

Noise Tolerance of Expanders and Sublinear Expander Reconstruction

Satyen Kale Yuval Peres

Microsoft Research

One Microsoft Way

Redmond, WA 98052

{satyen.kale, peres}@microsoft.com

C. Seshadhri

Dept. of Computer Science

Princeton University

35 Olden St, Princeton, NJ 08540

csesha@cs.princeton.edu

Abstract

We consider the problem of online sublinear expander reconstruction and its relation to random walks in “noisy” expanders. Given access to an adjacency list representation of a bounded-degree graph G , we want to convert this graph into a bounded-degree expander G' changing G as little as possible. The graph G' will be output by a distributed filter: this is a sublinear time procedure that given a query vertex, outputs all its neighbors in G' , and can do so even in a distributed manner, ensuring consistency in all the answers.

One of the main tools in our analysis is a result on the behavior of random walks in graph that are almost expanders: graphs that are formed by arbitrarily connecting a small unknown graph (the noise) to a large expander. We show that a random walk from almost any vertex in the expander part will have fast mixing properties, in the general setting of irreducible finite Markov chains. We also design sublinear time procedures to distinguish vertices of the expander part from those in the noise part, and use this procedure in the reconstruction algorithm.

1. Introduction

Large data sets are ubiquitous today with the advent of high speed computers and connectivity. Being able to gainfully use these data sets is a challenge, especially when they are used to answer online queries which are generated rapidly. Even linear time algorithms are too slow in this context, and therefore sublinear time algorithms become critical. In many applications, queries to large data sets typically expect the data set to satisfy some structural property that makes it useful. For example, users of a data set may expect that a set of points is in convex position, that a list of numbers is sorted, or that a large graph is a tree. These properties are sensitive to noise, and even a small amount of corruption could destroy the usefulness of the data set. Suppose we performed binary search on an array that is not

completely sorted: even a few values out of place could make the binary search process completely fail.

Because of various difficulties in gathering data on such a large scale, or just because of external noise, the data set may actually fail to satisfy the desired property. The extensive theory of property testing algorithms (see surveys [13, 14, 25]) provides means to detect such problems in existing data sets in sublinear time. However, in applications it may not suffice to just detect such problems, after all, one needs to actively process the data to perform useful tasks. Usually, it is reasonable to minimally modify the data to restore the property, so that further processing can take place without problems. However, since the input is so massive, one cannot afford to read the whole input and then fix it, particularly since this may require global changes.

Thus, this may seem like an impossible task, but usually access to the large input is provided in the form of local queries: for example, if the input is, say, the web graph, then typical queries to the input would ask for all the outgoing links from a given webpage. In such cases, we would like to repair the input as and when we read it in *sublinear time*. More precisely, we seek to design a *filter* algorithm that sits between users of the input and the input itself. The filter accepts queries and actively repairs the input while answering the queries. This was the motivation for the definition of *online property reconstruction* model in [1] (see also [27, 10] for more applications).

In this paper, we are concerned with online sublinear reconstruction of expanders. The input is a huge bounded degree graph represented by adjacency lists. The graph is supposed to be an *expander*, i.e. for any subset of vertices of at most half the size of the graph, the number of outgoing edges (i.e., to the complement of the subset) is a significant fraction of the total number of edges incident on the vertices in the set. The queries are in the form of requests for the adjacency list of a specified vertex. The filter’s job is to answer such queries in sublinear time, such that the final picture is that of a graph that really is an expander, while adding as few edges as possible to make it into an expander.

We show that expander reconstruction is connected to some intriguing questions about random walks in “noisy” expanders. Suppose one is given a graph G that consists of a large expander connected arbitrarily to some small unknown graph (the noise). The graph G is probably not an expander, but we would like to show that random walks on the expander part are not greatly affected by the noise. Turning this around, we can ask: what portion of the large expander is affected by this noise? Can we still hope that from almost all of the expander, we can reach a vast majority of the expander by short random walks? We prove the rather strong statement that the noise can affect a part of the expander that is proportional in size to the noise. We believe that this result should be useful in other contexts as well, and we present it in the general setting of arbitrary irreducible Markov chains rather than for our specific application.

For the reconstruction problem, we design a sublinear time procedure based on random walks that can very accurately distinguish between vertices in a high expansion part of the graph, and vertices of the noisy part. This procedure essentially measures the distance between the probability distribution induced by a random walk on an undirected graph to the stationary distribution. Previous algorithms for measuring such distances [8] are not efficient enough for our situation, and we design new tools for these scenarios. Using our general result on the noise-tolerance of expanders, we show that the number of vertices from which a random walk mixes poorly is a good measure of the combinatorial distance (in terms of edges changed) of a graph to being an expander, thus establishing a link between these two different notions of distance.

Our implementation of the filter algorithm allows it to answer queries in a *distributed fashion*. This is a very useful property, in settings where we imagine the filter to be a query processing engine for a giant database, and there are many independent users who all want to have access to the database. This model was first discussed in [27]. The ability to answer queries in a distributed manner is certainly more difficult than sequential answering, when the filter may be able to store a limited amount of history.

1.1. Online Sublinear Reconstruction of Expanders

The input is a large graph $G = (V, E)$, with n vertices. We assume that all the vertices have degrees bounded by some specified constant d , which we assume to be sufficiently large (say at least 10). The graph G is represented by *adjacency lists*: for every vertex v , there is list of vertices (of size at most d) adjacent to v . Given a subset of vertices $S \subseteq V$ of size at most $n/2$, let $\bar{S} = V \setminus S$, and let $E(S, \bar{S})$ denote the set of edges crossing the cut (S, \bar{S}) .

The expansion of the cut is defined to be $|E(S, \bar{S})|/|S|$. The expansion of the graph is the minimum expansion over all cuts in the graph. An expander is a generic term for a graph with high expansion.

We consider a related parameter, the conductance of a cut, which is defined to be just the expansion of the cut divided by $2d$. The conductance of the graph is its expansion divided by $2d$. We are given a parameter ϕ , and the input graph is supposed to (but may not) have the property that its conductance is at least ϕ .

