SIAM J. COMPUT.
Vol. 48, No. 4, pp. 1196–1223

A GENERAL FRAMEWORK FOR GRAPH SPARSIFICATION*

WAI-SHING FUNG†, RAMESH HARIHARAN‡, NICHOLAS J. A. HARVEY§, AND DEBMALYA PANIGRAHI¶

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 [*Approximating s-t minimum cuts in $\tilde{o}(n^2)$ time*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, ACM, New York, 1996, pp. 47–55], [*SIAM J. Comput.*, 44 (2015), pp. 290–319]), *effective resistance* (a result of Spielman and Srivastava [*SIAM J. Comput.*, 40 (2011), pp. 1913–1926]), or *edge connectivity*. 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 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 (due to Benczúr and Karger [*Approximating s-t minimum cuts in $\tilde{o}(n^2)$ time*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, ACM, New York, 1996, pp. 47–55], [*SIAM J. Comput.*, 44 (2015), pp. 290–319]) of $O(m \log^2 n)$ (for the unweighted case) and $O(m \log^3 n)$ (for the weighted case), respectively. Our algorithm constructs sparsifiers that contain $O(n \log n/\epsilon^2)$ edges in expectation. 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.

Key words. cut sparsification, edge connectivity, edge sampling

AMS subject classifications. 68W05, 68W20, 68W25

DOI. 10.1137/16M1091666

1. Introduction.

Sparsification is an important technique in the design of fast graph algorithms. The goal of sparsification is to represent a dense graph using a sparse graph so that important structural properties are approximately preserved. Remarkably, this is possible, for various structural properties. For example, given any undirected graph, there are sparse subgraphs that approximate *all* pairwise distances up to a multiplicative and/or additive error (see [33] and subsequent research on *spanners* and *emulators*), *every* cut to an arbitrarily small multiplicative error (called *cut sparsifiers*, introduced by Benczúr and Karger [7, 8]), or the entire Laplacian

*Received by the editors August 30, 2016; accepted for publication (in revised form) April 26, 2019; published electronically July 16, 2019. A preliminary version of these results appeared in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, 2011 [12].
https://doi.org/10.1137/16M1091666
Funding: The work of the fourth author was supported by NSF STC award 0939370.
†Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada N2L 3G1 (wsfung@uwaterloo.ca).
‡Strand Life Sciences, Bangalore 560024, India, and the Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India (ramesh@strandls.com).
§Department of Computer Science, University of British Columbia, Vancouver, BC, Canada V6T 1Z4 (nickhar@cs.ubc.ca). Part of this work was done while the author was at the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada N2L 3G1.
¶Department of Computer Science, Duke University, Durham, NC 27708 (debmalya@cs.duke.edu). Part of this work was done while the author was a graduate student in the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.

quadratic form up to an arbitrarily small multiplicative error (called *spectral sparsifiers*, introduced by Spielman and Teng [36, 37]), and so on. Such approximations are a cornerstone of numerous important results in theoretical computer science.

In this paper, we consider the problem of *cut sparsification*, i.e., approximating every cut arbitrarily well. This problem was originally studied by Karger [20] and Benczúr and Karger [7, 8], motivated by the design of fast connectivity algorithms. They proved that every undirected graph has cut sparsifiers and gave an efficient algorithm for their construction. They also put forward a conjecture about cut sparsification using edge connectivities and predicted that their cut sparsification results can be significantly simplified if the conjecture were true. We resolve this conjecture in the affirmative as a simple corollary of a general cut sparsification framework that we propose, and show that our framework can be used to simplify, unify, and improve previous cut sparsification results. Our structural insights also lead to improved cut sparsification algorithms, both for weighted and unweighted graphs. In particular, we give a strictly linear-time algorithm for cut sparsification in unweighted graphs. Since the introduction of cut sparsifiers, a stronger notion of sparsification—spectral sparsification—has emerged. Our results are restricted to cut sparsifiers only and the sparsifiers that we construct do not satisfy the stronger requirements of spectral sparsification.

The cut sparsification problem. In this problem, the input comprises a weighted, undirected graph $G = (V, E)$ and an error parameter ϵ. Throughout this paper, we will assume that the edge weights are integral, allowing a multigraph representation. This transformation does not impact our structural results, but it allows us to use unweighted graphs, thereby simplifying notation. The efficiency of our sparsification algorithms, however, is sensitive to the presence of edge weights, and we distinguish between unweighted and weighted graphs in algorithmic results.

