition oracles have many consequences for property testing and sublinear algorithms.

Recall the definition of property testers. Let $\mathcal{Q}$ be a property of graphs with degree bound d. The distance of G to $\mathcal{Q}$ is the minimum number of edge additions/removals required to make G have $\mathcal{Q}$, divided by dn. A property tester for $\mathcal{P}$ is a randomized procedure that takes query access to an input graph G and a proximity parameter, $\varepsilon > 0$. If $G \in \mathcal{P}$, the tester accepts with probability at least $2/3$. If the distance of G to $\mathcal{Q}$ is at least ε, the tester rejects with probability at least $2/3$. We often measure the query complexity as well as time complexity of the tester.

A direct consequence of Theorem I.2 is an "efficient" analogue (for monotone and additive properties) of a theorem of Newman-Sohler stating that all properties of hyperfinite graphs are testable. A graph property closed under vertex/edge removals is called *monotone*. A graph property closed under disjoint union of graphs is called *additive*.

Theorem I.3. *Let $\mathcal{Q}$ be any monotone and additive property of bounded degree graphs of a minor-closed family. There exists a $\mathrm{poly}(d\varepsilon^{-1})$-query tester for $\mathcal{Q}$.*

If membership in $\mathcal{Q}$ can be determined exactly in polynomial (in input size) time, then $\mathcal{Q}$ has $\mathrm{poly}(d\varepsilon^{-1})$-time testers.

An appealing consequence of Theorem I.3 is that the property of bipartite planar graphs can be tested in $\mathrm{poly}(d\varepsilon^{-1})$ time. For any fixed subgraph H, the property of H-free planar graphs can be tested in the same time. And all of these bounds hold for any minor-closed family.

As observed by Newman-Sohler, partition oracles give sublinear query algorithms for any additive graph parameter that is "robust" to edge changes. Again, Theorem I.2 implies an efficient version for minor-closed families.

Theorem I.4. *Let f be a real-valued function on graphs that changes by $O(1)$ on edge addition/removals, and has the property that $f(G_1 \cup G_2) = f(G_1) + f(G_2)$ for graphs G_1, G_2 that are not connected to each other.*

For any minor-closed family $\mathcal{P}$, there is a randomized algorithm that, given $\varepsilon > 0$ and $G \in \mathcal{P}$, outputs an ad-

ditive εn-approximation to $f(G)$ and makes $\mathrm{poly}(d\varepsilon^{-1})$ *queries. If f can be computed exactly in polynomial time, then the above algorithm runs in* $\mathrm{poly}(d\varepsilon^{-1})$ *time.*

The functions captured by Theorem I.4 are quite general. Functions such as maximum matching, minimum vertex cover, maximum independent set, minimum dominating set, maxcut, distances to additive and monotone properties, etc. all have the robustness property. As a compelling application of Theorem I.4, we can get $(1+\varepsilon)$-approximations[1] for the maximum matching in planar (or any minor-closed family) graphs in $\mathrm{poly}(d\varepsilon^{-1})$ time.

These theorems are easy consequences of Theorem I.2. Using the partition oracle, an algorithm can essentially assume that the input is a collection of connected components of size $\mathrm{poly}(d\varepsilon^{-1})$, and run an exact algorithm on a collection of randomly sampled components. We defer the proofs to the full version [17].

Since the publication of this work, Levi and Shoshan applied the partition oracle of Theorem I.2 for $\mathrm{poly}(d\varepsilon^{-1})$-query algorithms to test Hamiltonicity and construct almost optimal spanning subgraphs in minor-closed families [21].

B. *Related work*

The subject of property testing and sublinear algorithms in bounded degree graphs is a vast topic. We refer the reader to Chapters 9 and 10 of Goldreich's textbook [10]. We focus on the literature relevant to sublinear algorithms for minor-closed families.

The first step towards a characterization of testable properties in the bounded-degree model was given by Czumaj-Sohler-Shapira, who showed hereditary properties in non-expanding graphs are testable [5]. This was an indication that notions like hyperfiniteness are connected to property testing. Benjamini-Schramm-Shapira achieved a breakthrough by showing that all minor-closed properties are testable, in time triply-exponential in $d\varepsilon^{-1}$ [3]. Hassidim-Kelner-Nguyen-Onak introduced partition oracles, and designed one running in time $\exp(d\varepsilon^{-1})$. Levi-Ron improved this bound to quasipolynomial in $d\varepsilon^{-1}$, using a clever analysis inspired by algorithms for minimum spanning trees [19]. Newman-Sohler built on partition oracles for minor-close families to show that all properties of hyperfinite graphs are testable [23]. Fichtenberger-Peng-Sohler showed any testable property contains a hyperfinite property [9].

There are two dominant combinatorial ideas in this line of work. The first is using subgraph frequencies in neighborhood of radius $\mathrm{poly}(\varepsilon^{-1})$ to characterize properties. This naturally leads to exponential dependencies in $\mathrm{poly}(\varepsilon^{-1})$. The second idea is to use random edge contractions to reduce the graph size. Recursive applications lead to hyperfinite decompositions, and the partition oracles of HKNO and Levi-Ron simulate this recursive procedure. This is extremely non-trivial, and leads to a recursive local procedure with a depth dependent of ε. Levi-Ron do a careful simulation, ensuring that the recursion depth is at most $\log(d\varepsilon^{-1})$, but this simulation requires looking at neighborhoods of radius $\log(d\varepsilon^{-1})$. Following this approach, there is little hope of getting a recursion depth independent of ε, which is required for a $\mathrm{poly}(d\varepsilon^{-1})$-time procedure.

Much of the driving force behind this work was the quest for a $\mathrm{poly}(d\varepsilon^{-1})$-time tester for planarity. This question was resolved recently using a different approach from spectral graph theory, which was itself developed for sublinear time algorithms for finding minors [15], [16]. A major inspiration is the random walk based one-sided bipartiteness tester of Goldreich-Ron [11]. This paper is a continuation of that line of work, and is a further demonstration of the power of spectral techniques for sublinear algorithms. The tools build on local graph partitioning techniques pioneered by Spielman-Teng [28], which is itself based on classic mixing time results of Lovász-Simonovits [20]. In this paper, we develop new diffusion-based local partitioning tools that form the core of partition oracles.