We now explain the problem of sublinear reconstruction of expanders, in the framework of the distributed reconstruction model of [27]. Given an input graph G , we wish to construct a graph G' that has high expansion, and differs from G as little as possible. Moreover, we want to construct G' through a procedure called a *filter* which provides the interface to G' by outputting a list of the (at most) d neighbors of a given input vertex v in G' . Each such query needs to be processed in *sublinear* time in the size of the graph. Note that there is *no preprocessing* involved. A conductance parameter ϕ is specified in advance, and we wish to make the conductance of G' as close to ϕ as we can, while modifying G as little as possible.

In addition, we aim for a *distributed* implementation of the filter, which can handle any number of parallel queries and yet maintain consistency of answers across queries. Each query is handled by a separate invocation of the filter which has access to a *shared* random seed s of *sublinear size*. Conditioned on the seed, the filter is a deterministic procedure, i.e. it employs no other source of randomness. Our implementation should guarantee that with high probability over the choice of the random seed s , the final graph G' (obtained after all vertices have been queried) has degrees bounded by d , conductance at least ϕ' which is as close to ϕ as possible, and changes G in at most $\alpha \cdot \text{OPT}$ edges where OPT is the optimal number of edges needed to be changed to make the conductance at least ϕ , and α is the approximation factor.

We now state our main theorem regarding the properties of our filter:

Theorem 1 *There is a deterministic procedure RECONSTRUCT that takes as input a vertex v and a random seed s of length¹ $\tilde{O}(\sqrt{n}/\phi^2)$, and outputs the adjacency list of v in G' . All the following properties simultaneously hold with probability at least $1 - 1/n^2$ over the choice of the random seed s :*

1. *Each adjacency list is output in $\tilde{O}(\sqrt{n}/\phi^2)$ time,*
2. *All outputs are consistent: the vertex u is output as a neighbor of v iff v is output as a neighbor of u ,*

¹In this paper, we use the \tilde{O} notation to suppress dependence on poly $\log(n)$ factors.

3. The final graph G' has conductance at least $\phi' = \Omega(\phi^2 / \log n)$ and degree bound d ,
4. The number of edges changed is at most $O(\frac{1}{\phi} OPT)$, where OPT is the optimal number of edges needed to be changed to make the conductance at least ϕ .

Note that the distributed filter can be used as a property tester for expansion, so by the lower bound for testing expansion by Goldreich and Ron [18], the running time of our filter is optimal up to poly $\log(n)$ factors.

1.2. Comparison to Property Testing

In *property testing* [15, 26], we wish to accept inputs that satisfy some given property, and reject those that are sufficiently “far” from having that property. A vast array of combinatorial and graph properties [3, 4, 11, 15] are indeed testable without even reading the entire input. The problem of online reconstruction is closely connected to property testing: indeed, a distributed filter can be used to estimate the distance to a given problem in sublinear time, thereby, giving a property testing (and even a *tolerant tester* [24]).

The input model of adjacency lists for sparse graphs was introduced by Goldreich and Ron [18], where they designed testers for a large set of problems. More general results about the testability of large classes of properties were obtained by Czumaj and Sohler [11] and by Benjamini, Schramm, and Shapira [9]. The technique of using random walks for testing was first introduced by Goldreich and Ron [16] for testing bipartiteness.

Goldreich and Ron [17] formally posed the problem of expansion testing (by a result in [18], there is a lower bound of $\Omega(\sqrt{n})$ queries to the input graph). Given a d -degree bounded input graph G and a distance parameter ε , we want to distinguish between the case that G is an expander, and that at least εnd edges need to be changed in G to make it an expander². The first progress towards this was made by Czumaj and Sohler [12]. Further work on testing expansion improving certain parameters was done in [19, 23].

The reconstruction problem is a much harder problem than property testing for expansion for two reasons: first, the property testing problem is a decision problem, whereas reconstruction algorithm needs to actively take local action in each query to fix the input graph, and patch up sparse cuts. Second, in the notation of Theorem 1, property testing algorithms only need to distinguish between graphs which have $OPT = 0$ and $OPT \geq \varepsilon nd$, whereas reconstruction algorithms, via their guarantee on the number of edges changed, actually approximate OPT , for all values of OPT (although the difficulty of this task is mitigated somewhat since the guarantee is only required to hold after *all* the vertices have been queried).

²This is not the complete formulation, but it suffices for this discussion.

Thus, the known techniques in property testing algorithms are inadequate for reconstruction. Such approaches show that in a graph G that is far from being an expander, there exist many “bad” vertices starting from which random walks in G do not mix rapidly. One may expect that adding d edges at random to all such bad vertices will suffice to make it an expander, and indeed it does, but it may be an overkill: there are graphs G where all vertices are bad, yet G can be made an expander by adding very few edges.

We therefore need to devise completely new techniques for reconstruction. The main approach to reconstruction is to identify *weak* and *strong* vertices. Intuitively, weak vertices are those that lie on the smaller side of bad cuts, and therefore need edges to be added from them, whereas strong vertices are those that are part of a large expanding component (if one exists). The reconstruction procedure should be careful to only add edges to the weak vertices.

The property testing approach of distinguishing between weak and strong vertices by thresholding the distance from mixing for a short random walk fails: one can easily construct counter-examples for any threshold. Furthermore, the strongest property testing results are obtained using the \mathcal{L}_2 -distance, but this is too sensitive to noise, harming the mixing properties of many strong vertices. Thus, in this paper, we develop completely new techniques based on the \mathcal{L}_1 -distance, and prove noise-tolerance properties of Markov chains under this norm. These properties are crucial in our reconstruction algorithm.

2. Noise-tolerance of Markov Chains

First, we discuss our theorem about the noise-tolerance of Markov chains. Later, we apply this to our setting of expansion reconstruction. Let M be a finite Markov chain with state set V and transition probabilities p_{uv} for $u, v \in V$. The k -step transition probabilities will be denoted as p_{uv}^k . The vector p_u^k represents the probability distribution on V for a k -step random walk starting from u (for $k = 0$, $p_{uu}^0 = 1$ and $p_{uv}^0 = 0$ for $v \neq u$). For simplicity, we assume that M is irreducible, and that for any $u \in V$, $p_{uu} \geq 1/2$ (so that the chain is aperiodic). In this case, there is a unique stationary distribution π for the Markov chain. For a subset of states $S \subseteq V$, define $\pi(S) = \sum_{u \in S} \pi_u$. Note that the chain need not be reversible.