The goal of the cut sparsification problem is to output a weighted graph $G_\epsilon = (V, F)$ such that the value of every cut in G_ϵ is within a multiplicative factor $(1 \pm \epsilon)$ of the value of the corresponding cut in G. (Recall that the *value* of a cut is the number of edges in the cut for unweighted graphs, and the sum of edge weights for weighted graphs.) The graph G_ϵ is then called a *cut sparsifier* of G, which will often be abbreviated as $G_\epsilon \in (1 \pm \epsilon)G$. Cut sparsification is frequently used as a preprocessing step in connectivity algorithms so that the algorithms run on graphs containing fewer edges and are therefore faster.

1.1. Connectivity parameters. In this paper, we will use several connectivity parameters for undirected graphs, of which edge connectivity is perhaps the most natural.

DEFINITION 1.1. *For any pair of vertices u and v, the* edge connectivity *between u and v, denoted k_{uv}, is defined as the minimum value of a cut that separates u and v. The* connectivity *of edge $e = (u, v)$, denoted k_e, is defined as k_{uv}.*

Benczúr and Karger introduced a new connectivity parameter called *edge strength* and used it in their sparsification scheme.

DEFINITION 1.2. *A* k-strong component *of G is a maximal k-edge-connected, vertex-induced subgraph of G. The* strength *of edge $e = (u, v)$, denoted s_e or s_{uv}, is the maximum value of k such that a k-strong component of G contains both u and v.*

As a third connectivity parameter, we define electrical resistances and conduc-

tances of edges, which turns out to be useful for spectral sparsification.

DEFINITION 1.3. *The* effective conductance *of edge $e = (u, v)$, denoted c_e or c_{uv}, is the amount of current that flows when each edge e of weight w_e is viewed as a resistor of resistance $1/w_e$ and a unit voltage difference is applied between u and v. The* effective resistance *of an edge e is the reciprocal of its effective conductance.*

Nagamochi and Ibaraki [32, 31] introduced a simple graph partitioning scheme for estimating connectivities that leads to a new connectivity parameter called Nagamochi–Ibaraki (NI) indices.

DEFINITION 1.4. *A sequence of edge-disjoint spanning forests*[1] *$T_1, T_2, \ldots$ of a graph G is said to be an* 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, if weighted) NI forest in which e appears.*

The parameters s_e, c_e, and ℓ_e are mutually incomparable; however, the next lemma shows that edge connectivity k_e dominates all of these parameters.

LEMMA 1.5. *Suppose edge e in an undirected graph G has edge connectivity k_e, effective conductance c_e, edge strength s_e, and NI index ℓ_e. Then,*

$$k_e \geq \max(c_e, s_e, \ell_e).$$

Proof. $k_e \geq s_e$ follows from the fact that the strength of an edge is equal to its connectivity in a subgraph.

Consider a cut C of weight k_e separating the terminals of edge e. We contract each side of this cut into a single vertex. In other words, we increase the conductance of each edge, other than those in C, to ∞. By Rayleigh's monotonicity principle (see, e.g., [11]), the effective conductance of e does not decrease due to this transformation. Since the effective conductance of e after the transformation is k_e, $c_e \leq k_e$ in the original graph.

Note that there are ℓ_e edge-disjoint paths connecting the end-points of edge e in the first ℓ_e NI forests. It follows, by Menger's theorem (see, e.g., [10]), that $k_e \geq \ell_e$. ∎

Further, there are known bounds on the sum of reciprocals of these connectivity parameters. The bound on edge strengths is given in [7].

LEMMA 1.6 (see Benczúr and Karger [7]). *Suppose G is an undirected graph where edge e has weight w_e and strength s_e. Then,*

$$\sum_e \frac{w_e}{s_e} \leq n - 1.$$

The bound on edge connectivities now follows from Lemma 1.5.

COROLLARY 1.7. *Suppose G is an undirected graph where edge e has weight w_e and connectivity k_e. Then,*

$$\sum_e \frac{w_e}{k_e} \leq n - 1.$$

We now show similar bounds for conductances and NI indices. The bound for conductance is well known.

