A General Framework for Graph Sparsification

Wai Shing Fung
University of Waterloo
Ontario N2L 3G1
wsfung@uwaterloo.ca

Ramesh Hariharan
Strand Life Sciences
Bangalore 560024
ramesh@strandls.com

Nicholas J. A. Harvey
University of Waterloo
Ontario N2L 3G1
harvey@uwaterloo.ca

Debmalya Panigrahi
CSAIL, MIT
Cambridge, MA 02139
debmalya@mit.edu

ABSTRACT

We present a general framework for constructing cut sparsifiers in undirected graphs — weighted subgraphs for which every cut has the same weight as the original graph, up to a multiplicative factor of $(1 \pm \epsilon)$. Using this framework, we simplify, unify and improve upon previous sparsification results. As simple instantiations of this framework, we show that sparsifiers can be constructed by sampling edges according to their *strength* (a result of Benczúr and Karger), *effective resistance* (a result of Spielman and Srivastava), *edge connectivity*, or by sampling *random spanning trees*. Sampling according to edge connectivity is the most aggressive method, and the most challenging to analyze. Our proof that this method produces sparsifiers resolves an open question of Benczúr and Karger.

While the above results are interesting from a combinatorial standpoint, we also prove new algorithmic results. In particular, we develop techniques that give the first (optimal) $O(m)$-time sparsification algorithm for unweighted graphs. Our algorithm has a running time of $O(m) + \tilde{O}(n/\epsilon^2)$ for weighted graphs, which is also linear unless the input graph is very sparse itself. In both cases, this improves upon the previous best running times of $O(m \log^2 n)$ (for the unweighted case) and $O(m \log^3 n)$ (for the weighted case) respectively. Our algorithm produces sparsifiers containing $O(n \log n/\epsilon^2)$ edges in expectation; the only known construction of sparsifiers with fewer edges is by a substantially slower algorithm running in $O(n^3 m/\epsilon^2)$ time.

A key ingredient of our proofs is a natural generalization of Karger's bound on the number of small cuts in an undirected graph. Given the numerous applications of Karger's bound, we suspect that our generalization will also be of independent interest.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Theory, Algorithms

Keywords

Graph Sparsification, Edge Connectivity, Sampling

1. INTRODUCTION

Can any dense graph be approximated by a sparse graph? Surprisingly, the answer is a resounding "yes", under a variety of notions of approximation. For example, given any undirected graph,there are sparse subgraphs that approximate *all* pairwise distances up to a multiplicative and/or additive error (see [22] and subsequent research on *spanners*), *every* cut to an arbitrarily small multiplicative error [3, 4] (called *cut sparsifiers*), every eigenvalue to an arbitrarily small multiplicative error [2, 25, 26, 27] (called *spectral sparsifiers*), and so on. Such approximations are a cornerstone of numerous important results in theoretical computer science.

In this work, we consider the problem of approximating every cut arbitrarily well; this problem was originally studied by Karger [10, 11] and Benczúr and Karger [3, 4]. They proved that every undirected graph with n vertices and m edges (and potentially non-negative weights on its edges) has a subgraph with only $O(n \log n/\epsilon^2)$ edges (and a different set of weights on those edges) such that, for every cut, the weight of the cut in the original graph and its subgraph agree up to a multiplicative factor of $(1 \pm \epsilon)$. Such a subgraph is called a *cut sparsifier*, or simply a *sparsifier*. Benczúr and Karger also gave a randomized algorithm to construct a sparsifier in $O(m \log^2 n)$ time for unweighted graphs and $O(m \log^3 n)$ time for weighted graphs. Their result has now become a standard tool with widespread use in the design of fast algorithms relating to cuts and flows [3, 4, 5, 13, 15, 18, 24].

Spielman and Teng [27] realized that a stronger notion of sparsification would be useful for efficiently solving systems of linear equations defined by Laplacian matrices. They defined a *spectral sparsifier* to be a weighted subgraph such that the quadratic forms defined by the Laplacians of these two graphs agree up to a multiplicative factor of $(1 \pm \epsilon)$. Spectral sparsifiers are also cut sparsifiers, as can be seen by evaluating these quadratic forms at $\{0,1\}$-vectors. An efficient algorithm to construct a spectral sparsifier with $O(n \log n/\epsilon^2)$ edges in expectation was given by Spielman and Srivastava [25]; using later improvements to linear system solvers [16], this algorithm runs in $O(m \log^3 n)$ time. Furthermore, a spectral sparsifier with only $O(n/\epsilon^2)$ edges can be computed in $O(n^3 m/\epsilon^2)$ time [2].

The Benczúr-Karger and Spielman-Srivastava sampling schemes follow the same basic approach. First, they replace each edge e of weight w_e in the input graph G by w_e parallel unweighted edges.[1] Now, each unweighted edge is sampled independently with probability $p_e = \min\{\rho/\lambda_e, 1\}$ for some parameters ρ, λ_e; if chosen, the weight of edge e is increased in the sparsifier G_ϵ by $1/p_e$. Both algorithms choose $\rho = \Theta(\log n/\epsilon^2)$, but differ in their choice of λ_e.

In order to describe their respective choice of parameters λ_e, we require some definitions. For an edge (s,t), the (local) *edge connectivity* between s and t, denoted k_{st}, is defined to be the minimum weight of a cut that separates s and t. The *effective conductance* of edge (s,t), denoted c_{st}, is the amount of current that flows when each edge e of weight w_e is viewed as a resistor of value $1/w_e$ and a unit voltage difference is imposed between s and t. The *effective resistance* of (s,t) is $1/c_{st}$. A *k-strong component* of G is a maximal k-edge-connected, vertex-induced subgraph of G. The *strength* of edge (s,t), denoted k'_{st}, is the maximum value of k such that a k-strong component of G contains both s and t. Informally, all three of k_{st}, c_{st} and k'_{st} measure the connectivity between s and t.

Benczúr and Karger require $\lambda_e \leq k'_e$, whereas Spielman and Srivastava require $\lambda_e \leq c_e$. These hypotheses are incomparable since k'_{st} can be $\Omega(n)$ times larger than c_{st} or vice versa. However $k_{st} \geq \max\{c_{st}, k'_{st}\}$ always holds.

Sampling by Edge Connectivities. The primary objective of this paper is to consider the more aggressive regime of sampling according to edge connectivities, i.e., $\lambda_e \leq k_e$. In fact, Benczúr and Karger [4] conjectured that such a sampling scheme would also produce sparsifiers, and this would result in a simpler analysis and simpler algorithms. Our work proves this conjecture. Theorem 1.1 is a succinct corollary of our main theorem; more general results are described in Section 2.

Theorem 1.1. *Let G_ϵ be obtained from a weighted graph G by independently sampling edge e with probability $p_e = \rho/\lambda_e$, where $\rho = \Theta(\log^2 n/\epsilon^2)$ and $\lambda_e = k_e$. Then, G_ϵ contains $O(n \log^2 n/\epsilon^2)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ whp.*[2][3]

Since $k_e \geq \max\{c_e, k'_e\}$, our aggressive sampling scenario subsumes the scenarios of Benczúr-Karger and of Spielman-Srivastava, the main caveat being that Spielman and Srivastava prove spectral sparsification whereas we do not. On top of unifying these results, we also extend our technique to obtain a general sparsification framework and set out sufficient conditions for a sampling scheme to result in good sparsifiers.

[1]We assume throughout that all edge weights are integers.

[2] $G_\epsilon \in (1 \pm \epsilon)G$ will denote that G_ϵ approximates every cut in G to within a multiplicative factor of $(1 \pm \epsilon)$.

[3] A property is said to hold *with high probability* (or *whp*) if it *does not* hold with probability inverse polynomial in n.

This lets us show that some other natural sampling schemes also yield sparsifiers.

Sampling by Random Spanning Trees. Can we set $\rho = o(\log n)$ in the above sampling schemes? Unfortunately not. To see this, consider a clique of n vertices — if $\rho = o(\log n)$ and $\lambda_e = k_e$ then with probability tending to 1 the sampled graph would be disconnected and hence not approximate the original graph. Such examples also show that the Benczúr-Karger and Spielman-Srivastava algorithms require $\Omega(n \log n)$ edges.

One way to circumvent these examples is via dependent sampling, such as sampling spanning trees. This idea was explored by Goyal et al. [7] and was the key approach in the recent progress on ATSP [1]. Suppose we sample ρ uniformly random spanning trees. Then the sampled graph is certainly connected after choosing just one tree. Furthermore, sampling uniformly random spanning trees is closely related to sampling according to effective conductances, which leads to the following theorem.