The conductance of the Markov chain is defined to be the largest number ϕ such that for any subset of states $S \subseteq V$,

$$\sum_{u \in S, v \in V \setminus S} \pi_u p_{uv} \geq \phi \cdot \min\{\pi(S), \pi(V \setminus S)\}.$$

It is a well-known fact (see, for example [22, 20, 28]) that if the Markov chain has high conductance, then from any starting state, the chain converges to the stationary distribu-

tion exponentially fast at a rate determined by the conductance.

Suppose we are given a Markov chain that *almost* has high conductance, in the following sense. There is a “large” subset of states $V' \subseteq V$ such that the chain restricted to the subset V' has high conductance, and nothing can be said about the remaining “small” part of the state space, $B = V \setminus V'$ (the “noise”). Then, we would like to show that except for a set of states B' of stationary measure comparable to that of B , from all other starting states the chain will have some fast mixing properties (i.e. the noise has limited influence).

This turns out to be hard to prove because it can be true for completely different reasons. Suppose that the noise B is almost disconnected from V' . Then, a random walk starting from V' will almost never leave V' and since V' has high conductance, will definitely mix rapidly inside V' . On the other hand, suppose that the whole chain has high conductance (not just restricted to V'). In this case, although walks from inside V' will encounter B , all walks will still mix rapidly. We need a proof that can interpolate between these scenarios. A first approach to proving this would be to estimate the probability that a walk from V' never hits B . But such a bound would be too weak: there are many states (compared to $\pi(B)$) which are sufficiently “close” to B that a random walk from them will almost certainly hit B .

We now proceed to formalize the setting. Given a subset of states $V' \subseteq V$, define the V' -conductance of the chain to be the largest number ϕ such that for any set $S \subseteq V'$,

$$\sum_{u \in S, v \in V' \setminus S} \pi_u p_{uv} \geq \phi \cdot \min\{\pi(S), \pi(V' \setminus S)\}.$$

We also need the notion of a *uniform averaging walk* in the Markov chain of ℓ steps: in such a walk, we choose a number $k \in \{0, 1, 2, \dots, \ell - 1\}$ uniformly at random, and stop the chain after k steps. Thus, the distribution of the final state of the uniform averaging walk starting from u is $\bar{p}_{uv}^\ell = \sum_{k=0}^{\ell-1} p_{uv}^k$. The *total variation distance* between two distributions ξ and ψ on the states is defined to be $\|\xi - \psi\|_{TV} = \max_S |\xi(S) - \psi(S)| = \frac{1}{2} \|\xi - \psi\|_1$, where S is an arbitrary subset of states, and $\xi(S)$ and $\psi(S)$ are, respectively, the measures of S under ξ and ψ respectively.

Theorem 2 *Let the Markov chain M have V' -conductance ϕ . Let $\pi_0 = \min\{\pi_u/\pi(V') : u \in V'\}$. Then, there is a set B' such that $\pi(B') \leq 2\pi(B)$ with the property that starting from any state $s \in V \setminus B'$, if the uniform averaging Markov chain of $\ell \geq 8 \log(1/\varepsilon\pi_0)/(\varepsilon\phi^2)$ steps is run, then the final probability distribution \bar{p}_s^ℓ satisfies $\|\bar{p}_s^\ell - \pi\|_{TV} \leq \varepsilon + \pi(B)$.*

We prove this theorem as follows. We consider a closely related Markov chain that is restricted to V' . We retain all the original transitions in M between any pair of vertices

$u, v \in V'$ with the same probabilities. We assign all such transitions a cost of 1. The meaning of this cost will be explained later.

For any pair of vertices $u, v \in V'$, and for every integer $j \geq 2$, define q_{uv}^j to be the total probability of all length j walks from u to v all of whose states, except for the end points u and v , are in B . Then we add a new transition e_{uv}^j from u to v with cost j and probability q_{uv}^j . Since M is irreducible, any walk in M that enters B has to eventually leave it, and hence the new transitions define a Markov chain on V' . Call this new Markov chain M' . Since M is irreducible, so is M' . The chain M' is called an *induced* Markov chain by Aldous and Fill [2].

Now, for any walk in M' , define the cost of the walk to be the total cost of all transitions in the walk. The cost of a walk in M' is naturally mapped to the length of a corresponding walk taken in M (where taking one of the new transitions is to be interpreted as taking *all* the corresponding walks of length equal to the prescribed cost which are entirely in B , except for the end points).

This correspondence between walks in M and M' also implies that the stationary distribution in M' is one that assigns probability $\pi'_u = \pi_u/\pi(V')$ to state u . This can be seen by considering an arbitrary state $z \in V'$ and using the fact that the stationary probability of any state $u \in V'$ is proportional to the expected number of visits to u before returning to z , and then using the correspondence between the walks to argue that the expectation is the same in both M and M' . For convenience, for $u \in B$, we define $\pi'_u = 0$.

Lemma 1 *For any integer $t > 0$, there exists a set $\tilde{B} \subseteq V'$ such that $\pi(\tilde{B}) \leq \pi(B)$, with the property that for all $v \in V' \setminus \tilde{B}$,*

$$\mathbb{E}[\text{cost of } t \text{ step walk in } M' \text{ from } v] \leq 2t.$$

PROOF: Fix any starting state $s \in V'$. Let X^k be the k -th state on a t step walk in M' from s . For any $u \in V'$, let p_{su}^k be the probability of reaching u from s on step k of a random walk in M' .