[1] A *spanning forest* of a (not necessarily connected) graph is a collection of spanning trees, one on each connected component of the graph.

LEMMA 1.8 (see Bollobás [9, Exercise IX.23]). *Suppose G is an undirected graph where edge e has weight w_e and conductance c_e. Then,*

$$\sum_e \frac{w_e}{c_e} = n - 1.$$

On the other hand, the bound for NI indices is slighter weaker and follows from a counting argument.

LEMMA 1.9. *Suppose G is an undirected graph and let $T_1, T_2, \ldots$ be an NI forest packing where edge e has weight w_e and NI index ℓ_e. If $W = \max_e w_e$, then*

$$\sum_e \frac{w_e}{\ell_e} = O(n \log(nW)).$$

If particular, if all edge weights are polynomial in n, then

$$\sum_e \frac{w_e}{\ell_e} = O(n \log n).$$

Proof. Since the sum of edge weights $\sum_e w_e \le n^2 W$, the number of NI forests in any NI forest packing is also at most $n^2 W$. Now, since ℓ_e is the *last* index of a forest that contains a copy of e, we can upper bound $\sum_e \frac{w_e}{\ell_e}$ by treating edge e as a set of w_e distinct parallel edges, each having an NI index equal to the NI forest it belongs to. Then, NI forest T_i contributes at most $(n-1)/i$ to the sum, and the overall bound follows by summing over all i. ∎

1.2. Edge compression. A key idea in cut sparsification is that of edge compression.

DEFINITION 1.10. *If the input graph is unweighted, then an edge e is said to be* compressed *with probability p_e if the edge is sampled with probability p_e and if selected, it is given a weight of $1/p_e$ in the output. If the input graph is weighted and edge e has weight w_e, then it is said to be* compressed *with probability p_e if its weight in the output graph is $1/p_e$ times a binomial random variable with parameters w_e and p_e. Note that for integral w_e, this is equivalent to replacing the edge of weight w_e with w_e parallel unweighted edges and applying the unweighted compression procedure to each such edge independently.*

Note that the expected weight of an edge after compression is equal to its weight before compression. However, the variance of the edge weight after compression depends on the probability p_e.

1.3. Related work. For structural results in sparsification, we will describe the results for unweighted (multi-)graphs. This is without loss of generality (w.l.o.g.) for integer edge weights since an edge of weight w is equivalent to w parallel edges. However, as mentioned earlier, such a transformation affects algorithmic performance; hence, for algorithmic results, we will distinguish between weighted and unweighted graphs. It is important to note that in either case, the output graph, i.e., the sparsifier, is weighted due to edge compression.

The first result in cut sparsification was obtained by Karger [20] who proposed a uniform compression of edges.

THEOREM 1.11 (see Karger [20]). *Let G_ϵ be obtained from an unweighted graph G by independently compressing edge e with probability $p = \min(\rho/\lambda, 1)$, where*

$$\rho = 3(d+2) \ln n / \epsilon^2$$

and λ is the value of a minimum cut in G. Then, G_ϵ contains $O\left(\frac{m \log n}{\lambda \epsilon^2}\right)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$.

Karger also gave an $O(m)$-time implementation of this compression scheme for weighted graphs. This theorem depends on the value of the minimum cut in G. Benczúr and Karger [7] removed this dependence by using nonuniform compression of edges to show that that for every graph G, there exists a cut sparsifier containing only $O(n \log n/\epsilon^2)$ edges.

THEOREM 1.12 (see Benczúr and Karger [7]). *Let G_ϵ be obtained from an unweighted graph G by independently compressing edge e with probability $p_e = \min(\rho/s_e, 1)$, where*

$$\rho = 16(d+2) \ln n/\epsilon^2.$$

(Recall that s_e is the strength of edge e.) Then, G_ϵ contains $O(n \log n/\epsilon^2)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$.

Benczúr and Karger also gave an efficient randomized algorithm to construct a cut sparsifier containing $O(n \log n/\epsilon^2)$ edges in expectation. This algorithm runs in $O(m \log^2 n)$ time if G is unweighted and $O(m \log^3 n)$ time if G is weighted. They also conjectured that replacing edge strengths by edge connectivity in their compression scheme will also yield cut sparsifiers, and will lead to significant simplification in both sparsification algorithms and their analysis.