Theorem 1.2. *Let G be a weighted graph. Let G_ϵ be the union of $\rho = \Theta(\log^2 n/\epsilon^2)$ uniformly random trees where each edge is assigned weight c_e/ρ. Then G_ϵ has $O(n \log^2 n/\epsilon^2)$ edges and $G_\epsilon \in (1 \pm \epsilon)G$, whp.*

Surprisingly, we cannot take $\rho = o(\log n)$ here either. For any constant $c \geq 1$, if we wish to approximate all cuts to within a factor c, we show in section 6 that the sampling process of Theorem 1.2 requires $\rho = \Omega(\log n)$.

1.1 Sparsification Algorithms

Our framework yields sparsification algorithms that are not only simpler, but also faster. By a slight modification of known techniques [4], we can easily estimate the edge connectivities k_e and derive a linear-time algorithm that produces sparsifiers with $O(n \log^2 n/\epsilon^2)$ edges. This simple result is stated below as Theorem 1.3. A stronger result is given by Theorem 1.4, in which a more sophisticated approach is used to construct sparsifiers with $O(n \log n/\epsilon^2)$ edges in $O(m) + \tilde{O}(n/\epsilon^2)$ time.

Sampling by Nagamochi-Ibaraki indices. Nagamochi and Ibaraki devised a very simple method that finds good estimates to all edge connectivities. Their method simply partitions the graph into a sequence of maximal spanning forests. It can be implemented in $O(m)$-time for unweighted graphs [21], and $O(m + n \log n)$-time for weighted graphs [20].

More formally, a set of edge-disjoint spanning forests $T_1, T_2, \ldots, T_k$ of a graph G is said to be a *Nagamochi-Ibaraki (NI) forest* packing if T_i is a spanning forest on the edges left in G after removing those in $T_1, T_2, \ldots, T_{i-1}$. For weighted graphs, an edge with weight w_e must appear in w_e contiguous forests. The *NI index* of edge e, denoted ℓ_e, is the index of the *last* NI forest in which e appears. We obtain the following theorem as a simple instantiation of our general framework.

Theorem 1.3. *Let G_ϵ be obtained from a weighted graph G by independently sampling edge e with probability $p_e = \rho/\lambda_e$, where $\rho = \Theta(\log n/\epsilon^2)$ and $\lambda_e = \ell_e$. Then, G_ϵ contains $O(n \log^2 n/\epsilon^2)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ whp. Moreover, this algorithm runs in $O(m)$ time.*

Linear-time Sparsification Algorithm. We improve the above algorithm further in the next theorem.

Theorem 1.4. *There is an algorithm that produces sparsifiers containing $O(n \log n/\epsilon^2)$ edges in expectation, and runs in $O(m)$ time for unweighted graphs and $O(m) + \tilde{O}(n/\epsilon^2)$ time for weighted graphs.*

Note that this algorithm has optimal time complexity for unweighted graphs; for weighted graphs, the time complexity is slightly sub-optimal if the input graph is already very sparse. The previous best time complexity for an identical guarantee on the size of the sparsifier was $O(m \log^2 n)$ for unweighted graphs, and $O(m \log^3 n)$ for weighted graphs [4]. On the other hand, the only known algorithm that constructs sparsifiers with fewer edges takes $O(n^3 m/\epsilon^2)$ time [2], which is substantially slower. Our sparsification algorithm improves the running time for the numerous applications of sparsifiers for dense input graphs (e.g. [13, 15, 18, 24]).

1.2 Cut counting

An important ingredient in our proofs is an extension of Karger's random contraction algorithm for computing global minimum cuts [9, 14]. We give a variant of this algorithm that interleaves random edge contractions with edge *splitting-off* operations. The main purpose is to prove a generalization of the following cut counting theorem.

Theorem 1.5 (Karger [9, 14]). *For any $\alpha \geq 1$, the number of cuts of weight at most αK in an undirected weighted graph is at most $n^{2\alpha}$, where K is the minimum weight of a cut in the graph.*

To state our generalization, we need some definitions. An edge is said to be *k-heavy* if the edge connectivity of its endpoints is at least k; otherwise, it is said to be *k-light*. The *k-projection* of a cut is the set of k-heavy edges in it. Intuitively, we show that the large number of cuts of size αK for large α, as predicted by Karger's theorem, arises from *many* distinct k-projections of these cuts for small values of k, while there are *few* distinct k-projections of these cuts for large values of k.

Theorem 1.6. *Let $G = (V, E)$ be a weighted, undirected graph. For any k and any $\alpha \geq 1$, the number of distinct k-projections in cuts of weight at most αk is at most $n^{2\alpha}$.*

(Note that this theorem reduces to Karger's cut counting theorem by setting k to the weight of a global minimum cut.) Given the numerous applications of Karger's theorem, e.g. [1, 6, 12, 23], we suspect our generalization may be of further interest.

Roadmap. The next section contains an overview of the techniques used in obtaining the various results outlined above. Section 3 contains a proof of the cut counting theorem (Theorem 1.6), which is used in the proofs of the general framework presented in Section 4. We use the general framework to obtain Theorem 1.1 in Section 4.1. We present the linear-time sparsification algorithm in Section 5. Finally, some sampling lower bounds are presented in Section 6.

2. OVERVIEW OF OUR TECHNIQUES

Our first goal is to demonstrate sampling using edge connectivities, thereby proving Theorem 1.1. The basic intuition behind sparsification is two-fold:

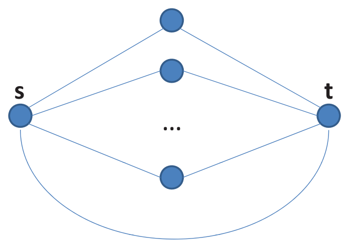

Figure 1: An example of a graph where Karger's cut counting theorem is not sufficient to prove that sampling using edge connectivities yields a good sparsifier.

1. Edges that are *well-connected*[4] are less critical to maintaining connectivity of the graph and can hence be sampled at lower probabilities than those that are not well-connected.

2. Most edges in a dense graph are well-connected; hence, most edges can be sampled at low probabilities leading to a sparse sample.

Now, suppose we want to sample edge e at probability $p_e = \Theta(\log n/k_e)$ and give it a weight of $1/p_e$ in G_ϵ if selected. The next lemma formalizes (2).

Lemma 2.1. *For any undirected, weighted graph $G = (V, E)$ where w_e and k_e respectively represent the weight and connectivity of edge e, $\sum_{e \in E} \frac{w_e}{k_e} \leq n - 1$.*

Formalizing (1) turns out to be more tricky. For example, consider the complete graph on n vertices. Here, all edges have connectivity $n - 1$, and therefore are sampled at probability $\Theta(\log n/n)$. Now, consider a cut containing Δ edges. Since edges are sampled independently, Chernoff bounds (see e.g. [19]) ensure that the *failure probability* for this cut (i.e., the probability that the sampled weight of the cut is not in $(1 \pm \epsilon)\Delta$) is $1/n^{\Omega(\Delta/n)}$. If $\Delta = \Theta(n)$, this bound is inverse polynomial in n. But there are exponentially many cuts; so a naive union bound that multiplies this probability by the number of cuts will not work.

A slightly more refined analysis would observe that there are only $n^{O(\Delta/n)}$ cuts with Δ edges, either by a direct counting argument or by applying Karger's cut counting theorem (Theorem 1.5). Since each such cut has failure probability $1/n^{\Omega(\Delta/n)}$, we can apply a union bound for each value of Δ. Summing over all values of Δ gives an overall failure probability that is inverse polynomial in n.

Unfortunately, this technique does not work for all graphs. For example, consider the graph in Figure 1. Here, the connectivity of each edge is two, except that of the (s, t) edge is $\Theta(n)$. Now, consider any cut separating s and t in this graph. The (s, t) edge has the lowest sampling probability ($= \Theta(\log n/n)$), and therefore has high variance in the sampled weight even though all the other edges have low variance. Unfortunately, the Chernoff bound does not recognize this difference in variance between the edges and yields a failure probability inverse polynomial in n. This is too weak, for

[4]The exact definition of *well-connectedness* varies from one sparsification scheme to another.

we have to union bound over the exponentially many cuts separating s and t. The problem lies in the use of the Chernoff bound — in spite of most edges in the cut being sampled at a relatively high probability (thereby reducing the variance of the sample), the Chernoff bound is very weak. To overcome this problem, we partition the edges of a cut in doubling ranges $[2^{i-1}, 2^i - 1]$ of their connectivity, and apply Chernoff bounds on each of these sets separately. Since edges in any such set (call the set of edges having connectivity in the range $[2^{i-1}, 2^i - 1]$ the *i-segment* of the cut) are sampled at roughly the same probability, the bounds obtained are tighter, especially for small values of i. We run into a technical hurdle here. Consider the example in Figure 1. The (s,t) edge is in a connectivity range on its own, and clearly one cannot obtain tight concentration bounds for just one edge. However, observe that whether this single edge appears in the sample has almost no bearing on the quality of the sample. To formalize this intuition, we will use the following generalization of the Chernoff bound.