For any $u \in V'$, we have $\mathbb{E}[\text{cost of step } k + 1 \mid X^k = u] = \mathbb{E}[\text{cost of one step from } u]$, by the Markov property. So define

$$\begin{aligned} c_u &:= \mathbb{E}[\text{cost of one step from } u] \\ &= \sum_{v \in V'} \left(p_{uv} + \sum_{j \geq 2} q_{uv}^j \cdot j \right). \end{aligned}$$

We have

$$\begin{aligned}
& \mathbb{E}[\text{cost of } t \text{ step walk from } s] \\
&= \sum_{k=0}^{t-1} \mathbb{E}[\text{cost of step } k+1] \\
&= \sum_{k=0}^{t-1} \sum_{u \in V'} \mathbb{E}[\text{cost of step } k+1 \mid X^k = u] \cdot p'_{su}{}^k \\
&= t + \sum_{k=0}^{t-1} \sum_{u \in V'} (c_u - 1) \cdot p'_{su}{}^k.
\end{aligned}$$

Now, we define

$$\tilde{B} := \left\{ s \in V' : \sum_{k=0}^{t-1} \sum_{u \in V'} (c_u - 1) \cdot p'_{su}{}^k \geq t \right\}. \quad (1)$$

With this definition, it is clear that for any starting state $s \in V' \setminus \tilde{B}$, the expected cost of a t step walk from s is at most $2t$, as required by the statement of the lemma. We proceed to bound $\pi(\tilde{B})$ as follows:

$$\begin{aligned}
t \cdot \pi(\tilde{B}) &\leq \sum_{s \in V'} \pi_s \cdot \left[\sum_{k=0}^{t-1} \sum_{u \in V'} (c_u - 1) \cdot p'_{su}{}^k \right] \\
&= \sum_{k=0}^{t-1} \sum_{u \in V'} (c_u - 1) \cdot \sum_{s \in V'} \pi_s p'_{su}{}^k \\
&= \sum_{k=0}^{t-1} \sum_{u \in V'} (c_u - 1) \cdot \pi_u \\
&= t \cdot \sum_{u \in V'} \pi_u (c_u - 1).
\end{aligned}$$

where the equality on the second line follows because $\pi' = \pi' P'^k$, where P' is the transition matrix of M' , which implies that $\sum_{s \in V'} \pi'_s p'_{su}{}^k = \pi'_u$, and because $\pi'_u \propto \pi_u$ for $u \in V'$.

We now need to bound $\sum_{u \in V'} \pi_u (c_u - 1)$. Note that c_u is exactly the expected time to return to the set V' starting from u in the original Markov chain M . Since M is irreducible, by Kac's lemma (see [2]), we have $\sum_{u \in V'} \pi_u c_u = 1$, and hence $\sum_{u \in V'} \pi_u (c_u - 1) = 1 - \pi(V') = \pi(B)$. Thus, we have $t \cdot \pi(\tilde{B}) \leq t \cdot \pi(B)$, which implies that $\pi(\tilde{B}) \leq \pi(B)$. \square

Now, we would like to prove that the set $B' = \tilde{B} \cup B$, where \tilde{B} is defined in Lemma 1 works for Theorem 2, when t is chosen to be the mixing time of M' , which can be bound in terms of the conductance of M' . The idea is that even without the newly added transitions, the Markov chain V' has high conductance. Thus, if a short walk in V' mixes, then by Lemma 1, a corresponding short walk in the original chain should mix as well.

PROOF: [Theorem 2]

We use the notion of *stopping rules* for Markov chains [21]. A stopping rule is a rule that observes the walk and tells us whether to stop or not, depending on the walk so far (but independently of the continuation of the walk). This decision may be reached using coin flips, so the stopping rule has to only specify for each finite walk w , the probability of continuing the walk, so that with probability 1, the walk is stopped in a finite number of steps. The expected time for a stopping rule to terminate the walk yields bounds on the mixing time of the chain.

Let $B' = \tilde{B} \cup B$, where the set \tilde{B} is as defined in equation (1) in Lemma 1. Let $s \in V \setminus B' = V' \setminus \tilde{B}$. We consider a random walk in the original Markov chain M starting from s with the following probabilistic stopping rule Γ : stop the walk as soon as it has taken t steps in the induced walk in M' . Note that this stopping rule always stops the walk on some state in V' . Denote by $\mathbb{E}_s[\Gamma]$ the expected number of steps the walk takes starting from s before being terminated by Γ .

Now, because the Markov chain M has V' -conductance at least ϕ , the Markov chain M' has conductance at least ϕ , and hence by the results of [22, 20], a $t = 2\lceil \log(1/\varepsilon\pi_0) \rceil / \phi^2$ step walk starting from s mixes on V' , i.e. $\|p'_s{}^t - \pi'\|_{\text{TV}} \leq \varepsilon/2$, where $p'_s{}^t$ is the probability vector for a t step random walk starting at s . Furthermore, we have $\|\pi' - \pi\|_{\text{TV}} = \pi(B)$, and hence by the triangle inequality, $\|p'_s{}^t - \pi\|_{\text{TV}} \leq \varepsilon/2 + \pi(B)$. Now, by Lemma 1, we have $\mathbb{E}_s[\Gamma] \leq 2t$. Thus, for all $s \in V \setminus B'$, the stopping rule Γ stops the walk in a distribution that is within total variation distance $\varepsilon/2 + \pi(B)$ of the stationary distribution within an expected $2t$ steps.

Lovász and Winkler [21] define $\mathcal{H}(s, d_\delta)$ to be the minimal expected time for a stopping rule to stop a chain started from s in a distribution that is within variation distance δ of the stationary distribution, and thus we have shown that $\mathcal{H}(s, d_{\varepsilon/2 + \pi(B)}) \leq 2t$. Now, Theorem 4.22 in [21] implies that for the uniform averaging chain of ℓ steps in M started from s , if \bar{p}_s^ℓ is the final distribution, then

$$\begin{aligned}
\|\bar{p}_s^\ell - \pi\|_{\text{TV}} &\leq \varepsilon/2 + \pi(B) + \frac{1}{\ell} \mathcal{H}(s, d_{\varepsilon/2 + \pi(B)}) \\
&\leq \varepsilon + \pi(B),
\end{aligned}$$

since $\ell \geq 4t/\varepsilon$, which completes the proof. \square

3. The Reconstruction Algorithm

We give an overview of the reconstruction algorithm. In the first step, the algorithm tries to identify vertices that are part of a low conductance cut (a ‘‘bad’’ cut), and in the second step, it attempts to boost the conductance of the cut by adding edges to the identified vertices. We use a random walk based procedure to separate vertices on the smaller

side of a bad cut from the larger. The details of this separation procedure are given in Section 3.1.