Levi and Shoshan recently introduced the weaker notion of covering partition oracles [21]. The output sets do not need to form a consistent partition, and are only required to contain the sets of a hyperfinite partitioning. They show that such oracles can be constructed from previous results on testing minor-freeness [16]. These oracles can be used to test additive and monotone properties, *assuming* that the input comes from a minor-closed family. (Theorem I.3 makes no such assumption.) We also mention other key results in the context of sublinear algorithms for minor-closed families, notably the Czumaj et al [4] upper bound of $O(\sqrt{n})$ for testing cycle minor-freeness, the Fichtenberger et al [8] upper bound of $O(n^{2/3})$ for testing $K_{2,r}$-minor-freeness, and $\mathrm{poly}(d\varepsilon^{-1})$ testers for outerplanarity and bounded treewidth graphs [30], [7].

II. Main Ideas

The starting point for this work are the spectral methods used in [15], [16]. These methods discover cut properties within a neighborhood of radius $\mathrm{poly}(d\varepsilon^{-1})$, without explicitly constructing the entire neighborhood.

[1]The maximum matching is $\Omega(n/d)$ for a connected bounded degree graph. One simply sets $\varepsilon \ll 1/d$ in Theorem I.4.

One of the key tools used in these results in a local partitioning algorithm, based on techniques of Spielman-Teng [28]. The algorithm takes a seed vertex s, performs a diffusion from s (equivalently, performs many random walks) of length $\mathrm{poly}(d\varepsilon^{-1})$, and tracks the diffusion vector to detect a low conductance cut around s in $\mathrm{poly}(d\varepsilon^{-1})$ time. We will use the term *diffusions*, instead of random walks, because we prefer the deterministic picture of a unit of "ink" spreading through the graph. A key lemma in previous results states that, for graphs in minor-closed families, this procedure succeeds from more than $(1-\varepsilon)n$ seed vertices. This yields a global algorithm to construct a hyperfinite decomposition with components of $\mathrm{poly}(d\varepsilon^{-1})$ size. Pick a vertex s at random, run the local partitioning procedure to get a low conductance cut, remove and recurse. Can there be a local implementation of this algorithm?

Let us introduce some setup. We will think of a global algorithm that processes seed vertices in some order. Given each seed vertex s, a local partitioning algorithm generates a low conductance set $C(s)$ containing s (this is called a cluster). The final output is the collection of these clusters. For any vertex v, let the *anchor* of v be the vertex s such that $v \in C(s)$. A local implementation boils down to finding the anchor of query vertex v.

Observe that at any point of the global procedure, some vertices have been clustered, while the remaining are still *free*. The global procedure described above seems hopeless for a local implementation. The cluster $C(s)$ is generated by diffusion in some subgraph G' of G, which was the set of free vertices when seed s was processed. Consider a local procedure trying to discover the anchor of v. It would need to figure out the free set corresponding to every potential anchor s, so that it can faithfully simulate the diffusion used to cluster v. From an implementation standpoint, it seems that the natural local algorithm is to use diffusions from v in G to discover the anchor. But diffusion in a subgraph G' is markedly different from G and difficult to simulate locally. Our first goal is to design a partitioning method using diffusions directly in G.

Finding low conductance cuts in subsets, by diffusion in supersets: Let us now modify the global algorithm with this constraint in mind. At some stage of the global algorithm, there is a set F of free vertices. We need to find a low conductance cut contained in F, while running random walks in G. Note that we must be able to deal with F as small as $O(\varepsilon n)$. Thus, random walks (even starting from F) will leave F quite often; so how can these walks/diffusions find cuts in F?

One of our main insights is that these challenges can be dealt with, even for diffusions of $\mathrm{poly}(d\varepsilon^{-1})$ length. We show that, for a uniform random vertex $s \in F$, a spectral partitioning algorithm that performs diffusion from s in G can detect low conductance cuts contained in F. Diffusion in the superset (all of V) provides information about the subset F. This is a technical and non-trivial result, and crucially uses the spectral properties of minor-closed families. Note that diffusions from F can spread very rapidly in short random walks, even in planar graphs. Consider a graph G, where F is a path on εn vertices, and there is a tree of size $1/\varepsilon$ rooted at every vertex of F. Diffusions from any vertex in F will initially be dominated by the trees, and one has to diffuse for at least $1/\varepsilon$ timesteps before structure within F can be detected. Thus, the proof of our theorem has to look at average behavior over a sufficiently large time horizon before low conductance cuts in F are "visible". Remarkably, it suffices to look at $\mathrm{poly}(d\varepsilon^{-1})$ timesteps to find structure in F, because of the behavior of diffusions in minor-closed families.

The main technical tool used is the Lovász-Simonovits curve technique [20], whose use was pioneered by Spielman-Teng [28]. We also use the truncated probability vector technique from Spielman-Teng to give cleaner implementations and proofs. A benefit of using diffusion (instead of random walks) on truncated vectors is that the clustering becomes deterministic.

The problem of ordering the seeds: With one technical hurdle out of the way, we end up at another gnarly problem. The above procedure only succeeds if the seed is in F. Quite naturally, one does not expect to get any cuts in F by diffusing from a random vertex in G. From the perspective of the global algorithm, this means that we need some careful ordering of the seeds, so that low conductance cuts are discovered. Unfortunately, we also need local implementations of this ordering. The authors struggled with carrying out this approach, but to no avail.

To rid ourselves of the ordering problem, let us consider the following, almost naive global algorithm. First, order the vertices according to a uniform random permutation. At any stage, there is a free set F. We process the next seed vertex s by running some spectral partitioning procedure, to get a low conductance cut $C(s)$. Simply output $C(s) \cap F$ (instead of $C(s)$) as the new cluster, and update F to $F \setminus C(s)$. It is easy to locally implement this procedure. To find the anchor of v, perform a diffusion of $\mathrm{poly}(\varepsilon^{-1})$ timesteps from v. For every vertex s with high enough value in the diffusion vector, determine if $C(s) \ni v$. The vertex

s that is lowest according to the random ordering is the anchor of v. Unfortunately, there is little hope of bounding the number of edges cut by the clustering. When s is processed, it may be that $s \notin F$, and there is no guarantee of $C(s) \cap F$. Can we modify the procedure to bound the number of cut edges, but still maintain its ease of local implementability?

The amortization argument: Consider the scenario when $F = \Theta(\varepsilon n)$. Most of the subsequent seeds processed are not in F and there is no guarantee on the cluster conductance. But every $\Theta(1/\varepsilon)$ seeds (in expectation), we will get a "good" seed s contained in F, such that $C(s) \cap F$ is a low conductance set. (This is promised by the diffusion algorithm that we develop in this paper, as discussed earlier.) Our aim is to perform some amortization, to argue that $|C(s) \cap F|$ is so large, that we can "charge" away the edges cut by the previous $\Theta(1/\varepsilon)$ seeds.