Theorem 2.2. *Let $X_1, X_2, \ldots, X_n$ be n random variables such that X_i takes value $1/p_i$ with probability p_i and 0 otherwise. Then, for any p such that $p \leq p_i$ for each i, any $\epsilon \in (0,1)$, and any $N \geq n$, the following bound holds:*[5]

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} X_i - n\right| > \epsilon N\right] < 2e^{-0.38\epsilon^2 pN}.$$

When we apply the above theorem to the i-segment of a cut C, we set N to the weight of the cut w_C, thereby obtaining a meaningful tail bound. For example, in Figure 1, if a segment comprises the solitary (s,t) edge, we define the failure event as a deviation of ϵn from the expected value of one, thereby ensuring that the (overwhelmingly probable) event of not including the edge (s,t) in the sample is *not* defined as a failure event. One deficiency of this approach is that the deviation is ϵw_C for each connectivity range, leading to an overall deviation of $\epsilon w_C \log n$. However, recall that the total deviation needs to be at most ϵw_C. So we can instead set the value of N to $w_C / \log n$ and, to ensure that the probability bound remains unchanged, increase the sampling probability by a factor of $\log n$. (We call this extra $\log n$ in the sampling probability the *overlap overhead*). We now use Theorem 1.6 to bound the failure probability over i-segments of all cuts of weight Δ. Finally, we union bound over all values of k and Δ to obtain Theorem 1.1.

As observed previously, Theorem 1.1 implies that sampling using edge strengths (Benczúr-Karger [4]) and effective resistances (Spielman-Srivastava [25]) yields sparsifiers. Since every edge e has NI index $\ell_e \leq k_e$, Theorem 1.1 also implies that sampling using NI indices yields a sparsifier. However, since the sampling probability is $\Theta(\log^2 n/\epsilon^2 \lambda_e)$ (for $\lambda_e = k'_e, c_e, \ell_e$ respectively), the resulting sparsifiers have $O(n \log^2 n/\epsilon^2)$ edges, whereas the Benczúr-Karger sparsifiers have only $O(n \log n/\epsilon^2)$ edges.

Next we describe our general framework which has several applications, including constructing sparsifiers with only $O(n \log n/\epsilon^2)$ edges under the sampling scheme of Benczúr and Karger (we omit the details of this construction due to space limitations), and proving Theorem 1.3.

[5]For any event $\mathcal{E}$, $\mathbb{P}[\mathcal{E}]$ represents the probability of event $\mathcal{E}$.

2.1 The General Framework

Consider the proof sketch for Theorem 1.1 outlined above. As a first abstraction, note that our argument does not depend on the exact value of the overlap overhead, i.e. the proof sketch continues to hold with $N = w_C/\alpha$ and $p_e = \Theta(\alpha \log n/\epsilon^2 2^i)$ (where e has connectivity in $[2^{i-1}, 2^i - 1]$) for any value α of the overlap overhead. Generalizing further, suppose we identify a subset of edges C_i (with weight w_{C_i}) in every cut C such that:

- Each edge in the i-segment of a cut is 2^i-heavy in a graph containing only the edges C_i for all cuts C.
- Each edge appears in C_i for at most α different values of i.

Then, $N = w_{C_i}/\alpha$ and $p_e = \Theta(\alpha \log n/\epsilon^2 2^i)$ are sufficient for the above proof. Moreover, we do not need to define i-segments by edge connectivities, rather we can define the i-segment of cut C as the set of edges sampled with probability $p_e \in \left[\frac{\alpha \log n}{\epsilon^2 2^i}, \frac{\alpha \log n}{\epsilon^2 (2^{i+1}-1)}\right]$. The proof sketch is valid provided the above two properties are satisfied by i-segments defined in this manner.

We now formalize the above intuition. Let $G = (V, E)$ be an undirected graph where edge e has weight w_e. Consider any $\epsilon \in (0,1)$. We construct a sparse graph G_ϵ where the weight of edge e is R_e/p_e, R_e being an independent (of other edges) binomial random variable with parameters w_e and p_e.[6] What values of p_e result in a sparse G_ϵ that satisfies $G_\epsilon \in (1 \pm \epsilon)G$ whp? Let $p_e = \min\left\{\frac{96\alpha \ln n}{0.38 \lambda_e \epsilon^2}, 1\right\}$, where α is independent of e and λ_e is some parameter of e satisfying $\lambda_e \leq 2^n - 1$. The exact choice of values for α and the λ_e's will vary from application to application. However, we describe below a sufficient condition that characterizes a *good* choice of α and λ_e's.

To describe this sufficient condition, partition the edges in G according to the value of λ_e into sets $F_0, F_1, \ldots, F_k$ where $k = \lfloor \lg \max_{e \in E}\{\lambda_e\} \rfloor \leq n-1$ and $e \in F_j$ iff $2^j \leq \lambda_e \leq 2^{j+1} - 1$. Now, let $\mathcal{G} = (G_0, G_1, G_2, \ldots, G_i, \ldots, G_k)$ (where $G_i = (V, E_i)$) be a set of subgraphs of G (we allow edges of G to be replicated multiple times in the G_i's) such that $F_i \subseteq E_i$ for every i. For a set of parameters $\pi = (\pi_0, \pi_1, \ldots, \pi_k)$, $\mathcal{G}$ is said to be a *(π, α)-certificate* corresponding to the above choice of α and λ_e's if the following properties are satisfied:

- **π-connectivity.** For $i \geq 0$, any edge $e \in F_i$ is π_i-heavy in G_i.
- **α-overlap.** For any cut C of weight w_C in G, let $e_i^{(C)}$ be the weight of edges that cross C in G_i. Then, for all cuts C, $\sum_{i=0}^{k} \frac{e_i^{(C)} 2^{i-1}}{\pi_i} \leq \alpha w_C$.

Theorem 2.3 describes the sufficient condition. We gave the intuition behind the theorem at the beginning of this section; a formal proof appears in Section 4.

Theorem 2.3. *If there exists a (π, α)-certificate for a particular choice of α and λ_e's, then $G_\epsilon \in (1 \pm \epsilon)G$ with probability*

[6]This is equivalent to taking w_e unweighted copies of e, sampling each copy independently with probability p_e and adding a weight of $1/p_e$ to edge e in G_ϵ for each copy selected in the sample.

at least $1 - 4/n$*. Furthermore* G_ϵ *has* $O(\frac{\alpha \log n}{\epsilon^2} \sum_{e \in E} \frac{w_e}{\lambda_e})$ *edges in expectation.*

2.2 Sparsification Algorithms

Our first algorithmic application of the general framework is to show that the expected size of the sparsifier obtained when sampling by NI indices[7] is $O(n \log^2 n/\epsilon^2)$. This proves Theorem 1.3. In this sampling scheme, λ_e is the NI index of edge e. Our key observation is that any edge in NI forests $T_{2^i}, T_{2^i+1}, \dots, T_{2^{i+1}-1}$ is 2^{i-1}-heavy in a graph $G_i = (V, E_i)$ containing all edges in $T_{2^{i-1}}, T_{2^{i-1}+1}, \dots, T_{2^i}, \dots, T_{2^{i+1}-1}$ (i.e., two successive doubling ranges of NI forests). This lets us define $\pi_i = 2^{i-1}$ and $\alpha = 2$ since each edge is present in at most two E_i's. Since the graphs G_i are a (π, α)-certificate for this choice of parameters, we use Theorem 2.3 to conclude that the expected size of the sparsifier is $O(n \log^2 n/\epsilon^2)$.