One way to fix the bad cuts is to add random edges to the identified vertices. This works, but doesn't allow for distributed query processing which requires the randomness to be fixed in advance. Instead, we use an explicit expander, and construct a hybrid of the original graph and the expander. The hybridization is locally done to allow distributed query processing. Details of the hybridization procedure are given in Section 3.2.

3.1. Separating vertices

We define a Markov chain on the graph, and study random walks in this Markov chain. The states of the chain are the vertices of the graph, and the edges represent transitions, such that from each vertex u , any outgoing edge (u, v) is taken with probability $p_{uv} = 1/2d$. With the remaining probability, the walk stays at the current vertex. Note that this is an irreducible (assuming the graph is connected), aperiodic, and time-reversible chain, and the stationary distribution is uniform on all the vertices.

Let $\ell = c \log(n)/\phi^2$, where c is a sufficiently large enough constant. We will consider the *uniform averaging walk* of length ℓ , in two forms based on related probabilistic stopping rules:

1. Pick an integer $t \in \{0, 1, 2, \dots, \ell - 1\}$ uniformly at random, and stop the walk after t steps. This is the uniform averaging walk of length ℓ . Let q_{uv} be the probability of reaching v from u after such a random walk (i.e. $q_{uv} = \bar{p}_{uv}^\ell$, in the notation of Section 2).
2. Pick two integers $t_1, t_2 \in \{0, 1, 2, \dots, \ell - 1\}$ uniformly at random, and stop the walk after $t_1 + t_2$ steps. Let Q_{uv} be the probability of reaching v from u after such a random walk.

Let q_u be the probability vector of q_{uv} 's. For a subset of vertices S , let $q_u(S) := \sum_{v \in S} q_{uv}$ (similarly we define Q_u and $Q_u(S)$). It is easy to see that for any two vertices u and v , we have $Q_{uv} = \sum_w q_{uw}q_{wv} = \sum_w q_{uw}q_{vw}$, because $q_{wv} = q_{vw}$, as all edges have the same probability of $1/2d$. For ease of notation, we will refer to the above walks as q -random walks and Q -random walks. In our procedures and definitions, we will use some constants: c is a sufficiently large integer and $\alpha, \beta, \gamma, \delta > 0$ are sufficiently small (values $\alpha = 1/1000, \beta = 1/10, \gamma, \delta = 1/100$ work³).

We now define *weak* and *strong* vertices. Intuitively, strong vertices are those that can reach a vast majority of vertices with probability $\Omega(1/n)$ through a short random walk. Formally, we will look at the mixing properties of

random walks in the \mathcal{L}_1 norm. Note that there can be vertices that are neither strong nor weak. In the following definition, $\vec{1}$ is the all 1's vector.

Definition 1 (Strong and Weak vertices)

1. A vertex $u \in V$ is called *strong* if

$$\|q_u - \vec{1}/n\|_{TV} \leq \alpha.$$

2. A vertex $u \in V$ is called *weak* if

$$\|Q_u - \vec{1}/n\|_{TV} \geq 1/4.$$

Intuitively, strong vertices are those that reach a vast majority of vertices (through short q -walks) with probability $\Omega(1/n)$. The basic idea is to perform many q -walks from u (and v) and compute the number of collisions between these walks (at the endpoints): this is a twist on the idea of Goldreich and Ron [17] for using random walks to estimate the mixing of a random walk. A birthday paradox like argument suggests that $O(\sqrt{n})$ walks should be sufficient to estimate probabilities of $\Omega(1/n)$. Unfortunately, assuming nothing about the vectors q_u and q_v , we cannot get a reasonable bound on the variance of our randomized estimate⁴. For this reason, we define the *reduced collision probability*, which disregards collisions that occur due to vertices reached with extraordinarily high probability. This quantity attempts to approximate Q_{uv} and can be estimated in sublinear time. For the purpose of separating weak vertices from strong ones, it suffices to estimate these probabilities.

Definition 2 (Reduced Collision Probability) For a given pair of vertices u, v , define the set $S_u = \{w : q_{uw} \leq 1/\sqrt{n}\}$ and similarly the set S_v . The reduced collision probability of vertices u, v (denoted by r_{uv}) is:

$$r_{uv} = \sum_{w \in S_u \cap S_v} q_{uw}q_{vw}$$

We now describe a procedure SEPARATE that distinguishes weak vertices from strong ones. It uses a subroutine ESTIMATE-RCP, that outputs accurate estimates \hat{r}_{uv} of the actual r_{uv} values.

SEPARATE (Input: Vertex $u \in V$)

1. Choose a random set R of $c \log n/\gamma$ vertices.
2. For every $v \in R$, run ESTIMATE-RCP(u, v), $c \log n$ times. If for a majority of these runs, ESTIMATE-RCP does not abort and outputs $\hat{r}_{uv} \geq (1 - \beta)(1 - \delta)/n$, then call v *accessible*.
3. If more than a $(1 - 2\gamma)$ -fraction of vertices in R are accessible, then accept u . Otherwise reject u .

⁴As an extreme example: suppose for some w , $q_{uw} = 1 - 1/n - 1/n^2$, and $q_{uz} = 1/n^2$ for $z \neq w$; whereas all $q_{vz} = 1/n$. Although $Q_{uv} = \Omega(1/n)$, it is very unlikely that a sublinear time procedure can detect a collision between q -random walks from u and v .

³We have not optimized these constants.

Lemma 2 *The procedure SEPARATE runs in $O(\sqrt{n\ell} \log^2 n)$ time, and with probability at least $1 - n^{-3}$ has the following behavior:*

1. Assume there are at least $(1 - \gamma)n$ strong vertices, for some constant γ . If u is strong, then the algorithm accepts u .
2. If u is weak, then the algorithm rejects u .

PROOF SKETCH: For any two strong vertices u, v , we can show that $r_{uv} \geq (1 - \beta)/n$, for $\beta \geq 3\sqrt{\alpha}$. Thus, if there are at least $(1 - \gamma)n$ strong vertices, then a majority of nodes in the sample are strong, and hence the first part follows. As for the second part, if u is a weak vertex, then for most vertices v , $r_{uv} \ll 1/n$, and thus u is rejected with high probability. \square

We now describe the ESTIMATE-RCP procedure. It uses a simple procedure FIND-SET (described in the full version of the paper) which, for any given u , outputs an approximation to the set S_u . The following Lemma 3 is not *exactly* true, but it conveys the correct meaning.