This amortization is possible because our spectral tools give us much flexibility in the (low) conductances obtained. Put differently, we essentially prove that existence of many cuts of extremely low conductance, and show that it is "easy" for a diffusion-based algorithm to find such cuts. (This is connected to the spectral behavior of minor-closed families.) As a consequence, we can actually pre-specify the size of the low conductance cuts obtained. We show that as long as $|F| = \Omega(\varepsilon n)$, we can find a *size threshold* $k = \mathrm{poly}(\varepsilon^{-1})$ such that for at least $\Omega(\varepsilon^2 n)$ vertices $s \in F$, a spectral partitioning procedure seeded at s can find a cut of size $\Theta(k)$ and conductance at most ε^c. Moreover, this cut is guaranteed to contain at least $\varepsilon^{c'} k$ vertices in F, despite the procedure being oblivious to F. The parameter c can be easily tuned, so we can increase c arbitrarily while keeping c' fixed, at the cost of polynomial increases in running time. This tunability is crucial to our amortization argument. We also show that given query access to F, a size threshold k can be computed in $\mathrm{poly}(d\varepsilon^{-1})$ time.

So when the global algorithm processes seed s, it runs the above spectral procedure to try to obtain a set of size $\Theta(k)$ with conductance at most ε^c. (If the procedure fails, the global algorithm simply set $C(s) = \{s\}$.) Thus, we cut $O(\varepsilon^c kd)$ edges for each seed processed. But after every $O(1/\varepsilon)$ seeds, we choose a "good" seed such that $|C(s) \cap F| > \varepsilon^{c'} k$. The total number of edges cut is $O(\varepsilon^c kd \times \varepsilon^{-1}) = O(\varepsilon^{c-1} kd)$. The total number of new vertices clustered is at least $\varepsilon^{c'} k$. Because we can tune parameters with much flexibility, we can set $c \gg c'$. So the total number of edges cut is $O(\varepsilon^{c-c'-1} d)$ times the number of vertices clustered, where $c - c' - 1 > 1$. Overall, we will cut only $O(\varepsilon nd)$ edges.

Making it work through phases: Unfortunately, as the process described above continues, F shrinks. Thus, the original choice of k might not work, and the guarantees on $|C(s) \cap F|$ for good seeds no longer hold. So we need to periodically recompute the value of k. In a careful analysis, we show that this recomputation is only required $\mathrm{poly}(\varepsilon^{-1})$ times. Formally, we implement the recomputation through *phases*. Each vertex is independently assigned to one of $\mathrm{poly}(\varepsilon^{-1})$ phases. (Technically, we choose the phase of a vertex by sampling an independent geometric random variable. We heavily use the memoryless property of the geometric distribution.)

For each phase, the value of k is fixed. The local partition oracle will compute these size thresholds for all phases, as a $\mathrm{poly}(d\varepsilon^{-1})$ time preprocessing step. The oracle (for v) runs a diffusion from v to get a collection of candidate anchors. For each candidate s, the oracle determines its phase, runs the spectral partitioning algorithm with correct phase parameters, and determines if the candidate's low conductance cut contains v. The anchor is simply such a candidate of minimum phase, with ties broken by vertex id.

III. Global partitioning and its local implementation

There are a number of parameters that are used in the algorithm. We list them out here for reference. It is convenient to fix the value of ε in advance, so that all the values of the following parameters are fixed. Note that all these parameters are polynomial in d and ε. We will express all running times as polynomials in these parameters, ensuring all running time are $\mathrm{poly}(d\varepsilon^{-1})$.

- $\rho = d^{-60}\varepsilon^{3000}$: Minimum probability for truncation.
- $\ell = d^6\varepsilon^{-30}$: Maximum random walk length.
- $\beta = \varepsilon/10$: Unclustered fraction cutoff.
- $\delta = d^{-70}\varepsilon^{3100}$: Phase probability.
- $\alpha = \frac{\varepsilon^{4/3}}{300{,}000}$: Heavy bucket parameter.
- $\phi = \varepsilon^{10}$: Conductance parameter.

A. Truncated diffusion

The main process used to find sets of the partition is a *truncated diffusion*. We assume that the input graph G is connected, has n vertices, and degree bound d. Define the lazy symmetric random walk matrix M as follows. For every edge (u, v), $M_{u,v} = M_{v,u} = 1/2d$. For every vertex v, $M_{v,v} = 1 - d(v)/2d$, where $d(v)$ is the degree of v. The matrix M is doubly stochastic, symmetric, and the (unique) stationary distribution is the uniform distribution.

Given a vector $\vec{x} \in (\mathbb{R}^+)^n$, diffusion is the evolution $M^t\vec{x}$. We define a truncated version, where after every step, small values are removed. For any vector $\vec{x}$, let $\mathrm{supp}(\vec{x})$ denote the support of the vector.

Definition III.1. *Define the operator $\widehat{M}\colon (\mathbb{R}^+)^n \to (\mathbb{R}^+)^n$ as follows. For $\vec{x} \in (\mathbb{R}^+)^n$, the vector $\widehat{M}\vec{x}$ is obtained by zeroing out all coordinates in $M\vec{x}$ whose value is at most ρ.*

For $t > 1$, the operator $\widehat{M^t}$ is the t-step truncated diffusion, and is recursively defined as $\widehat{M}(\widehat{M}^{t-1}\vec{x})$.

Define $\widehat{p}_{v,t}(w)$ to be the coordinate corresponding to vertex w in the t-step truncated diffusion starting from vertex v.

We stress that the t-step truncated diffusion is obtained from a standard diffusion by truncating low values at *every* step of the diffusion. Note that as the truncated diffusion progresses, the l_1-norm of the vector may decrease at each step. Importantly, for any distribution vector $\vec{x}$, $\mathrm{supp}(\widehat{M^t}\vec{x})$ has size at most ρ^{-1}. We heavily use this property in our running time analysis.

We define *level sets*, a standard concept in spectral partitioning algorithms. Somewhat abusing notation, for vertex $v \in V$, we use $\vec{v}$ to denote the unit vector in $(\mathbb{R}^+)^n$ corresponding to the vertex v. (We never use to vector notation for any other kind of vectors.)

Definition III.2. *For vertex $v \in V$, length t, and threshold k, let $L_{v,t,k}$ be the set of vertices corresponding to the k largest coordinates in $\widehat{M^t}\vec{v}$ (ties are broken by vertex id).*

For any set S of vertices, the conductance of S is $\Phi(S) := E(S,\overline{S})/[2\min(|S|,|\overline{S}|)d]$. (We use $E(S,\overline{S})$ to denote the number of edges between S and its complement.)