Our goal now is to improve the size of the sparsifier to $O(n \log n/\epsilon^2)$ while maintaining linear running time. To this end, we abstractly view the NI index-based sampling scheme as an iterative algorithm that finds a set of edges E_i in iteration i (these are the edges in NI forests $T_{2^i}, T_{2^i+1}, \dots, T_{2^{i+1}-1}$ and are sampled with probability $\Theta(\log n/2^i)$) with the following properties:

- **(P1)** Each edge in E_i has connectivity of $\Theta(2^i)$ in E_{i-1}.
- **(P2)** The number of edges in E_i is $\Theta(n \cdot 2^i)$.

Our first observation is that property **(P1)** can be weakened — using the general framework, we show it is sufficient for each edge in E_i to have connectivity of $\Theta(2^i)$ in $G_{i-1} = (V, F_{i-1})$ where $F_{i-1} = E_{i-1} \cup E_i \cup \dots$. Since we are aiming for a sparser sample than in the previous algorithm, we also need to make **(P2)** stricter. Our new requirement is that the number of edges in E_{i-1} from any connected component $\mathbf{C}$ of G_{i-1} is $O(2^i)$ times the number of components into which $\mathbf{C}$ decompose in G_i. It is not difficult to show that this stricter condition ensures that the expected number of edges in G_ϵ decreases to $\Theta(n \log n/\epsilon^2)$.

To complete the description of the algorithm, we need to give a linear-time construction of E_i's satisfying the above properties. Iteration i runs on each component of G_i separately; we describe the algorithm for any one component $\mathbf{C}$. First, $(2^i + 1)$ NI forests $T_1, T_2, \dots, T_{2^i+1}$ are constructed in $\mathbf{C}$ and all edges in T_{2^i+1} are contracted; let the resulting graph be $G_\mathbf{C} = (V_\mathbf{C}, E_\mathbf{C})$. If $|E_\mathbf{C}| = O(|V_\mathbf{C}| \cdot 2^i)$, we add the edges in $E_\mathbf{C}$ to E_i and retain the remaining edges for iteration $i + 1$. Otherwise, we construct $(2^i + 1)$ NI forests on $G_\mathbf{C}$, contract the edges in the $(2^i + 1)$st NI forest, and update $G_\mathbf{C}$ to this contracted graph. We repeat these steps until $|E_\mathbf{C}| = O(|V_\mathbf{C}| \cdot 2^i)$; then, we add the edges in $E_\mathbf{C}$ to E_i and retain the remaining edges for iteration $i + 1$. One may verify that properties **(P1)** and **(P2)** are satisfied by the E_i's constructed by this algorithm.

This algorithm, with a pre-processing step where the number of edges is reduced to $\tilde{O}(n)$ by sampling using NI indices, runs in $O(m) + \tilde{O}(n)$ time, and yields a sparsifier of expected size $O(n \log n/\epsilon^2)$. This is already optimal for all but very sparse input graphs. We need one additional idea to turn this into a strictly linear-time algorithm for unweighted graphs. Observe that we would ideally like to place as many edges as we can in subsets E_i for large values of i so as to obtain a sparse G_ϵ. On the other hand, the fact that these edges are retained till the later iterative stages implies that we pay for them in our time complexity repeatedly. To overcome this dilemma, we use the following trick: instead of sampling these edges with probability $1/2^i$ in iteration i, we sample them with probability 1/2 in each iteration $j < i$, and retain them in the set of edges for the next iteration only if selected in the sample. Now, we are able to reduce the size of our edge set by a factor of two (in expectation) in each iteration; therefore, implementing a single iteration in linear time immediately yields a linear time algorithm overall. However, this iterative sampling scheme creates several technical hurdles since it introduces dependencies between the sampling processes for different edges. Our key technical contribution is in showing that these dependencies are mild enough for us to continue to use the framework that we have developed above for independent sampling of edges. We present the details of this algorithm and its analysis in Section 5.

[7] Supposing that $w_e = n^{O(1)}$, one would obtain a weaker bound of $O(n \log^3 n/\epsilon^2)$ edges from a straightforward application of Theorem 1.1 and the previously known fact [4] that $\sum_e w_e/\ell_e = O(n \log \sum_e w_e)$.

Weighted Graphs. Weighted graphs pose additional challenges. First, consider a graph with super-polynomial edge weights. An immediate concern for such graphs is that the number of doubling categories of sampling probabilities can now be polynomial rather than logarithmic. For example, in Theorem 1.1, the overlap overhead will now be polynomial instead of logarithmic.

We have two techniques, each separately solving this problem. First, a more refined use of the Chernoff bound allows us to, roughly speaking, show that only $\log n$ doubling categories have substantial contribution to the sampled weight of the cut. The analysis is intricate; so we omit the details from this extended abstract. Second, we can use the *windowing technique* due to Benczúr and Karger [4]. The basic idea is that if the connectivity of an edge e is k_e, then removing all edges with weight at most k_e/n^3 and contracting all edges with weight greater than k_e does not significantly alter the connectivity of e. This trick lets us deal with only a polynomial range of edge weights at any point of time, thus alleviating the problem described above. For algorithmic applications, we also need to estimate the edge connectivities for making these alterations to the input graph. The key observation (due to Benczúr and Karger) is that a *maximum spanning tree* can be used to obtain a polynomial approximation to the connectivity values, and this is sufficient for our purpose.

Edge weights also introduce complications in the analysis of running times. Can we sample a weighted edge in $O(1)$ time? Recall (from the general framework) that sampling an edge e involves the generation of a binomial random variable with parameters w_e and p_e. This can be done in $O(w_e p_e)$ time for edge e (see e.g. [8]), and therefore $O(\sum_{e \in E} w_e p_e)$ time overall. It can be verified that this time complexity is asymptotically identical to the bound on the size of the sparsifier obtained from Theorem 2.3 for the general framework, and therefore can be ignored in the running time analysis.

Finally, we note that the linear-time sparsification algorithm for unweighted graphs takes $O(m) + O(n \log^{3.5} n)$ time on weighted graphs (details omitted due to space limitations).

Algorithm 1 An algorithm for finding a small k-projection by splitting off light vertices.

```
procedure Contract($G,\ k,\ \alpha$)
input: A graph $G = (V, E)$, a parameter $k \geq K$ where $K$ is the weight of a minimum cut in $G$, and an approximation factor $\alpha$
output: a $k$-projection
While there exists a $k$-light vertex $v$
    Perform admissible splitting-off at $v$ until $v$ becomes an isolated vertex
    Remove $v$
While there are more than $\lceil 2\alpha \rceil$ vertices remaining
    Pick an edge $e$ uniformly at random
    Contract $e$ and remove any self loops
    While there exists a $k$-light vertex $v$
        Perform admissible splitting-off at $v$ until $v$ becomes an isolated vertex
        Remove $v$
Output the $k$-projection of a cut selected uniformly at random
```

3. CUT COUNTING

In this section,[8] we will prove Theorem 1.6. Our proof strategy, as outlined in the introduction, is to give an algorithm (Algorithm 1) with the following property, which immediately implies Theorem 1.6. Here, $q(F)$ denotes the minimum weight of a cut whose k-projection is F.

Theorem 3.1. *For any k-heavy set of edges F with $q(F) \leq \alpha k$, Algorithm 1 outputs F with probability at least $n^{-2\alpha}$.*

To describe Algorithm 1, we need some additional definitions. A vertex is said to be *k-heavy* if it is incident to a k-heavy edge; otherwise, it is *k-light*. The algorithm adds new edges to G; for notational convenience, we will call these edges k-light irrespective of their connectivity. Therefore, the k-projection of a cut does not include any of these edges.

Note that when k is the minimum weight of a cut in G, there is no k-light vertex and Algorithm 1 reduces to Karger's random contraction algorithm. The main idea is that we can remove the k-light vertices while preserving the connectivities of all k-heavy edges by using the *splitting-off* operation. This operation replaces a pair of edges (u, v) and (v, w) with the edge (u, w), and is said to be *admissible* if it does not change the edge connectivity between any two vertices $s, t \neq v$. A deep theorem of Mader [17] asserts that admissible splitting-off exists under mild assumptions.

Theorem 3.2 (Mader [17]). *Let $G = (V, E)$ be a connected graph where $v \in V$ is a vertex which has degree $\neq 3$ and is not incident to any cut edge.*[9] *Then, there is a pair of edges (u, v) and (v, w) such that their splitting-off is admissible.*

Since uniformly scaling edge weights does not affect the conditions of Theorem 3.1, we may assume that G is Eulerian and 2-edge-connected. Moreover these conditions are maintained in our algorithm. Therefore the inner while loop of Algorithm 1 is feasible.

To prove Theorem 3.1, we fix a k-projection F with $q(F) \leq \alpha k$. It is sufficient to show that, with good probability, the algorithm maintains the following invariants.