ESTIMATE-RCP (*Input:* Vertices $u, v \in V$)

1. Let $\hat{S}_u := \text{FIND-SET}(u)$, and $\hat{S}_v := \text{FIND-SET}(v)$.
2. Keep performing q -random walks from u until $m = \sqrt{n}/\delta^2$ such walks end at vertices in \hat{S}_u (call this set of walks W_u). If more than $20m$ walks are performed, then ABORT. Repeat for walks starting from v as well to obtain the set of m walks W_v .
3. Let X be the number of pairwise collisions between walks in W_u and W_v (if a walk from W_u and a walk from W_v end at the same vertex, then this counts as a pairwise collision). Output X/m^2 .

Lemma 3 *Let u and v be two vertices. The running time of ESTIMATE-RCP is $O(\sqrt{n\ell} \log n)$. If both $q_u(S_u), q_v(S_v) \geq 1/2$, then ESTIMATE-RCP aborts with probability less than $e^{-O(\sqrt{n})}$. If ESTIMATE-RCP does not abort, then it outputs an estimate \tilde{r}_{uv} such that*

$$\Pr[|\tilde{r}_{uv} - r_{uv}| > \delta \max\{r_{uv}, 1/2n\}] \leq 1/10.$$

PROOF SKETCH: Assume that \hat{S}_u, \hat{S}_v are exactly S_u, S_v and that ESTIMATE-RCP does not abort. The probability of a collision between walks started from u and v at a vertex in $S_u \cap S_v$ is exactly r_{uv} . Since the r_{uv} values are based on collisions at only those vertices that are not reached with extraordinarily high probability, we can show that the variance of our estimate is sufficiently low, and then the high probability bound follows by an application of Chebyshev's inequality. \square

3.2. Hybridizing the graph with an expander

Now that we have a separating procedure, we can describe the actual reconstruction procedure RECONSTRUCT that hybridizes an expander with our original graph. Given a query vertex v , the procedure will output at most d vertices which will be the neighbors of v in the reconstructed graph. We will refer to the final reconstructed graph as G' , and for cuts (S, \bar{S}) in G' , we use the notation $E'(S, \bar{S})$ to refer to the set of edges crossing the cut.

Since we are describing a distributed filter, we assume we have access to a sublinear sized random seed s of size $\tilde{O}(\sqrt{n})$ which is fixed for all queries. Since the total number of calls to SEPARATE is at most $O(n)$ (for each call to the RECONSTRUCT, we will make $O(1)$ calls to SEPARATE), by taking a union bound over all the error probabilities, we can ensure that the guarantees of Lemma 2 hold with probability at least $1 - 1/n^2$. Since the seed is fixed, we can unambiguously refer to vertices as *accepted* or *rejected*, based on SEPARATE.

For constructing G' , we use an explicit bounded degree edge expander G^* with n vertices. The explicit construction allows us to find all neighbors of a vertex $v \in G^*$ in $\text{poly}(\log n)$ time. The precise expansion property of G^* we need is the following:

Property 1 *The expander G^* has degree bound $d/2$. For any set of vertices S in G^* with $|S| \leq n/2$, we have $|E^*(S, \bar{S})| \geq \eta|S|d$, where η is a constant, and $E^*(S, \bar{S})$ is the set of edges crossing the cut (S, \bar{S}) .*

Naturally, there is a one-to-one correspondence between the vertices of G and G^* . Abusing notation, given a vertex v in G , we will refer to the corresponding vertex in G^* also as v . To prevent confusion about edges, we will call edges G, G^* , or G' -edges depending on the graph in consideration.

For the sake of intuition, we can think of the accepted vertices as strong, and the rejected as weak. The reconstructed graph G' will be a careful combination of G and G^* . Starting with G , here is an informal description of how G' is built. For any rejected vertex v , we remove all G -edges incident to v and add all G^* -edges incident to v to get the G^* -edges. This is to ensure that any subset of rejected vertices will have large conductance. For accepted vertices, we would like to just keep the same G -edges.

This construction could potentially make some (accepted) vertices have a degree larger than d . Therefore, we have to remove some G -edges incident to accepted vertices to maintain the degree bound. These edges are removed based on a simple *local* rule. Note that this choice cannot be arbitrary, since we want a distributed filter (for example, if the G -edge (u, v) is removed on querying v , then it must also be removed on querying u). Unfortunately, this might

affect the conductance of subsets of accepted vertices. We ensure that every time we remove such an edge, we replace it by a very short path (again decided by local considerations) between the endpoints without affecting the degree bound. This gives us G' with the desired properties. Because we want every query to be handled in sublinear time, we give a procedure that determines the neighbors of a vertex in G' by running SEPARATE on a constant number of vertices.

Lemma 4 *The procedure RECONSTRUCT has the following properties:*

- (a) *It runs in $O(\sqrt{nd}d^3 \log^2(n))$ time and needs a random seed of $O(\sqrt{nd}d^3 \log^2(n) \log(d))$ bits.*
- (b) *Its outputs are consistent over all queries (i.e. vertex v is output as a neighbor of u (in G') iff u is output as a neighbor of v).*
- (c) *The final graph G' has degree bound d .*
- (d) *Any cut in G' has at least half the conductance of the same cut in G .*
- (e) *Any weak vertex has all its G^* -neighbors adjacent in G' .*

We assume that there is some global ordering of the vertices (say, according to the value of their indices). Thus, given any vertex v , there is an *ordered* list of the neighbors of v , with possibly some “null” entries at the end (because v might have degree less than d). When we refer to the i^{th} vertex in some list of vertices, we mean the i^{th} vertex in the list in the global ordering.

As mentioned before, if any vertex u is rejected by SEPARATE, we remove all the G -edges incident on it and replace them by G^* -edges. To avoid increasing the degree of accepted vertices, we need to remove some G -edges between accepted vertices as well. We now describe a procedure that given an edge $e = (u, v)$ of G , where u and v are both accepted by SEPARATE, outputs whether the edge needs to be removed, and if so, what edges are added.