We describe the key subroutine that finds low conductance cuts. It performs a sweep cut over the truncated diffusion vector.

```
cluster($v,t,k$)
  1) Determine $\widehat{M^t}\vec{v}$
  2) For all $k' \in [k,2k]$ calculate $\Phi(L_{v,t,k'})$.
  3) Find the largest $k' \in [k,2k]$ (if any) with the
     following properties: $\Phi(L_{v,t,k'} \cup \{v\}) \leq \phi$ and
     $L_{v,t,k'} \in \mathrm{supp}(\widehat{M^t}\vec{v})$.
  4) If such a $k'$ exists, set $C := L_{v,t,k'} \cup \{v\}$, else
     $C := \{v\}$.
  5) Return $C$.
```

Claim III.3. *The procedure `cluster`(v,t,k) runs in time $O(\rho^{-1}td\log(\rho^{-1}td) + kd\log k)$. The output set C has the following properties. (i) $v \in C$. (ii) If C is not a singleton, then $|C| \in [k,2k]$, $\Phi(C) \leq \phi$, and $C \subseteq \mathrm{supp}(\widehat{M^t}\vec{v})$.*

Proof: Deferred to the full version [17] ■

B. The global partitioning procedure

The global partitioning procedure `globalPartition` will output a partition of the vertices satisfying the conditions in Definition I.1. This global procedure will run in linear time. In the next subsection, we show how the output of the global procedure can be generated locally in $\mathrm{poly}(\varepsilon^{-1})$ time, thereby giving us the desired partition oracle. It will be significantly easier to understand and analyze the partition properties of the global procedure.

The key ingredient in `globalPartition` that allows for a local implementation is a *preprocessing* step. The preprocessing allows for the "coordination" required for consistency of various local partitioning steps. All the randomness is used in the preprocessing, after which the actual partitioning is deterministic. The job of the preprocessing is to find the following sets of values, which are used for two goals: (i) ordering vertices, (ii) setting parameters for calls to `cluster`.

The preprocessing generates, for all vertices v, the following values.

- h_v: The *phase* of v.
- k_v: The size threshold of v.
- t_v: The walk length of v.

Before giving the procedure description, we explain how these values are generated.

Phases: For each v, h_v is set to $\max(X,\overline{h})$, where X is independently sampled from $Geo(\delta)$, the geometric distribution with parameter δ. Moreover $\overline{h} := 2\delta^{-1}\log(\delta^{-1})$, so the maximum phase value is capped.

Size thresholds: The computation of these thresholds is the most complex part of our algorithm (and analysis), and is the "magic ingredient" that makes the partition oracle possible. We first run a procedure `findr` that runs in $\mathrm{poly}(\varepsilon^{-1})$ time and outputs a set of *phase size thresholds* $k_1, k_2, \ldots, k_{\overline{h}}$. All the thresholds have value at most ρ^{-1} and $k_{\overline{h}}$ will be zero. The (involved) description of `findr` and its properties are in §IV. For now, it suffices to say that its running time is $\mathrm{poly}(\varepsilon^{-1})$, and that it outputs phase size thresholds. The size threshold for a vertex v is simply k_{h_v}, corresponding to the phase it belongs to.

Walk lengths: These are simply chosen independently and uniformly in $[1,\ell]$.

The analysis is more transparent when we assume that all the randomness used by the algorithm is in a random seed $\boldsymbol{R}$, of $O(n \cdot \mathrm{poly}(\varepsilon^{-1}))$ length. The seed $\boldsymbol{R}$ is passed as an argument to the partitioning procedure, which uses $\boldsymbol{R}$ to generate all the values described above. (For convenience, we will assume random access to the adjacency list of G, without passing the graph as a parameter.)

It is convenient to define an ordering on the vertices, given these values. For cleaner notation, we drop the dependence on $\boldsymbol{R}$.

Definition III.4. *For vertex $u, v \in V$, we say that $u \prec v$ if: $h_u < h_v$ or if $h_u = h_v$, the id of v is less than that of v.*

```
globalPartition($\boldsymbol{R}$)
Preprocessing:
  1) For every $v \in V$:
     a) Use $\boldsymbol{R}$ to set $h_v := \max(X, \overline{h})$ ($X \sim Geo(\delta)$).
     b) Use $\boldsymbol{R}$ to set $t_v$ uniform random in $[1, \ell]$.
  2) Call findr($\boldsymbol{R}$) to generate values $k_1, k_2, \ldots, k_{\overline{h}}$.
     For every $v \in V$, set $k_v = k_{h_v}$.
Partitioning:
  1) Initialize the partition $\boldsymbol{P}$ as an empty collection.
     Initialize the free set $F := V$.
  2) For all vertices in $V$ in increasing order of $\prec$:
     a) Compute $C = $ cluster$(v, t_v, k_v)$.
     b) Add the connected components of $C \cap F$ to
        the partition $\boldsymbol{P}$.
     c) Reset $F = F \setminus C$.
  3) Output $\boldsymbol{P}$.
```

Since all of our subsequent discussions are about `globalPartition`, we abuse notation assuming that the preprocessing is fixed. We refer to `cluster`(v) to denote `cluster`(v, t_v, k_v). These are the only calls to cluster that are ever discussed, so it is convenient to just parametrize by the vertex argument. Furthermore, for ease of notation, we sometimes refer to the output of the procedure as `cluster`(v).

We observe that the output $\boldsymbol{P}$ is indeed a partition of V into connected components. At any intermediate step, the free set F is precisely the set of vertices that have not been assigned to a cluster. Note that `cluster`(v) always contains v (Claim III.3), so all vertices eventually enter (the sets of) $\boldsymbol{P}$.

We note that v might not be in F when `cluster`(v) is called. This may lead to new components in $\boldsymbol{P}$ that do not involve v, which may actually not be low conductance cuts. This may seem like an oversight: why initiate diffusion clusters from vertices that are already partitioned? Many challenges in our analysis arise from such clusters. On the other hand, such an "oblivious" partitioning scheme leads to a simple local implementation.

C. The local implementation

A useful definition in the local implementation is that of *anchors* of vertices. As mentioned earlier, we fix the output of the preprocessing (which is equivalent to fixing $\boldsymbol{R}$).

Definition III.5. *Consider the running of* `globalPartition`$(\boldsymbol{R})$. *The* anchor *of w is the (unique) vertex w such that the component in $\boldsymbol{P}$ containing v was created by the call to* `cluster`(v).

Suppose we label every vertex by its anchor. We can easily determine the sets of $\boldsymbol{P}$ locally.