- **(I1):** F is a k-projection in the remaining graph,
- **(I2):** $q(F) \leq \alpha k$ (where $q(F)$ now minimizes over cuts in the remaining graph), and
- **(I3):** every remaining k-heavy edge e has connectivity at least k.

The only modifications to the graph made by Algorithm 1 are admissible splitting-offs, contraction of edges, and removal of self-loops. Clearly removing self-loops does not affect the invariants. Now consider the splitting-off operation. **(I1)** is preserved because we only split-off k-light edges; **(I2)** is preserved because splitting-off never increases the size of any cut; **(I3)** is preserved because we only split-off at a light vertex and the splitting-offs are admissible.

Lemma 3.3. *Let the number of remaining vertices be r. Assuming that the invariants hold, they will continue to hold after the contraction operation with probability at least $1 - 2\alpha/r$.*

Proof. For **(I3)**, note that since contraction does not create new cuts, the edge connectivity of an uncontracted edge cannot decrease. Now consider the graph before the contraction. Since every remaining vertex v is k-heavy, the degree of each vertex is at least k; thus the number of remaining edges is at least $kr/2$. Let C be a cut such that F is the k-projection of C and $w_C = q(F)$. Note that **(I1)** and **(I2)** are preserved if the contracted edge $e \notin C$. Since e is picked uniformly at random, the probability that $e \in C$ is $\mathbb{P}[e \in C] \leq q(F)/(kr/2) = 2q(F)/kr \leq 2\alpha/r$. ■

Let the number of remaining vertices after the splitting-off operations of iteration i in Algorithm 1 be r_i. Then, the probability that all the invariants hold throughout Algorithm 1, and F is the output is at least

$$\left(1 - \frac{2\alpha}{r_0}\right)\left(1 - \frac{2\alpha}{r_1}\right)\cdots\left(1 - \frac{2\alpha}{\lceil 2\alpha \rceil + 1}\right)2^{-(\lceil 2\alpha \rceil - 1)} \geq n^{-2\alpha}.$$

4. THE GENERAL FRAMEWORK

We will now use Theorem 1.6 to prove Theorem 2.3. We re-use the notation defined in section 2.1, and introduce some additional notation. For any cut C, let $F_i^{(C)} = F_i \cap C$ and $E_i^{(C)} = E_i \cap C$ for $0 \leq i \leq k$;[10] let $f_i^{(C)} = |F_i^{(C)}|$ and $e_i^{(C)} = |E_i^{(C)}|$. Also, let $\widehat{f_i^{(C)}}$ be the total weight of all edges in $F_i^{(C)}$ in the sampled graph G_ϵ. Note that the expected value $\mathbb{E}[\widehat{f_i^{(C)}}] = f_i^{(C)}$. We first prove a key lemma.

Lemma 4.1. *For any fixed i, with probability at least $1 - 4/n^2$, every cut C in G satisfies*

$$\left| f_i^{(C)} - \widehat{f_i^{(C)}} \right| \leq \frac{\epsilon}{2} \max\left\{ \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}, f_i^{(C)} \right\}$$

[8]In the next two sections, an edge of weight w is replaced by w unweighted parallel edges.

[9]A *cut edge* is an edge whose removal separates a connected graph into two disconnected components.

[10]For any cut C and any set of edges Z, $Z \cap C$ denotes the set of edges in Z that cross cut C.

Proof. By the π-connectivity property described in section 2.1, any edge $e \in F_i$ is π_i-heavy in G_i for any $i \geq 0$. Therefore, $e_i^{(C)} \geq \pi_i$. Let $\mathcal{C}_{ij}$ be the set of all cuts C such that $\pi_i \cdot 2^j \leq e_i^{(C)} \leq \pi_i \cdot 2^{j+1} - 1$, $j \geq 0$. We will prove that with probability at least $1 - 2n^{-2^{j+1}}$, all cuts in $\mathcal{C}_{ij}$ satisfy the property of the lemma. The lemma follows by the union bound over j (keeping i fixed) since $2n^{-2} + 2n^{-4} + \ldots + 2n^{-2j} + \ldots \leq 4n^{-2}$.

We now prove the above claim for cuts $C \in \mathcal{C}_{ij}$. Let $X_i^{(C)}$ denote the set of edges in $F_i^{(C)}$ that are sampled with probability strictly less than one; correspondingly, let $x_i^{(C)} = |X_i^{(C)}|$ and let $\widehat{x_i^{(C)}}$ be the total weight of edges in $X_i^{(C)}$ in the sampled graph G_ϵ. Since edges in $F_i^{(C)} \setminus X_i^{(C)}$ retain their weight exactly in G_ϵ, it is sufficient to show that with probability at least $1 - 2n^{-2^{j+1}}$,

$$|x_i^{(C)} - \widehat{x_i^{(C)}}| \leq \left(\frac{\epsilon}{2}\right) \max\left\{\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}, x_i^{(C)}\right\}$$

for all cuts $C \in \mathcal{C}_{ij}$. Since each edge $e \in X_i^{(C)}$ has $\lambda_e < 2^{i+1}$, we can use Theorem 2.2 with the lower bound on probabilities $p = \frac{96\alpha \ln n}{0.38 \cdot 2^{i+1}\epsilon^2}$. There are two cases. In the first case, suppose $x_i^{(C)} \leq \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}$. Then, for any $X_i^{(C)}$ where $C \in \mathcal{C}_{ij}$, by Theorem 2.2, we have

$$\mathbb{P}\left[\left|x_i^{(C)} - \widehat{x_i^{(C)}}\right| > \left(\frac{\epsilon}{2}\right) \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}\right]$$
$$< 2e^{-0.38\frac{\epsilon^2}{4}\left(\frac{96\alpha \ln n}{0.38 \cdot 2^{i+1}\epsilon^2}\right)\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}} \leq 2e^{-6 \cdot 2^j \ln n},$$

since $e_i^{(C)} \geq \pi_i \cdot 2^j$ for any $C \in \mathcal{C}_{ij}$. In the second case, $x_i^{(C)} > \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}$. Then, for any $X_i^{(C)}$ where $C \in \mathcal{C}_{ij}$, by Theorem 2.2, we have

$$\mathbb{P}\left[\left|x_i^{(C)} - \widehat{x_i^{(C)}}\right| > \left(\frac{\epsilon}{2}\right) x_i^{(C)}\right]$$
$$< 2e^{-0.38\frac{\epsilon^2}{4}\left(\frac{96\alpha \ln n}{0.38 \cdot 2^{i+1}\epsilon^2}\right)x_i^{(C)}} < 2e^{-6 \cdot 2^j \ln n},$$

since $x_i^{(C)} > \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha} \geq \frac{2^{i+j-1}}{\alpha}$ for any $C \in \mathcal{C}_{ij}$. Thus, we have proved that

$$\mathbb{P}\left[\left|x_i^{(C)} - \widehat{x_i^{(C)}}\right| > \left(\frac{\epsilon}{2}\right) \max\left\{\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}, x_i^{(C)}\right\}\right]$$
$$< 2e^{-6 \cdot 2^j \ln n} = 2n^{-6 \cdot 2^j}$$

for any cut $C \in \mathcal{C}_{ij}$. Now, by the π-connectivity property, we know that edges in $F_i^{(C)}$, and therefore those in $X_i^{(C)}$, are π_i-heavy in G_i. Therefore, by Theorem 1.6, the number of distinct $X_i^{(C)}$ sets for cuts $C \in \mathcal{C}_{ij}$ is at most $n^{2\left(\frac{\pi_i \cdot 2^{j+1}}{\pi_i}\right)} = n^{4 \cdot 2^j}$. Using the union bound over these distinct $X_i^{(C)}$ edge sets, we conclude that with probability at least $1 - 2n^{-2^{j+1}}$, all cuts in $\mathcal{C}_{ij}$ satisfy the property of the lemma. ■

We now use the above lemma to prove Theorem 2.3.
Proof (of Theorem 2.3). Lemma 4.1 bounds the sampling error for a fixed i. In this theorem we bound the total error by summing from $i = 0, \ldots, k$. Recall that $k \leq n - 1$.