REMOVE (*Input: Edge $(u, v) \in G$*)

1. Find the G^* -neighbors of u and v and consider all the rejected vertices (according to SEPARATE).
2. Suppose v is the i^{th} neighbor of u , and u is the j^{th} neighbor of v . Let s be the i^{th} rejected G^* -neighbor of u , and let t be the j^{th} rejected G^* -neighbor of v . Note that one or both of s and t could be null.
3. If both s and t are null, then (u, v) is not removed. If t is null, then the edges (s, u) and (s, v) replace (u, v) . If s is null, then the edges (t, u) and (t, v) replace (u, v) . If neither s nor t is null, then the edges (u, s) , (s, t) , (t, v) replace (u, v) . The edge (s, t) is added parallel to any existing (s, t) edges. Finally, output the new neighbors to u and v .

Using the procedure REMOVE, we can describe the main reconstruction procedure, which outputs all the G' -neighbors of an input vertex v .

RECONSTRUCT (*Input: Vertex $v \in V$*)

1. If v is accepted by SEPARATE:
 - (a) For all the rejected G -neighbors u of v , remove the edge (u, v) .
 - (b) For all the accepted G -neighbors u of v , call REMOVE on the edge (u, v) .
 - (c) For all the rejected G^* -neighbors u of v , add the edge (u, v) , if not added previously.
2. If v is rejected by SEPARATE:
 - (a) Remove all the G -edges incident on v .
 - (b) For all G^* -neighbors u of v , add the edge (u, v) . If u is accepted by SEPARATE, then let v be the i^{th} rejected G^* -neighbor of u . Let w be the i^{th} G -neighbor of u . If w is not null, and is accepted by SEPARATE, call REMOVE on (u, w) .
3. Output the final neighbors of v .

PROOF: [Lemma 4]

The running time bound (item (a)) follows from Lemma 2 because we run SEPARATE on at most $O(d^3)$ nodes. Item (e) follows directly from the specification of RECONSTRUCT.

We now prove item (c). If v is accepted by SEPARATE, then for every new edge (u, v) added to v for a G^* -neighbor u of v that is rejected by SEPARATE, either we have an space for an extra edge (because w is null in step 2(b)), or we call REMOVE on (v, w) . The call to REMOVE will remove edge (v, w) and the only edge incident to v that it will insert is (u, v) (that was added anyway). Thus, the degree v remains bounded by d . If v is rejected by SEPARATE, then for every G^* -neighbor u of v , we add at most 2 edges to v (see step 3. of REMOVE). Thus, the G' -degree of v is at most d .

As for item (d): for some cut, let (u, v) be a cut edge. If edge (u, v) is removed by REMOVE, it is replaced by a path from u to v in G' . Furthermore, for two different edges incident to u , these paths are edge-disjoint. In the worst case, the path from u to v in G' might use an edge that is already present in G . Nonetheless, the number of edges incident to u crossing the cut in G' will be at least half of the number in G .

Finally, we prove item (b). This is a matter of verifying the details. Suppose u is an accepted vertex and let v be output as a G' -neighbor of u . The neighbor v came about either because it is a rejected G^* -neighbor of v or because of a call to REMOVE on a G -edge (u, w) for some accepted node w . In the former case, when RECONSTRUCT is called on v , we will add the edge (u, v) in step 2(b), and thus u

is output as a neighbor of v . In the latter case, either v is accepted or rejected. If v is accepted, then $w = v$ and the edge (u, v) is a G -edge that is untouched by REMOVE, and thus u is output as neighbor of v when RECONSTRUCT is called on v . Otherwise, if v is a rejected node, then when RECONSTRUCT is called on v , in step 2(b) REMOVE will be called on (u, w) , which outputs u as a G' -neighbor of v .

Similar arguments show consistency of the output in case u is a rejected vertex as well. \square

4. Bounds on Number of Edges Changed and Conductance

We now bound the number of edges changed by RECONSTRUCT in terms of the optimal number of edges to be changed to make the conductance at least ϕ . Theorem 3 gives such a bound and thus establishes a link between two notions of measuring the “distance” to having a large conductance: number of edges that needed to be changed, and the number of vertices from which the random walk mixes poorly (i.e. the weak vertices).

Theorem 3 RECONSTRUCT achieves an approximation ratio of $O(1/\phi)$ to the optimal number of edges to be changed to make the conductance of the graph at least ϕ .

This follows from the following lemma that shows that there is a large cut of low conductance:

Lemma 5 Let S be the set of strong vertices. Then there is a cut $(B, V \setminus B)$ such that $\min\{|B|, |V \setminus B|\} \geq \Omega(n - |S|)$, with conductance less than $\phi/2$.

PROOF SKETCH: We keep recursively partitioning the graph G , finding cuts in the remaining induced graph of conductance less than $\phi/2$, and aggregating these cuts. To be more precise: start out with $B = \{\}$. Let $\bar{B} = V \setminus B$. If there is a cut (T, \bar{T}) in \bar{B} with $|T| \leq |\bar{B}|/2$ having conductance less than $\phi/2$, then we set $B := B \cup T$, and continue, as long as $|B| \leq n/2$. It is easy to check that the final cut (B, \bar{B}) also has conductance less than $\phi/2$.

If $|B| \geq \frac{\alpha}{2}n$, then note that $|B| \leq (\frac{1}{2} + \frac{\alpha}{4})n$, and we are done since $\min\{|B|, |V \setminus B|\} \geq \Omega(n) \geq \Omega(n - |S|)$.