As noted earlier, Spielman and Teng [37] introduced spectral sparsification as a generalization of cut sparsification and proved that every graph has spectral sparsifiers with $O(n \log^c n/\epsilon^2)$ edges for a large constant c. This was improved by Spielman and Srivastava [35] who obtained spectral sparsifiers containing $O(n \log n/\epsilon^2)$ edges.

THEOREM 1.13 (see Spielman and Srivastava [35]). *Let G_ϵ be obtained from an unweighted graph G by independently compressing edge e with probability $p_e = \min(\rho/c_e, 1)$, where*

$$\rho = 6 \ln n/\epsilon^2.$$

(Recall that c_e is the conductance of edge e.) Then, G_ϵ contains $O(n \log n/\epsilon^2)$ edges in expectation, and G_ϵ is a spectral sparsifier of G with constant probability.

Remark 1. Actually, the result proven by Spielman and Srivastava [35] is slightly different: they construct the sparsifier by drawing exactly $\rho(n-1)$ independently and identically distributed (i.i.d.) samples, in which each edge e is sampled with probability $\frac{1}{c_e(n-1)}$ and is assigned weight $\frac{c_e}{\rho}$. One may modify their analysis to obtain Theorem 1.13 by using the "matrix Chernoff bound," e.g., Tropp [38]. On the other hand, the analysis of our paper can easily be modified to use their i.i.d. sampling process instead of our edge compression process. To do so, one simply modifies the proof of Lemma 3.1 to use a different form of the Chernoff bound.

Spielman and Srivastava [35] also gave an efficient algorithm to construct a spectral sparsifier with $O(n \log n/\epsilon^2)$ edges in expectation; using later improvements to linear system solvers [25, 26, 24], the best algorithm for producing a spectral sparsifier containing $O(n \log n/\epsilon^2)$ edges now runs in $O(\min(m \log^2 n, m \log n + n \log^5 n))$ time (ignoring $\log \log n$ factors).

Further improvement in spectral sparsification was achieved by Batson, Spielman, and Srivastava [6], who showed the existence of spectral sparsifiers containing $O(n/\epsilon^2)$ edges for every graph, which is optimal for spectral sparsifiers [6], and even for cut sparsifiers [4]. Batson, Spielman, and Srivastava also gave a deterministic algorithm for constructing such spectral sparsifiers in $O(n^3 m)$ time.

Subsequent work. Subsequent to our work, connections between spectral sparsifiers and graph spanners [18], and variants of spectral sparsification where specific subgraphs need to be retained [23] have been studied. Both cut sparsification [1, 14, 2, 3, 15] and spectral sparsification [22] have also been studied recently in the semi-streaming model. Furthermore, a series of efficient algorithms for spectral sparsification have been proposed [40, 39, 27], the current best being an algorithm with running time $\tilde{O}\big(\frac{qmn^{5/q}}{\epsilon^{4+4/q}}\big)$ for constructing a spectral sparsifier containing $O(qn/\epsilon^2)$ edges.

1.4. Our contributions. We now outline our main contributions.

1.4.1. A general sparsification framework. We propose a general sparsification framework and set out sufficient conditions for a sampling scheme to result in cut sparsifiers. In describing the framework, we will assume that the input graph G is unweighted (allowing for parallel edges). Let G_ϵ be obtained from G by independently compressing edge e with probability

$$p_e = \min\left(\frac{112\gamma \ln n}{0.38\lambda_e \epsilon^2}, 1\right),$$

where γ is independent of e and λ_e is a parameter defined on edge e. We describe below a sufficient condition on the γ and λ_e values for G_ϵ to be a cut sparsifier.