Let w_C and $\widehat{w_C}$ be the weight of edges crossing a cut C in G and G_ϵ respectively. By a union bound, the conclusion of Lemma 4.1 holds for every value of i with probability at least $1 - 4/n$. Therefore

$$\sum_{i=0}^{k} |\widehat{f_i^{(C)}} - f_i^{(C)}| \leq \sum_{i=0}^{k} \left(\frac{\epsilon}{2}\right) \max\left\{\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha}, f_i^{(C)}\right\}$$

for all cuts C. Then, with probability at least $1 - 4/n$,

$$|\widehat{w_C} - w_C| = \left|\sum_{i=0}^{k} \widehat{f_i^{(C)}} - \sum_{i=0}^{k} f_i^{(C)}\right| \leq \sum_{i=0}^{k} |\widehat{f_i^{(C)}} - f_i^{(C)}|$$
$$\leq \frac{\epsilon}{2}\left(\sum_{i=0}^{k} \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha} + \sum_{i=0}^{k} f_i^{(C)}\right) \leq \epsilon w_C,$$

since $\sum_{i=0}^{k} \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \alpha} \leq w_C$ by the α-overlap property and $\sum_{i=0}^{k} f_i^{(C)} \leq w_C$ since $F_i^{(C)}$'s form a partition of the edges in C.

We now prove the bound on the expected number of edges in G_ϵ. The expected number of distinct edges in G_ϵ is

$$\sum_{e \in E} (1 - (1 - p_e)^{w_e}) \leq \sum_{e} w_e p_e.$$

The bound follows by substituting the value of p_e. ■

4.1 Sampling using Edge Connectivities

We now use the general framework to prove Theorem 1.1. For any edge $e = (u, v)$, set λ_e to the connectivity k_e of e; also set $\alpha = 3 + \lg n$ and $\pi_i = 2^{i-1}$. F_i is defined as the set of all edges e with $2^i \leq \lambda_e \leq 2^{i+1} - 1$ for any $i \geq 0$. For any $i \geq 1 + \lg n$, let G_i contain all edges in NI forests $T_{2^{i-1-\lg n}}, T_{2^{i-1-\log n}+1}, \ldots, T_{2^{i+1}-1}$ and all edges in F_i. For $i \leq \lg n$, G_i contains all edges in $T_1, T_2, \ldots, T_i$ and all edges in F_i. For any $i \geq 0$, let Y_i denote the set of edges in G_i but not in F_i. For any $i \neq j$, $F_i \cap F_j = \emptyset$ and each edge appears in Y_i for at most $2 + \log n$ different values of i; this proves α-overlap. To prove π-connectivity, we note that for any pair of vertices u, v with connectivity[11] $k(u, v)$ and for any $i \geq 1$, u, v are at least $\min\{k(u, v), i\}$-connected in the first i NI forests, i.e. in $T_1 \cup T_2 \cup \ldots \cup T_i$. Thus, any edge $e \in F_i$ is at least 2^i-heavy in the (union of) the NI forests $T_1, T_2, \ldots, T_{2^{i+1}-1}$. Since there are at most 2^{i-1} edges overall in $T_1, T_2, \ldots, T_{2^{i-1-\lg n}-1}$, any edge $e \in F_i$ is 2^{i-1}-heavy in G_i. This proves π-connectivity. Theorem 1.1 now follows directly from Theorem 2.3 and Lemma 2.1.

5. LINEAR-TIME ALGORITHM

The algorithm has three phases. The first phase has the following steps:

- If $m \leq 2\rho n$, where $\rho = \frac{1014 \ln n}{0.38\epsilon^2}$, then $G_\epsilon = G$.
- Otherwise, we construct a set of NI forests of G and all edges in the first 2ρ NI forests are included in G_ϵ with weight one. We call these edges F_0. The edge set Y_0 is then defined as $E \setminus F_0$.

[11]The connectivity of a pair of vertices is the minimum weight of a cut separating them.

The second phase is iterative. The input to iteration i is a graph (V, Y_{i-1}), which is a subgraph of the input graph to iteration $i-1$ (i.e. $Y_{i-1} \subseteq Y_{i-2}$). Iteration i comprises the following steps:

- If the number of edges in Y_{i-1} is at most $2\rho n$, we take all those edges in G_ϵ with weight 2^{i-1} each, and terminate the algorithm.
- Otherwise, all edges in Y_i are sampled with probability 1/2; call the sample X_i and let $G_i = (V, X_i)$.
- We identify a set of edges in X_i (call this set F_i) that has the following properties:
  - The number of edges in F_i is at most $2k_i|V_c|$, where $k_i = \rho \cdot 2^{i+1}$, and V_c is the set of components in (V, Y_i), where $Y_i = X_i \setminus F_i$.
  - Each edge in Y_i is k_i-heavy in G_i.
- We give a sampling probability $p_i = \min\left\{\frac{3}{169 \cdot 2^{2i-9}}, 1\right\}$ to all edges in F_i.

The final phase consists of replacing each edge in F_i (for each i) with 2^i parallel edges, and then sampling each parallel edge independently with probability p_i. If an edge is selected in the sample, it is added to G_ϵ with weight $1/p_i$.

We now give a short description of the sub-routine that constructs the set F_i in the second phase of the algorithm. This sub-routine is iterative itself: we start with $V_c = V$ and $E_c = X_i$, and let $G_c = (V_c, E_c)$. We repeatedly construct $k_i + 1$ NI forests for G_c where $k_i = \rho \cdot 2^{i+1}$ and contract all edges in the $(k_i + 1)$st forest to obtain a new G_c, until $|E_c| \leq 2k_i|V_c|$. The set of edges E_c that finally achieves this property forms F_i.

The complete algorithm is given in Algorithm 2.

Algorithm 2 The linear-time sparsification algorithm.

```
procedure Sparsify($G$)
input: An undirected unweighted graph $G = (V, E)$, a parameter $\epsilon \in (0, 1)$
output: An undirected weighted graph $G_\epsilon = (V, E_\epsilon)$
Set $\rho = \frac{1014 \ln n}{0.38\epsilon^2}$.
If $m \leq 2\rho n$, then $G_\epsilon = G$ and terminate; else, continue.
Construct NI forests $T_1, T_2, \ldots$ for $G$.
Set $i = 0$; $X_0 = E$; $F_0 = \cup_{1 \leq j \leq 2\rho} T_j$; $Y_0 = X_0 \setminus F_0$.
Add each edge in $F_0$ to $G_\epsilon$ with weight 1.
OuterLoop: If $|Y_i| \leq 2\rho n$, then add each edge in $Y_i$ to $G_\epsilon$ with weight $2^{i-1}$ and terminate; else, continue.
Sample each edge in $Y_i$ with probability 1/2 to construct $X_{i+1}$.
Increment $i$ by 1; set $E_c = X_i$; $V_c = V$; $k_i = \rho \cdot 2^{i+1}$.
InnerLoop: If $|E_c| \leq 2k_i|V_c|$, then
    Set $F_i = E_c$; $Y_i = X_i \setminus E_c$.
    For each edge $e \in F_i$, set $\lambda_e = \rho \cdot 4^i$.
    Go to OuterLoop.
Else,
    Construct NI forests $T_1, T_2, \ldots, T_{k_i+1}$ for graph $G_c = (V_c, E_c)$.
    Update $G_c$ by contracting all edges in $T_{k_i+1}$.
    Go to InnerLoop.
For each $i$, for each edge $e \in F_i$,
    Set $p_e = \min\left\{\frac{9216 \ln n}{0.38\lambda_e\epsilon^2}, 1\right\} = \min\left\{\frac{3}{169 \cdot 2^{2i-9}}, 1\right\}$.
    Generate $r_e$ from Binomial($2^i, p_e$).
    If $r_e > 0$, add edge $e$ to $G_\epsilon$ with weight $r_e/p_e$.
```

Cut Preservation. We use the following notation throughout: for any set of unweighted edges Z, cZ denotes these edges with a weight of c given to each edge. Our goal is to prove the following theorem.

Theorem 5.1. *$G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - 8/n$.*

Let K be the maximum value of i for which $F_i \neq \emptyset$; let $S = \left(\cup_{i=0}^{K} 2^i F_i\right) \cup 2^K Y_K$ and $G_S = (V, S)$. Then, we prove the following two theorems, which together yield Theorem 5.1 using the union bound.[12]

Theorem 5.2. *$G_S \in (1 \pm \epsilon/3)G$ with probability at least $1 - 4/n$.*

Theorem 5.3. *$G_\epsilon \in (1 \pm \epsilon/3)G_S$ with probability at least $1 - 4/n$.*

The following property is key to proving both theorems.