Claim III.6. *The sets of $\boldsymbol{P}$ are exactly the maximal connected components of vertices with the same anchor.*

Proof: We prove by induction over the $\prec$ ordering of vertices. The base case is vacuously true. Suppose, just before v is considered, all current sets in $\boldsymbol{P}$ are maximal connected components with the same anchor, which cannot be v. No vertex in F can have an anchor yet; otherwise, it would be clustered and part of (a set in) $\boldsymbol{P}$. All the new vertices clustered have v as anchor. Moreover, the sets added to $\boldsymbol{P}$ are precisely the maximal connected components with v as anchor. ■

We come to a critical definition that allows for searching for anchors. We define the "inverse ball" of a vertex: this is the set of all vertices that reach v through truncated diffusions. We note that reachability is not symmetric, because the diffusion is truncated at every step.

Definition III.7. *For $v \in V$, let $IB(v) = \{w \mid \exists t \in [0, \ell], v \in \mathrm{supp}(\widehat{M^t \vec{w}})\}$.*

Claim III.8. $|IB(v)| \leq \ell\rho^{-1}$.

Proof: All vertices $w \in IB(v)$ have the property that (for some $t \leq \ell$) $\widehat{p}_{w,t}(v) \neq 0$. That implies that $p_{w,t}(v) \geq \rho$. By the symmetry of the random walk, $p_{v,t}(w) \geq \rho$. For any fixed t, there are at most ρ^{-1} such vertices w. Overall, there can be at most $\ell\rho^{-1}$ vertices in $IB(v)$. ■

Now we have a simple characterization of the anchor that allows for local implementations.

Lemma III.9. *The anchor of v is the smallest vertex (according to $\prec$) in the set $\{s | s \in IB(v)$ and $v \in$ `cluster`$(s)\}$.*

Proof: Let the anchor of v be the vertex u. We first argue that u in the given set. Clearly, $v \in$ `cluster`(u).

If $u = v$, then $u = v \in IB(v)$ and we are done. Suppsoe $u \neq v$. Then $\texttt{cluster}(u)$ is not a singleton (since it contains v). By Claim III.3, $\texttt{cluster}(u)$ is contained in the support of $\widehat{M}^{t_v}\vec{u}$, implying that $v \in \mathrm{supp}(\widehat{M}^{t_v}\vec{u})$. Thus, $u \in IB(v)$ and the anchor u is present in the given set.

It remains to argue that u is the smallest such vertex. Suppose there exists $u' \prec u$ in this set. In `globalPartition`, $\texttt{cluster}(u')$ is called before $\texttt{cluster}(u)$. At the end of this call, v is partitioned and would have u' as its anchor. Contradiction. ■

We are set for the local implementation. For a vertex v, we compute $IB(v)$ and run $\texttt{cluster}(u)$ for all $u \in IB(v)$. By Lemma III.9, we can compute the anchor of v, and by Claim III.6, we can perform a BFS to find all connected vertices with the same anchor.

We begin by a procedure that computes $IB(v)$. Since the truncated diffusion is not symmetric, this requires a little care. We use $N(u)$ to denote the neighborhood of vertex u.

```
findIB($v$)
  1) Initialize $S = \{v\}$.
  2) For every $t = 1, \ldots, \ell$:
     a) For every $w \in S \cup N(S)$, compute $\widehat{M}^t\vec{w}$. If $v \in \mathrm{supp}(\widehat{M}^t\vec{w})$, add $v$ to $S$.
  3) Return $S$.
```

Claim III.10. *The output of* $\texttt{findIB}(v)$ *is* $IB(v)$. *The running time is* $O(d^2\ell^3\rho^{-2})$.

Proof: Deferred to the full version, [17]. ■

We can now describe the local partitioning oracle (modulo the description of `findr`).

```
findAnchor($v, \boldsymbol{R}$)
  1) Run findr($\boldsymbol{R}$) to get the set $K = \{k_1, k_2, \ldots, k_{\overline{h}}\}$.
  2) Run findIB($v$) to compute $IB(v)$.
  3) Initialize $A = \emptyset$.
  4) For every $s \in IB(v)$:
     a) Using $\boldsymbol{R}$ determine $h_s, t_s$. Using $K$, determine $k_s$.
     b) Compute $C = \texttt{cluster}(s, t_s, k_s)$.
     c) If $C \ni v$, then add $s$ to $A$.
  5) Output the smallest vertex according to $\prec$ in $A$.
```

```
findPartition($v, \boldsymbol{R}$)
  1) Call findAnchor($v, \boldsymbol{R}$) to get the anchor $s$.
  2) Perform BFS from $v$. For every vertex $w$ encountered, first call findAnchor($w, \boldsymbol{R}$). If the anchor is $s$, add $w$ to the BFS queue (else, ignore $w$).
  3) Output the set of vertices that entered the BFS queue.
```

The following claim is a direct consequence of Lemma III.9 and Claim III.10.

Claim III.11. *The procedure* $\texttt{findAnchor}(v, \boldsymbol{R})$ *outputs the anchor of v and runs in time* $O((d\ell\rho^{-1})^3)$ *plus the running time of* `findr`.

Proof: Observe that $\texttt{findAnchor}(v, \boldsymbol{R})$ finds $IB(v)$, computes $\texttt{cluster}(s)$ for each $s \in IB(v)$, and outputs the smallest (by $\prec$) s such that $v \in \texttt{cluster}(s)$. By Lemma III.9, the output is the anchor of v.

By Claim III.10, the running time of $\texttt{findIB}(v)$ is $O(d^2\ell^3\rho^{-2})$. The number of calls to `cluster` is $|IB(v)|$, which is at most $\ell\rho^{-1}$ (Claim III.8). Each call to `cluster` runs in time $O(d\ell\rho^{-2})$, by Claim III.3 and the fact that $k_s \leq \rho^{-1}$). Ignoring the call to `findr`, the total running time is $O(d^2\ell^3\rho^{-3})$. ■

Theorem III.12. *The output of* $\texttt{findPartition}(v, \boldsymbol{R})$ *is precisely the set in* $\boldsymbol{P}$ *containing v, where* $\boldsymbol{P}$ *is the partition output by* $\texttt{globalPartition}(\boldsymbol{R})$. *The running time of* $\texttt{findPartition}(v, \boldsymbol{R})$ *is* $O((d\ell\rho^{-1})^4)$ *plus the running time of* `findr`.