To describe this sufficient condition, we partition the edges in G according to the value of λ_e into sets $F_0, F_1, \ldots, F_\Lambda$, where

$$\Lambda = \lfloor \lg \max_{e \in E}\{\lambda_e\} \rfloor$$

and

$$F_i = \{e : 2^i \le \lambda_e \le 2^{i+1} - 1\}.$$

We will obtain concentration bounds for each F_i separately since edges in any F_i have roughly the same sampling probability in the compression scheme. Ideally, we would like to bound the error due to compression of edges in F_i by a multiplicative factor of the size of F_i. Then, summing over all F_i's would immediately yield a concentration bound on the entire graph since the F_i's are disjoint. However, it might so happen that the number of edges in a particular F_i is small, yet these edges have a low sampling probability. This is inconvenient since we cannot hope to bound the error due to such an F_i by a multiplicative factor of the size of F_i. To overcome this problem, we define a subgraph G_i of G (with edges replicated, if required) for each F_i such that edges in F_i are well connected in G_i and, therefore, the error due to F_i can be bounded by a multiplicative factor of the size of G_i. The goal then becomes one of choosing G_i's such that no edge in G is replicated a large number of times across all the G_i's. This ensures that the sum of the individual error bounds on the F_i's in terms of the G_i's can be expressed as a multiplicative error on the entire graph G.

Formally, let $\mathcal{G} = \{\, G_i = (V, E_i) : 1 \le i \le \Lambda \,\}$ be a set of subgraphs of G such that $E_i \supseteq F_i$ for every i. (As mentioned above, we are allowed to make multiple copies of the same edge in G in defining G_i. This flexibility will be crucial to us in an application of the framework.) For a given set of parameters $\Pi = (\pi_0, \pi_1, \ldots, \pi_\Lambda)$, the following properties will play a crucial role in our definition of $\mathcal{G}$:

- **(Π-connectivity.)** The connectivity of any edge $e \in F_i$ in graph G_i is at least π_i.

- **(γ-overlap.)** For any cut C,

$$\sum_{i=0}^{\Lambda} \frac{e_i^{(C)} 2^{i-1}}{\pi_i} \leq \gamma \cdot e^{(C)},$$

where $e^{(C)}$ and $e_i^{(C)}$ denote the value of cut C in graphs G and G_i, respectively. (Recall that γ is a parameter in the edge compression probabilities.)

Theorem 1.14 describes the properties of such a sampling scheme, and is our central theorem of the paper.

THEOREM 1.14. *Fix the parameters γ and λ_e for each edge e. If there exists $\mathcal{G}$ satisfying Π-connectivity and γ-overlap for some Π, then $G_\epsilon \in (1\pm\epsilon)G$ with probability at least $1 - 4/n$, where G_ϵ is obtained by edge compression using parameters γ and λ_e's. Furthermore, G_ϵ has $O\big(\frac{\gamma \log n}{\epsilon^2} \sum_{e\in E} \frac{1}{\lambda_e}\big)$ edges in expectation.*

1.4.2. Applications of the sparsification framework. Our first application of the sparsification framework is to show that compressing by edge connectivities yields cut sparsifiers, thereby resolving the conjecture of Benczúr and Karger.

THEOREM 1.15. *Let G_ϵ be obtained from an unweighted graph G by independently compressing edge e with probability $p_e = \min(\rho/\kappa_e, 1)$, where*

$$\rho = Cd \ln^2 n/\epsilon^2$$

for a large enough constant C and $\kappa_e \leq k_e$. Then, $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$. Additionally, if $\kappa_e = k_e$, then G_ϵ contains $O(n \log^2 n/\epsilon^2)$ edges in expectation.

The next three corollaries of this theorem follow from Lemmas 1.5, 1.6, 1.8, and 1.9.

COROLLARY 1.16. *Let G_ϵ be obtained from an unweighted graph G by independently compressing edge e with probability $p_e = \min(\rho/s_e, 1)$, where*

$$\rho = Cd \ln^2 n/\epsilon^2$$

for a large enough constant C. Then, G_ϵ contains $O(n \log^2 n/\epsilon^2)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$.

Recall that the corresponding result of Benczúr and Karger [7, Theorem 1.12] is stronger than this result since it produces sparsifiers containing $O(n \log n/\epsilon^2)$ edges in expectation. We show later that we can match the Benczúr–Karger bound (up to constant factors) by using our sparsification framework directly.

COROLLARY 1.17. *Let G_ϵ be obtained from an unweighted graph G by independently compressing edge e with probability $p_e = \min(\rho/c_e, 1)$, where*

$$\rho = Cd \ln^2 n/\epsilon^2$$

for a large enough constant C. Then, G_ϵ contains $O(n \log^2 n/\epsilon^2)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$.