Lemma 5.4. *For any $i \geq 0$, any edge $e \in Y_i$ is k_i-heavy in $G_i = (V, X_i)$, where $k_i = \rho \cdot 2^{i+1}$.*

Proof. Since all edges in Y_0 are in NI forests $T_{2\rho+1}, T_{2\rho+2}, \ldots$ of $G_0 = G$, the lemma holds for $i = 0$.

We now prove the lemma for $i \geq 1$. Let $G_e = (V_e, E_e)$ be the component of G_i containing e. We will show that e is k_i-heavy in G_e; since G_e is a subgraph of G_i, the lemma follows. In the execution of the else block of **InnerLoop** on G_e, there are multiple contraction operations, each comprising the contraction of a set of edges. We show that any such contracted edge is k_i-heavy in G_e; it follows that e is k_i-heavy in G_e.

Let G_e have t contraction phases and let the graph produced after contraction phase r be $G_{e,r}$. We now prove that all edges contracted in phase r must be k_i-heavy in G_e by induction on r. For $r = 1$, since e appears in the $(k_i + 1)$st NI forest of phase 1, e is k_i-heavy in G_e. For the inductive step, assume that the property holds for phases $1, 2, \ldots, r$. Any edge that is contracted in phase $r + 1$ appears in the $(k_i + 1)$st NI forest of phase $r + 1$; therefore, e is k_i-connected in $G_{e,r}$. By the inductive hypothesis, all edges of G_e contracted in previous phases are k_i-heavy in G_e; therefore, an edge that is k_i-heavy in $G_{e,r}$ must have been k_i-heavy in G_e. ■

Proof of Theorem 5.2. The next lemma (proof omitted due to space limitations) follows from the general framework.

Lemma 5.5. *With probability at least $1 - 4/n^2$, for every cut C in G_i, $|2x_{i+1}^{(C)} + f_i^{(C)} - x_i^{(C)}| \leq \frac{\epsilon/13}{2^{i/2}} \cdot x_i^{(C)}$, where $x_i^{(C)}, x_{i+1}^{(C)}$ and $f_i^{(C)}$ respectively denote the weight of $X_i \cap C$, $X_{i+1} \cap C$ and $F_i \cap C$.*

[12]Observe that since $\epsilon \leq 1$, $(1+\epsilon/3)^2 \leq 1+\epsilon$ and $(1-\epsilon/3)^2 \geq 1-\epsilon$.

We use the above lemma to prove the following lemma, of which Theorem 5.2 is a corollary for $j = 0$.

Lemma 5.6. *Let* $S_j = \left(\cup_{i=j}^{K} 2^{i-j} F_i\right) \cup 2^{K-j} Y_K$ *for any* $j \geq 0$. *Then,* $S_j \in (1 \pm (\epsilon/3)2^{-j/2}) G_j$ *with probability at least* $1 - 4/n$, *where* $G_j = (V, X_j)$.

Proof. To prove this lemma, we need to use the following fact (proof omitted due to space limitations).

Fact 5.1. *Let* $x \in (0, 1]$ *and* $r_i = 13 \cdot 2^{i/2}$. *Then, for any* $k \geq 0$, $\prod_{i=0}^{k}(1 + x/r_i) \leq 1 + x/3$ *and* $\prod_{i=0}^{k}(1 - x/r_i) \geq 1 - x/3$.

For any cut C in G, let the edges crossing C in S_j be $S_j^{(C)}$, and let their total weight be $s_j^{(C)}$. Also, let $X_i^{(C)}$, $Y_i^{(C)}$ and $F_i^{(C)}$ be the set of edges crossing cut C in X_i, Y_i and F_i respectively, and let their weights be $x_i^{(C)}$, $y_i^{(C)}$ and $f_i^{(C)}$.

Since $K \leq n-1$, we can use the union bound on Lemma 5.5 to conclude that with probability at least $1 - 4/n$, for every $0 \leq i \leq K$ and for all cuts C,

$$\begin{aligned} 2x_{i+1}^{(C)} + f_i^{(C)} &\leq (1 + \epsilon/r_i)x_i^{(C)} \\ 2x_{i+1}^{(C)} + f_i^{(C)} &\geq (1 - \epsilon/r_i)x_i^{(C)}, \end{aligned}$$

where $r_i = 13 \cdot 2^{i/2}$. Then,

$$\begin{aligned} s_j^C &= 2^{K-j}y_K^{(C)} + 2^{K-j}f_K^{(C)} + 2^{K-1-j}f_{K-1}^{(C)} + \ldots + f_j^{(C)} \\ &= 2^{K-j}x_K^{(C)} + 2^{K-1-j}f_{K-1}^{(C)} + \ldots + f_j^{(C)} \\ &\quad \text{since } y_K^{(C)} + f_K^{(C)} = x_K^{(C)} \\ &= 2^{K-1-j}(2x_K^{(C)} + f_{K-1}^{(C)}) + (2^{K-2-j}f_{K-2}^{(C)} + \ldots) \\ &\leq (1 + \epsilon/r_{K-1})2^{K-1-j}x_{K-1}^{(C)} + (2^{K-2-j}f_{K-2}^{(C)} + \ldots) \\ &\leq (1 + \epsilon/r_{K-1})(2^{K-1-j}x_{K-1}^{(C)} + 2^{K-2-j}f_{K-2}^{(C)} + \ldots) \\ &\ldots \\ &\leq (1 + \epsilon/r_{K-1})(1 + \epsilon/r_{K-2})\ldots(1 + \epsilon/r_j)x_j^{(C)} \\ &\leq (1 + (\epsilon 2^{-j/2})/r_{K-1-j})(1 + (\epsilon 2^{-j/2})/r_{K-2-j})\ldots \\ &\quad \ldots(1 + (\epsilon 2^{-j/2})/r_0)x_j^{(C)} \quad \text{since } r_{j+i} = r_i \cdot 2^{j/2} \\ &\leq (1 + (\epsilon/3)2^{-j/2})x_j^{(C)} \quad \text{by Fact 5.1.} \end{aligned}$$

Similarly, we can show that $s_j^C \geq (1 - (\epsilon/3)2^{-j/2})x_j^{(C)}$. ■

Proof of Theorem 5.3. First, observe that edges $F_0 \cup 2^K Y_K$ are identical in G_S and G_ϵ. Therefore, we do not consider these edges in the analysis below. For any $i \geq 1$, let $\psi(i)$ be such that $2^{\psi(i)} \leq \rho \cdot 4^i \leq 2^{\psi(i)+1} - 1$. Note that for any j, $\psi(i) = j$ for at most one value of i. Then, for any $j \geq 1$, $R_j = F_i$ if $j = \psi(i)$ and $R_j = \emptyset$ if there is no i such that $j = \psi(i)$. We set $\alpha = 32/3$; $\pi_j = \rho \cdot 4^K$; for any $j \geq 1$, $Q_j = (V, W_j)$ where $W_j = \cup_{i-1 \leq r \leq K} 4^{K-r+1}2^r F_r$ if $R_j \neq \emptyset$ and $j = \psi(i)$, and $W_j = \emptyset$ if $R_j = \emptyset$.

The following lemma ensures π-connectivity (proof omitted due to space limitations).

Lemma 5.7. *With probability at least* $1 - 4/n$, *every edge* $e \in F_i = R_{\psi(i)}$ *for each* $i \geq 1$ *is* $\rho \cdot 4^K$*-heavy in* $Q_{\psi(i)}$.

We now prove the α-overlap property. For any cut C, let $f_i^{(C)}$ and $w_i^{(C)}$ respectively denote the total weight of edges crossing cut C in F_i and $W_{\psi(i)}$ respectively for any $i \geq 0$. Further, let the number of edges crossing cut C in $\cup_{i=0}^{K} 2^i F_i$ be $f^{(C)}$. Then,

$$\begin{aligned} \sum_{i=1}^{K} \frac{w_i^{(C)} 2^{\psi(i)-1}}{\pi} &= \sum_{i=1}^{K} \sum_{r=i-1}^{K} \frac{f_r^{(C)} \cdot 2^r \cdot 4^{K-r+1}}{2 \cdot 4^{K-i}} \\ = \sum_{r=0}^{K} \frac{f_r^{(C)}}{2^r} \sum_{i=1}^{r+1} 2^{2i+1} &= \frac{32}{3} \sum_{r=0}^{K} 2^r f_r^{(C)} = \frac{32}{3} f^{(C)}. \end{aligned}$$

Using Theorem 2.3, we conclude the proof of Theorem 5.3.