Proof: By Claim III.11, `findAnchor` correctly outputs the anchor. By Claim III.6, the set S in $\boldsymbol{P}$ containing v is exactly the maximal connected component of vertices sharing the same anchor (as v). The set S in $\boldsymbol{P}$ is generated in $\texttt{globalPartition}(\boldsymbol{R})$ by a call to `cluster`, whose output is a set of size at most ρ^{-1}. The total number of calls to `findAnchor` made by $\texttt{findPartition}(v, \boldsymbol{R})$ is at most $d\rho^{-1}$, since a call is made to either a vertex in the set S or a neighbor of S. Overall, the total running time is $O((d\ell\rho^{-1})^5)$ plus the running time of `findr`. (Instead of calling `findr` in each call to `findAnchor`, one can simply store its output.) ■

IV. Coordination through the size thresholds: the procedure `findr`

We now come to the heart of our algorithm; coordination through `findr`. This section gives the crucial ingredient in arguing that the partitioning scheme does not cut too many edges. The ordering of vertices (to form clusters) is chosen independent of the graph structure. It is highly likely that, as the partitioning proceeds, newer `cluster`(v) sets overlap heavily with the existing partition. Such clusters may cut many new edges, without clustering enough vertices. Note that `cluster`(v) is a low conductance cut only in the original graph; it might have high conductance restricted to F (the current free set).

To deal with such "bad" clusters, we need to prove that every so often, `cluster`(v) will successfully partition enough new vertices. Such "good" clusters allow the partitioning scheme to suffer many bad clusters. This argument is finally carried about by a careful charging argument. First, we need to argue that such good clusters exist. The key tool is given by the following theorem, which is proved using spectral graph theoretic methods. We state the theorem as an independent statement.

Theorem IV.1. *Let G be a bounded degree graph in a minor-closed family. Let F be an arbitrary set of vertices of size at least βn. There exists a size threshold $k \leq \rho^{-1}$ such that the following holds. For at least $(\beta^2/\log^2\beta^{-1})n$ vertices $s \in F$, there are at least $(\beta/\log^2\beta^{-1})\ell$ timesteps $t \leq \ell$ such that: there exists $k' \in [k, 2k]$ such that (i) $L_{s,t,k'} \subseteq \mathrm{supp}(\widehat{M}^t\vec{s})$, (ii) $\Phi(L_{s,t,k'} \cup \{s\}) < \phi$, and (iii) $|L_{s,t,k'} \cap F| \geq \beta^3 k$.*

The proof of this theorem is deferred to the full version, [17]. In this section, we apply this theorem to complete the description of the partition oracle and prove its guarantees.

We discuss the significance of this theorem. The diffusion used to define $L_{s,t,k'}$ occurs in G, but we are promised a low conductance cut with non-trivial intersection with F (since $\phi \ll \beta^3$). Moreover, such cuts are obtained for a non-trivial fraction of timesteps, so we can choice one uar. Given oracle access to membership in F, it is fairly easy to find such a size threshold by random sampling.

The importance of phases: Recall the global partitioning procedure `globalPartition`. We can think of the partitioning process as divided into phases, where the hth phase involves calling `cluster`(v, t_v, k_v) for all vertices v whose phase value is h. Consider the free set at the beginning of a phase h, denoting it F_h. We apply Theorem IV.1 to determine the size threshold k_h. Since all k_v values in this phases are precisely k_h, this size threshold "coordinates" all clusters in this phase. As the phase proceeds, the free set shrinks, and the size threshold k_h stops satisfying the properties of Theorem IV.1. Roughly speaking, at this point, we start a new phase $h+1$, and recompute the size threshold. The frequency of recomputation is chosen carefully to ensure that the total running time remains $\mathrm{poly}(\varepsilon^{-1})$.

We now discuss the randomness involved in selecting phases and why geometric random variables are used. Recall that h_v is independently (for all v) set to be $\min(X, \overline{h})$, where $X \sim Geo(\delta)$. We first introduce some notation regarding phases.

Definition IV.2. *The* phase h seeds, *denoted V_h, are the vertices whose phase value is h. Formally, $V_h = \{v \mid h_v = h\}$. We use $V_{<h}$ to denote $\bigcup_{h'<h} V_{h'}$. (We analogously define $V_{\leq h}, V_{\geq h}$.)*

The free set at phase h, *denoted F_h, is the free set F in* `globalPartition`*, just before the first phase h vertex is processed. Formally, $F_h = V \setminus \bigcup_{v \in V_{<h}}$ `cluster`(v).*

One can think of the V_hs being generated iteratively. Assume that we have fixed the vertices in $V_1, \ldots, V_{h-1}$. All other vertices are in $V_{\geq h}$, implying that $h_v \geq h$ for such vertices. By the properties of the geometric random variables, $\Pr[h_v = h+1 | h_v > h] = \delta$. Thus, we can imagine that V_{h+1} is generated by independently sampling each element in $V_{\geq h}$ with δ probability. We restate this observation as Claim IV.4. Claim IV.5 is a simple Chernoff bound argument.

Before proceeding, we state some standard Chernoff bounds (Theorem 1.1 of [6]).

Theorem IV.3. *Let $X_1, X_2, \ldots, X_r$ be independent variables in $[0,1]$. Let $\mu := \mathbf{E}[\sum_i X_i]$.*

- $\Pr[X \geq 3\mu/2] \leq \exp(-\mu/12)$.
- $\Pr[X \leq \mu/2] \leq \exp(-\mu/8)$.
- *For $t \geq 6\mu$,* $\Pr[X \geq t] \leq 2^{-t}$.

Claim IV.4. *For all $v \in V$ and $1 < h < \overline{h}$,* $\Pr[v \in V_h \mid v \in V_{\geq h}] = \delta$.

Claim IV.5. *Let $h < \overline{h}$. Condition on the randomness used to specify $V_1, V_2, \ldots, V_{h-1}$. Let S be an arbitrary subset of $V_{\geq h}$. With probability at least $1 - 2\exp(-\delta|S|/12)$ over the choice of V_h, $|S \cap V_h| \in [\delta|S|/2, 2\delta|S|]$.*

Proof: For every $s \in S$, let X_s be the indicator random variable for $s \in V_h$. By Claim IV.4 and independent phase choices for each vertex, the X_s are independent Bernoullis with δ probability. By the Chernoff lower tail of Theorem IV.3, $\Pr[\sum_{s \in S} X_s \leq \delta|S|/2] \leq \exp(-\delta|S|/8)$ and $\Pr[\sum_{s \in S} X_s \geq 2\delta|S|] \leq \exp(\delta|S|/12)$. A union bound completes the proof. ■

Claim IV.6. *With probability at least $1 - 2^{-\delta n}$, $|V_{\overline{h}}| \leq \delta n$.*

Proof: Recall that $\overline{h}$ is the last phase and $\overline{h} = 2\delta^{-1}\log(\delta^{-1})$. The probability that $X \sim Geo(\delta)$ is at least $2\delta^{-1}\log(\delta^{-1})$ is $(1-\delta)^{2\delta^{-1}\log(\delta^{-1})-1} < \delta/6$. Hence, the probability that any vertex lies in $V_{\overline{h}}$ is at most $\delta/6$ and the expectation of $V_{\overline{h}}$ is at most $\delta n/6$. . By the Chernoff bound of Theorem IV.3, $\Pr[|V_{\overline{h}}| \geq \delta n] \leq 2^{-\delta n}$. ■

With this preamble, we proceed to the description of `findr` and the main properties of its output.