Otherwise, $|B| < \frac{\alpha}{2}n$ and the subgraph induced on \bar{B} has conductance at least $\phi/2$. The graph G is basically a noisy expander. We apply Theorem 2 to G , to conclude that there is a set B' , $|B'| \leq 2|B|$, such that a q -random walk starting from any vertex $s \in V \setminus B'$ gets to within total variation distance α of the uniform distribution, or, in other words, $V \setminus B' \subseteq S$. Thus,

$$\min\{|B|, |V \setminus B|\} = |B| \geq \Omega(n - |S|).$$

\square

Now we can prove Theorem 3:

PROOF: [**Theorem 3**]

Lemma 5 immediately implies that the optimal reconstruction of the graph must add at least $\Omega(\phi d(n - |S|))$ edges to patch up the cut $(B, V \setminus B)$. Whenever the reconstruction algorithm adds an edge, then one of the endpoints is a rejected vertex. The total number of removed edges is at most the number of added edges. Therefore, we can bound the total change (up to constant factors) by d times the number of rejected vertices. Suppose the number of strong vertices is less than $(1 - \gamma)n$. The graph G is trivially changed by at most $O(nd) = O(d(n - |S|))$ edges. If the number of strong vertices is more than $(1 - \gamma)n$, then by Lemma 2 all strong vertices are accepted. Our reconstruction algorithm adds at most $O(d(n - |S|))$ edges, which means that we have an approximation ratio of $O(1/\phi)$ to the optimal number of edges to be changed. \square

Now we give bounds on the conductance of G' .

Theorem 4 The reconstructed graph G' has conductance at least $\Omega(\phi^2 / \log n)$.

PROOF SKETCH: Let S be a set of nodes with $|S| \leq n/2$. If it has at least $(1 - \eta/2)|S|$ weak nodes, then all the G^* -edges incident on these weak nodes are present in G' . By property 1 of G^* , even after accounting for the edges of G^* incident on the weak nodes which end up inside S itself, at least $\eta d|S|/4$ such edges cross the cut (S, \bar{S}) , thus making its conductance $\Omega(1)$.

Otherwise, if S has less than $(1 - \eta/2)|S|$ weak nodes, then the conductance must already be $\eta/16\ell$ in G (and hence, by Lemma 4(d), the cut has conductance at least $\eta/32\ell$ in G'): otherwise, using an argument similar to the one in Lemma 4.7 in [12], with probability at least $3/4$, from at least $(1 - \eta/2)|S|$ starting vertices in S , the Q -random walk, which has length at most 2ℓ , never even exits the set S . This implies that all such starting vertices are weak, a contradiction. \square

5. Further Directions of Research

One of the main direction of future research is improvement of the conductance bound of G' to $\Omega(\phi^2)$ (instead of $\Omega(\phi^2 / \log n)$). For this, we need to have definitions of weak/strong that distinguish vertices on the basis of very small probability differences (a total variation distance of the order of $1/n^\epsilon$). These differences can be algorithmically tested, but to ensure that not too many edges are added to get G' , we would need much stronger results about walks in noisy expanders.

It seems highly likely that the algorithmic procedures we use here could be used for efficient graph partitioning algorithms [5, 6, 7]. The partitions would be decided by performing random walks from vertices, and might lead to

easier proofs of earlier results. We may also be able to generate sublinear routines which can implicitly represent such a partition: answering queries such as whether two vertices are in the same graph of the partition or not. Improved conductance bounds for reconstruction may lead to better graph partitioning algorithms.

6. Acknowledgements

The authors would like to thank Reid Andersen, Itai Benjamini, Bernard Chazelle, Fan Chung, László Lovász, Eyal Lubetzky, Elchanan Mossel, Dana Randall and Shang-Hua Teng for helpful discussions.

References

- [1] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Property preserving data reconstruction. *Proc. of 15th ISAAC*, pages 16–27, 2004.
- [2] D. J. Aldous and J. Fill. *Time-reversible Markov chains and random walks on graphs*. (book in preparation).
- [3] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties : it’s all about regularity. *Proc. 38th STOC*, pages 251–260, 2006.
- [4] N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. *Proc. 46th FOCS*, pages 429–438, 2005.
- [5] R. Andersen. A local algorithm for finding dense subgraphs. *Proc. of 19th SODA*, pages 1003–1009, 2008.
- [6] R. Andersen, F. R. K. Chung, and K. Lang. Local graph partitioning using pagerank vectors. *Proc. of 47th FOCS*, pages 475–486, 2006.
- [7] R. Andersen and K. Lang. An algorithm for improving graph partitions. *Proc. of 19th SODA*, pages 651–660, 2008.
- [8] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. *Proc. of 41st FOCS*, pages 259–269, 2000.
- [9] I. Benjamini, O. Schramm, and A. Shapira. Every minor-closed property of sparse graphs is testable. *arxiv*, arXiv:0801.2797v2, 2008.
- [10] B. Chazelle and C. Seshadhri. Online geometric reconstruction. *Proc. of 22nd SOCG*, pages 386–394, 2006.
- [11] A. Czumaj and C. Sohler. On testable properties in bounded degree graphs. *Proc. 18th SODA*, pages 494–501, 2007.
- [12] A. Czumaj and C. Sohler. Testing expansion in bounded degree graphs. *Proc. 48th FOCS*, pages 570–578, 2007.
- [13] E. Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of EATCS*, 75:97–126, 2001.
- [14] O. Goldreich. Combinatorial property testing - a survey. *Randomization Methods in Algorithm Design*, 75:45–60, 1998.
- [15] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.
- [16] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.
- [17] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. *ECCC*, TR00-020, 2000.
- [18] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.
- [19] S. Kale and C. Seshadhri. Testing expansion in bounded degree graphs. In *Proc. of the 35th ICALP*, pages 527–538, 2008.
- [20] L. Lovász and M. Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *Proc. of 31st FOCS*, pages 346–354, 1990.
- [21] L. Lovász and P. Winkler. Mixing times. *Microsurveys in Discrete Probability, DIMACS Series in Discrete Math. and Theor. Comp. Sci., AMS*, pages 85–133, 1998.
- [22] M. Mihail. Conductance and convergence of markov chains—a combinatorial treatment of expanders. In *Proc. of 30th FOCS*, pages 526–531, 1989.
- [23] A. Nachmias and A. Shapira. Testing the expansion of a graph. *ECCC*, TR07-118, 2007.
- [24] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *ECCC*, TR04-010, 2004.
- [25] D. Ron. Property testing. *Handbook on Randomization*, II:597–649, 2001.
- [26] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.
- [27] M. Saks and C. Seshadhri. Parallel monotonicity reconstruction. *Proc. of 19th SODA*, pages 962–971, 2008. Full version is titled “Distributed monotonicity reconstruction”.
- [28] A. Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. *Combinatorics, Probability & Computing*, 1:351–370, 1992.