Size of G_ϵ. We now prove that the expected number of edges in G_ϵ is $O(n \log n/\epsilon^2)$. For $i \geq 1$, define D_i to be the set of connected components in the graph $G_i = (V, X_i)$; let D_0 be the single connected component in G. For any $i \geq 1$, if any connected component in D_i remains intact in D_{i+1}, then there is no edge from that connected component in F_i. On the other hand, if a component in D_i splits into η components in D_{i+1}, then the algorithm explicitly ensures that $\sum_{e \in F_i} \frac{w_e}{\lambda_e}$[13] from that connected component is $\sum_{e \in F_i} \frac{2^i}{\rho \cdot 4^i} \leq \left(\frac{\rho \cdot 2^{i+2} \cdot 2^i}{\rho \cdot 4^i}\right)\eta = 4\eta \leq 8(\eta - 1)$. Therefore, if $d_i = |D_i|$, then

$$\sum_{i=1}^{K} \sum_{e \in F_i} \frac{w_e}{\lambda_e} \leq \sum_{i=1}^{K} 8(d_{i+1} - d_i) \leq 8n,$$

since we can have at most n singleton components. It follows from Theorem 2.3 that the expected number of edges added to G_ϵ by the sampling is $O(n \log n/\epsilon^2)$.

Time complexity. If $m \leq 2\rho n$, the algorithm terminates after the first step which takes $O(m)$ time. Otherwise, we prove that the expected running time of the algorithm is $O(m + n \log n/\epsilon^2) = O(m)$ since $\rho = \Theta(\log n/\epsilon^2)$. First, observe that phase 1 takes $O(m + n \log n)$ time. In iteration i of phase 2, the first step takes $|Y_{i-1}|$ time. Using arguments similar to [4], we can show that all the remaining steps take $O(|X_i| + n \log n)$ time. Since $X_i \subseteq Y_{i-1}$ and the steps are executed only if $Y_{i-1} = \Omega(n \log n/\epsilon^2)$, it follows that the total time complexity of iteration i of phase 2 is $O(|Y_{i-1}|)$. Since $Y_i \subset X_i$ and $\mathbb{E}[|X_i|] = \mathbb{E}[|X_{i-1}|]/2$, and $|Y_0| \leq m$, it follows that the expected overall time complexity of phase 2 is $O(m)$. Finally, the time complexity of phase 3 is $O(m + n \log n/\epsilon^2)$ (see e.g. [8]).

6. LOWER BOUNDS

We have already noted that independent sampling of edges cannot produce sparsifiers containing $o(n \log n)$ edges. A possible alternative is to sample spanning trees uniformly at random, and Theorem 1.2 asserts that this sampling technique indeed produces cut sparsifiers. We now give a lower bound for the tradeoff between the number of trees (i.e., the value ρ) and the quality of sparsification in Theorem 1.2.

Lemma 6.1. *For any constant* $c \geq 1$, *there is a graph such that* $\rho = \Omega(\log n)$ *spanning trees have to be sampled uniformly at random to approximate all cuts within a factor* c *with constant probability.*

[13] w_e is the number of parallel copies of e in the Binomial sampling step.

Proof. Let G be a graph defined as follows. Its vertices are $\{u_1, \ldots, u_n\} \cup \{v_1, \ldots, v_{n+1}\}$. For every $i = 1, \ldots, n$, add k parallel edges $v_i v_{i+1}^{(1)}, \ldots, v_i v_{i+1}^{(k)}$, and a single length-two path v_i-u_i-v_{i+1}. The edges $v_i v_{i+1}^{(j)}$ are called *heavy*, and the edges $v_i u_i$ and $u_i v_{i+1}$ are called *light*. Note that the heavy edges each have effective conductance exactly $(2k+1)/2$. The light edges each have effective conductance exactly $(2k+1)/(k+1) < 2$.

A uniform random spanning tree in this graph can be constructed by repeating the following experiment independently for each $i = 1, \ldots, n$. With probability $2k/(2k+1)$, add a uniformly selected heavy edge $v_i v_{i+1}^{(j)}$ to the tree, and add a uniformly selected light edge $v_i u_i$ or $u_i v_{i+1}$ to the tree. In this case we say that the tree is "heavy in position i". Otherwise, with probability $1/(2k+1)$, add both light edges $v_i u_i$ and $u_i v_{i+1}$ to the tree but no heavy edges. In this case we say that the tree is "light in position i".

Our sampling procedure produces a sparsifier that is the union of ρ trees, where every edge e in the sparsifier is assigned weight c_e/ρ. Suppose there is an i such that all sampled trees are light in position i. Then the cut defined by vertices $\{v_1, u_1, v_2, u_2, \ldots, v_i\}$ has weight exactly $(2k+1)/(k+1) < 2$ in the sparsifier, whereas the true value of the cut is $k+1$.

The probability that at least one tree is heavy in position i is $1-(2k+1)^{-\rho}$. The probability that there exists an i such that every tree is light in position i is $p = 1-(1-(2k+1)^{-\rho})^n$. Suppose $\rho = \ln n / \ln(2k+1)$. Then $\lim_{n\to\infty} p = 1-1/e$. So with constant probability, there is an i such that every tree is light in position i, and so the sparsifier does not approximate the original graph better than a factor $\frac{k+1}{2}$. ■

Acknowledgments. Partial support for conducting this research was provided by an NSERC Discovery Grant for W. S. Fung and N. J. A. Harvey, and by NSF-STC Award 0939370 for D. Panigrahi. D. Panigrahi would like to thank David Karger for helpful discussions.

7. REFERENCES

[1] A. Asadpour, M. X. Goemans, A. Madry, S. O. Gharan, and A. Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 379–389, 2010.

[2] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-Ramanujan sparsifiers. In *Proc. 41st Annual ACM Symposium on Theory of Computing*, pages 255–262, 2009.

[3] A. A. Benczúr and D. R. Karger. Approximate s-t min-cuts in $\tilde{O}(n^2)$ time. In *Proc. 28th Annual ACM Symposium on Theory of Computing*, 1996.

[4] A. A. Benczúr and D. R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs, 2002. `http://arxiv.org/abs/cs/0207078`.

[5] A. Goel, M. Kapralov, and S. Khanna. Perfect matchings in $\tilde{O}(n^{1.5})$ time in regular bipartite graphs, 2009. `http://arxiv.org/abs/0902.1617`.

[6] M. X. Goemans, N. J. A. Harvey, K. Jain, and M. Singh. A randomized rounding algorithm for the asymmetric traveling salesman problem, 2009. `http://arxiv.org/abs/0909.0941`.

[7] N. Goyal, L. Rademacher, and S. Vempala. Expanders via random spanning trees. In *Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 576–585, 2009.

[8] V. Kachitvichyanukul and B. W. Schmeiser. Binomial random variate generation. *Commun. ACM*, 31(2):216–222, 1988.

[9] D. R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 21–30, 1993.

[10] D. R. Karger. Random sampling in cut, flow, and network design problems. In *Proc. 26th Annual ACM Symposium on Theory of Computing*, 1994.

[11] D. R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, May 1999.

[12] D. R. Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM J. Comput.*, 29(2):492–514, 1999.

[13] D. R. Karger and M. S. Levine. Random sampling in residual graphs. In *Proc. 34th Annual ACM Symposium on Theory of Computing*, pages 63–66, 2002.

[14] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, July 1996.

[15] R. Khandekar, S. Rao, and U. V. Vazirani. Graph partitioning using single commodity flows. *Journal of the ACM*, 56(4), 2009.

[16] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving SDD systems. In *Proc. 51th Annual IEEE Symposium on Foundations of Computer Science*, pages 235–244, 2010.

[17] W. Mader. A reduction method for edge-connectivity in graphs. *Ann. Discrete Math.*, 3:145–164, 1978.

[18] A. Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Proc. 51th Annual IEEE Symposium on Foundations of Computer Science*, pages 245–254, 2010.

[19] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1997.

[20] H. Nagamochi and T. Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM J. Discrete Math.*, 5(1):54–66, 1992.

[21] H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5&6):583–596, 1992.

[22] D. Peleg and A. A. Schäffer. Graph spanners. *J. Graph Th.*, 13(1):99–116, 1989.

[23] S. Rao and S. Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

[24] J. Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut. In *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 363–372, 2009.

[25] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proc. 40th Annual ACM Symposium on Theory of Computing*, pages 563–568, 2008.

[26] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. 36th Annual ACM Symposium on Theory of Computing*, pages 81–90, 2004. `http://arxiv.org/abs/cs.DS/0310051`.

[27] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs, 2008. `http://arxiv.org/abs/0808.4134`.