A. The procedure `findr`

It is convenient to assume that for all v, h_v and t_v have been chosen. These quantities are chosen independently for each vertex using simple distributions, so we will not carry as arguments the randomness used to decide these quantities. Recall that the output of `findr` is the set of size thresholds $\{k_1, k_2, \ldots, k_{\overline{h}}\}$. It is convenient to use K_h to denote $\{k_1, k_2, \ldots, k_h\}$. Before describing `findr`, we define a procedure that is a membership oracle for F_h.

```
IsFree($u, h, K_{h-1}$)
  1) If $h = 1$, output YES.
  2) Run findIB($u$) to determine $IB(u)$. Let $C$ be
     $IB(u) \cap V_{<h}$.
  3) Using $K_{h-1}$, determine $k_v$ for all $v \in C$.
  4) For all $v \in C$, compute cluster($v, t_v, k_v$). If
     the union contains $u$, output NO. Else, output
     YES.
```

Claim IV.7. *Assume that K_{h-1} is provided correctly. Then* `IsFree`(v, h, K_{h-1}) *outputs YES iff $v \in F_h$. The running time is $O((d\ell\rho^{-1})^3)$.*

Proof: Deferred to the full version, [17]. ■

We have the necessary tools to define the procedure `findr`. We will need the following definition in our description and analysis of `findr`.

Definition IV.8. *Assume $F_h \geq \beta n$. A vertex $s \in V_{\geq h}$ is called* (h,k)-viable *if $C :=$ `cluster`(s, t_s, k) is not a singleton and $|C \cap F_h| \geq \beta^3 k$. (If $F_h < \beta n$, no vertex is (h,k)-viable.)*

Let us motivate this definition. When $C :=$ `cluster`(s, t_s, k) is not a singleton, it is a low conductance cut of $\Theta(k)$ vertices. The vertex s is (h,k)-viable if C contains a non-trivial fraction of free vertices available in the hth phase. The viable vertices are those from which clustering will make significant "progress" in the hth phase. For each h, the procedure `findr` searches for values of k that lead to many (h,k)-viable vertices. In the next section, we prove that having sufficiently many clusters come from viable vertices ensures the cut bound of Definition I.1.

```
findr($\boldsymbol{R}$)
  1) For $h = 1$ to $\overline{h}$:
     a) Sample $\beta^{-10}$ uar vertices independently. Let
        $S_h$ be the multiset of sampled vertices that are
        in phase $\geq h$.
     b) If $|S_h| \leq \beta^{-9}/2$, set $k_h = 0$ and continue for
        loop. Else, reset $S_h$ to the multiset of the first
        $\beta^{-8}$ vertices sampled.
     c) For $k \in [\rho^{-1}]$ and for every $s \in S_h$:
        i) Compute $C :=$ cluster($s, t_s, k$).
        ii) For all $u \in C$, call IsFree($u, h, K_{h-1}$)
            to determine if $u \in F_{h-1}$.
        iii) If $C$ is not a singleton and $|C \cap F_{h-1}| \geq$
             $\beta^3 k$, mark $s$ as being $(h,k)$-viable.
     d) If there exists some $k$ such that there are at
        least $12\beta^4|S_h|$ $(h,k)$-viable vertices, assign an
        arbitrary such $k$ as $k_h$. Else, assign $k_h := 0$.
  2) Output $K_{\overline{h}} = \{k_1, k_2, \ldots, k_{\overline{h}}\}$.
```

Claim IV.9. *The running time of* `findr` *is $O((d\ell\delta^{-1}\rho^{-1})^5)$.*

Proof: There are $\overline{h} = 2\delta^{-1}\log(\delta^{-1})$ iterations. We compute the running time of each iteration. There are at most $\rho^{-1}\beta^{-8}$ calls to `cluster`, each of which takes $O(d\ell\rho^{-2})$ time by Claim III.3. For each call to `cluster`, there are at most ρ^{-1} calls to `IsFree`. Each call to `IsFree` takes $O((d\ell\rho^{-1})^3)$ time (Claim IV.7). The running time of each iteration is $O(\beta^{-10} + d\ell\rho^{-3}\beta^{-8} + d^3\ell^3\rho^{-5}\beta^{-8})$. By the parameter settings, since $\ell^2 \geq \varepsilon^{2\cdot 30} \geq (\varepsilon/10)^{-8} = \beta^{-8}$, the running time of each iteration $O((d\ell\rho^{-1})^5)$. The total running time is $O((d\ell\delta^{-1}\rho^{-1})^5)$. ■

The following theorem gives the main guarantee of `findr`. The proof is a fairly straightforward Chernoff bound on top of an application of Theorem IV.1. Quite simply, the proof just says the following. Theorem IV.1 shows the existence of (h,k) pairs for which many vertices are viable. The `findr` procedure finds such pairs by random sampling.

Theorem IV.10. *The following property of the values $K_{\overline{h}}$ of* `findr`$(\boldsymbol{R})$ *and the preprocessing choices holds with probability at least $1 - \exp(-1/\varepsilon)$ over all the randomness in $\boldsymbol{R}$. For all $h \leq \overline{h}$, if $|F_h| \geq \beta n$, at least $\beta^5\delta n$ vertices in V_h are (h, k_h)-viable.*

Proof: Deferred to the full version, [17] ■

V. Proving the cut bound: the amortization argument

We come to the final piece of proving the guarantees of Theorem I.2. We need to prove that the number of edges cut by the partition of `globalPartition` is at most εnd. This requires an amortization argument explained below. For the sake of exposition, we will ignore constant factors in this high-level description. One of the important takeaways is how various parameters are chosen to prove the cut bound.

Consider phase h where $|F_h| \geq \beta n$. Let us upper bound the number of edges cut by the clustering done on this phase. Roughly speaking, $|V_h| = \delta n$, so there are δn clusters created in this phase. Each cluster in this phase has at most $2k_h$ vertices. The number of edges cut by each such cluster is at most $2\phi k_h d$ (since `cluster` outputs a low conductance cut; ignore singleton outputs). So the total number of edges cut is at most $2\phi\delta k_h nd$.