This is weaker than the result of Spielman and Srivastava stated earlier in Theorem 1.13: they prove spectral sparsification, not just cut sparsification, and their sparsifier only has $O(n \log n/\epsilon^2)$ edges.

COROLLARY 1.18. *Let G_ϵ be obtained from an unweighted graph G by independently compressing edge e with probability $p_e = \min(\rho/\ell_e, 1)$, where*

$$\rho = Cd \ln^2 n/\epsilon^2$$

for a large enough constant C. Then, G_ϵ contains $O(n \log^2 n \log(nW)/\epsilon^2)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$, where W is the maximum weight of an edge in G.

Note that the additional factor of $\log(nW)$ is due to this term in Lemma 1.9.

As in the case of edge strengths, we will show later that this result can be improved by applying the sparsification framework directly to obtain the following theorem. We state this theorem for weighted graphs since we will use this theorem for algorithmic applications.

THEOREM 1.19. *Let G_ϵ be obtained from a weighted graph G by independently compressing edge e with probability $p_e = \min(\rho/\ell_e, 1)$, where*

$$\rho = Cd \ln n/\epsilon^2$$

for a large enough constant C. Then, G_ϵ contains $O(n \log n \log(nW)/\epsilon^2)$ edges in expectation, and $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$, where W is the maximum weight of an edge in G.

Note that for both Corollary 1.18 and Theorem 1.19, the $\log(nW)$ factor is replaced by $\log n$ for input graphs with polynomial edge weights.[2]

1.4.3. Sparsification algorithms. Our framework yields sparsification algorithms that are not only simpler, but also faster. For simplicity, we will state our running times for these algorithms assuming that the edge weights in the input graph are polynomial in n. For larger weights, the running time of the algorithm and size of the sparsifier typically have a $\log(nW)$ factor instead of a $\log n$ factor. We do not state these more general results for the sake of brevity.

Nagamochi and Ibaraki showed that an NI forest packing can be constructed in $O(m)$-time for unweighted graphs [32], and $O(m + n \log n)$-time for weighted graphs [31]. For weighted graphs, note 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 (see, e.g., [17]); therefore, $O(\sum_{e \in E} w_e p_e)$ time overall for all edges. Recalling that $p_e = \min(\rho/\ell_e, 1)$, it follows from Lemma 1.9 that the time complexity of sampling all edges is $O(n \log^2(n)/\epsilon^2)$, which is $O(m)$. (If $m = O(n \log^2(n)/\epsilon^2)$ then we can retain all edges; therefore, we assume w.l.o.g. that $m = \Omega(n \log^2(n)/\epsilon^2)$.) Coupled with Theorem 1.19, we get the following theorem.

THEOREM 1.20. *For any input graph G (with edge weights polynomial in n) and any constants $\epsilon \in (0, 1), d > 0$, there is a randomized algorithm that runs in $O(m)$-time and produces a graph G_ϵ containing $O(n \log^2 n/\epsilon^2)$ edges in expectation, where $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$.*

The cut sparsifier produced by this algorithm contains $O(n \log^2 n/\epsilon^2)$ edges in expectation, which is a factor of $\log n$ greater than that produced by previous algorithms

[2] We remark that the $\log(nW)$ factor can be replaced by $\log(n)$, even with arbitrarily large edge weights, via a slightly more involved argument that appears in our technical report [13, Corollary 1.2]. However, this alternative argument is intended for the scenario that $p_e = \min(\rho/k_e, 1)$ and does not use the general framework of this paper, so it leads to slightly worse algorithmic results and we do not discuss it herein.

of Benczúr and Karger. However, the output of this algorithm can be postprocessed using the previous algorithm for weighted graphs to obtain a cut sparsifier containing $O(n \log n/\epsilon^2)$ edges. Recall that the best previously known algorithm runs in $O(m \log^3 n)$ time for weighted graphs.

COROLLARY 1.21. *For any input graph G (with edge weights polynomial in n) and any constants $\epsilon \in (0,1), d > 0$, there is a randomized algorithm that runs in $O(m + n \log^5 n)$-time, and produces a graph G_ϵ containing $O(n \log n/\epsilon^2)$ edges in expectation, where $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$.*

We give a new algorithm for unweighted, simple graphs that reduces the running time to the optimal $O(m)$ (in expectation) without increasing the number of edges in the cut sparsifier. In the rest of the paper, when we talk of an unweighted graph, we mean a simple graph with no parallel edges and no edge weights (alternatively, every edge has unit weight).