Let us now lower bound the number of new vertices that are partitioned in phase h; this is the set $F_{h+1} \setminus F_h$. For each (h, k_h)-viable v in V_h, `cluster`(v) contains at least $\beta^3 k_h$ vertices in F_h. These will be newly partitioned vertices. Here comes the primary difficulty: the clusters for the different such v might not be disjoint. We need to lower bound the union of the clustered vertices in F_h. An alternate description of the challenge is as follows. We are only guaranteed that clusters from viable vertices v contains many vertices in F_h, the free set at the *beginning* of phase h. What we really need is for the cluster from v to contain many free vertices *at the time* that v is processed. Phases were introduced to solve this problem. By reducing δ, we can limit the size of V_h, thereby limiting the intersection between the clusters produced in this phase.

We now explain the math behind this argument. Consider some $w \in F_h$ and let c_w be the number of vertices in $V_{\geq h}$ that cluster v (call these seeds). Thus, $c_w = |\{s \mid s \in V_{\geq h}, v \in \texttt{cluster}(s)\}$. The vertex w is clustered in phase h iff one of these c_w seeds is selected in V_h. By Claim IV.4, each such seed is independently selected in V_h with probability δ. The probability that w is clustered in this phases is precisely $1-(1-\delta)^{c_w}$. Crucially, $c_w \leq |IB(w)| \leq \ell\rho^{-1}$. We chose $\delta \ll \ell\rho^{-1}$, so $1-(1-\delta)^{c_w} \approx \delta c_w$.

Thus, the expected number of newly clustered vertices is at least $\sum_{w\in F_h} \delta c_w$. By rearranging summations, $\sum_{w\in F_h} c_w = \sum_{v\in V_{\geq h}} |\texttt{cluster}(v) \cap F_h|$. For every (h, k_h)-viable vertex v in $V_{\geq h}$, $|\texttt{cluster}(v) \cap F_h| \geq \beta^3 k_h$. The arguments in the proof of Theorem IV.10 shows that there are $\beta^5 n$ such vertices in $V_{\geq h}$ whp. Hence, we can lower bound (in expectation) the new number of newly clustered vertices as follows:

$$\sum_{w\in F_h} \delta c_w \geq \delta \cdot (\beta^5 n) \cdot (\beta^3 k_h) = \delta\beta^8 k_h n$$

We upper bounded the number of edges cut by $2\phi\delta k_h nd$. The ratio of edges cut to vertices clustered is $8\phi\beta^{-8}d$. The parameters are set to ensure that $8\phi\beta^{-8} \ll \varepsilon$, so the total number of edges cut is εnd.

The formal analysis requires some care to deal with conditional probabilities and dependencies between various phases. Also, Theorem IV.10 talks about V_h and not $V_{\geq h}$, which necessitates some changes. But the essence of the argument is the same and is deferred to the full version, [17].

Acknowledgements

We acknowledge Reut Levi for pointing out a correction in the statement of Theorem I.3.

References

[1] Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990.

[2] Noga Alon, Paul D. Seymour, and Robin Thomas. Planar separators. *SIAM J. Discrete Math.*, 7(2):184–193, 1994. 1

[3] I. Benjamini, O. Schramm, and A. Shapira. Every minor-closed property of sparse graphs is testable. In *Symposium on the Theory of Computing (STOC)*, pages 393–402, 2008. 3

[4] Artur Czumaj, Oded Goldreich, Dana Ron, C Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 45(2):139–184, 2014. 3

[5] Artur Czumaj, Asaf Shapira, and Christian Sohler. Testing hereditary properties of nonexpanding bounded-degree graphs. *SIAM Journal on Computing*, 38(6):2499–2510, 2009. 3

[6] D. P. Dubhashi and A. Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge, 2009. 9

[7] Alan Edelman, Avinatan Hassidim, Huy N. Nguyen, and Krzysztof Onak. An efficient partitioning oracle for bounded-treewidth graphs. In *Workshop on Randomization and Computation (RANDOM)*, pages 530–541, 2011. 3

[8] Hendrik Fichtenberger, Reut Levi, Yadu Vasudev, and Maximilian Wötzel. On testing minor-freeness in bounded degree graphs with one-sided error. *CoRR*, abs/1707.06126, 2017. 3

[9] H. Fichtenberger, P. Peng, and C. Sohler. Every testable (infinite) property of bounded-degree graphs contains an infinite hyperfinite subproperty,. In *Symposium on Discrete Algorithms (SODA)*, page 714726, 2019. 3

[10] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. 2, 3

[11] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999. 3

[12] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. 1

[13] A. Hassidim, J. Kelner, H. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Foundations of Computer Science (FOCS)*, pages 22–31, 2009. 1, 2

[14] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974. 1

[15] Akash Kumar, C. Seshadhri, and Andrew Stolman. Finding forbidden minors in sublinear time: A $o(n^{1/2 + o(1)})$-query one-sided tester for minor closed properties on bounded degree graphs. In *Foundations of Computer Science (FOCS)*, pages 509–520, 2018. 3

[16] Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors II: a poly($d\,\epsilon^{-1}$)-query tester for minor-closed properties of bounded degree graphs. In *STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 559–567, 2019. 3

[17] Akash Kumar and C. Seshadhri and Andrew Stolman, *Random walks and forbidden minors III: poly*(d/ϵ)*-time partition oracles for minor-free graph classes*, Electron. Colloquium Comput. Complex. (2021), volume 28. 3, 6, 8, 9, 10, 11

[18] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematica*, 15:271–283, 1930. 1

[19] Reut Levi and Dana Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Transactions on Algorithms (TALG)*, 11(3):24, 2015. 2, 3

[20] László Lovász and Miklós Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *Foundations of Computer Science (FOCS)*, pages 346–354, 1990. 3, 4

[21] R. Levi and N. Shoshan. Testing hamiltonicity (and other problems) in minor-free graphs. Technical Report 2102.11728, arXiv, 2021. 3

[22] Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. 1

[23] Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013. 2, 3

[24] N. Robertson and P. D. Seymour. Graph minors. XII. Distance on a surface. *Journal of Combinatorial Theory Series B*, 64(2):240–272, 1995. 1

[25] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory Series B*, 63(1):65–110, 1995. 1

[26] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory Series B*, 92(1):325–357, 2004. 1

[27] D. Spielman. Lecture notes on spectral graph theory. http://www.cs.yale.edu/homes/spielman/eigs/.

[28] D. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2012. 3, 4

[29] K. Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114:570–590, 1937. 1

[30] Yuichi Yoshida and Hiro Ito. Testing outerplanarity of bounded degree graphs. *Algorithmica*, 73(1):1–20, 2015. 3