THEOREM 1.22. *For any unweighted input graph G and any constants $\epsilon \in (0,1), d > 0$, there is a randomized algorithm with runtime $O(m)$ in expectation that produces a graph G_ϵ containing $O(n \log n/\epsilon^2)$ edges in expectation, where $G_\epsilon \in (1 \pm \epsilon)G$ with probability at least $1 - n^{-d}$.*

1.5. Roadmap. This paper is organized as follows. Section 2 gives proofs of a cut counting theorem which is the main technical tool that we use to prove properties of the sparsification framework (Theorem 1.14) in section 3. Applications of the framework to various sampling schemes appear in section 4. Finally, we present sparsification algorithms in section 5.

2. Counting cut projections. One of our main ingredients is a natural generalization of the following cut counting theorem.

THEOREM 2.1 (see Karger [19, 21]). *For any $\alpha \geq 1$, the number of cuts of value at most $\alpha\lambda$ in a graph is at most $n^{2\alpha}$, where λ is the minimum value of a cut in the graph.*

To state our generalization, we need some definitions.

DEFINITION 2.2. *An edge is said to be* k-heavy *if its connectivity 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 for a larger value of α, the large number of cuts of size $\alpha\lambda$ predicted by Karger's theorem arises from *many* distinct k-projections of these cuts for small values of k, whereas there are *few* distinct k-projections of these cuts for large values of k.

THEOREM 2.3. *For any $k \geq \lambda$ and any $\alpha \geq 1$, the number of distinct k-projections in cuts of value at most αk in a graph is at most $n^{2\alpha}$, where λ is the minimum value of a cut in the graph.*

Before proceeding further, we need to introduce the splitting-off operation.

DEFINITION 2.4. *The* splitting-off *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 k_{st} between any two vertices $s,t \neq v$.*

A key technical tool in our proof of Theorem 2.3 is a theorem of Mader [28] on the feasibility of the splitting-off operation, whose statement requires us to define cut edges first.

DEFINITION 2.5. *A* cut edge *is an edge whose removal separates a connected graph into two disconnected components.*

THEOREM 2.6 (see Mader [28]). *Let $G = (V, E)$ be a connected graph where $v \in V$ is a vertex that has degree $\neq 3$ and is not incident to any cut edge. 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 2.3, we may assume that G is Eulerian and does not have any cut edge. Therefore, Theorem 2.6 applies to our graph. The operation of splitting-off of edges can also be extended to vertices.

DEFINITION 2.7. *The* splitting-off *operation on an even-degree vertex v repeatedly performs admissible splitting-off operations on the edges incident on v until v becomes an isolated vertex.*

Note that Theorem 2.6 implies that we can split-off any vertex in a Eulerian graph with no cut edge.

Our proof strategy for Theorem 2.3 is to give an algorithm (Algorithm 1) with the following property, which immediately implies Theorem 2.3. Here, $q(F)$ denotes the minimum value of a cut whose k-projection is F.

LEMMA 2.8. *For any k-projection F with $q(F) \leq \alpha k$, Algorithm 1 outputs F with probability at least $n^{-2\alpha}$.*

To describe Algorithm 1, we need an additional definition.

DEFINITION 2.9. *A vertex is said to be* k-heavy *if it is incident to a k-heavy edge; otherwise, it is* k-light.

As a preprocessing step, Algorithm 1 splits-off all k-light vertices in G. Since Algorithm 1 preserves Eulerianness in G and does not introduce any cut edge, this step (and subsequent splitting-off operations) is feasible. Next, it performs a set of iterations, where in each iteration it contracts an edge selected uniformly at random (where an edge e of weight w_e is replaced by w_e parallel edges), removes all self-loops, and splits-off any vertices that may have become k-light as a result of the contraction. The iterations terminate when at most $\lceil 2\alpha \rceil$ vertices are left in the graph. At this point, the algorithm outputs the k-projection of a cut selected uniformly at random. Note that the algorithm adds new edges to G via the splitting-off process. All new edges are treated as k-light irrespective of their connectivity. Therefore, the k-projection of a cut that is output by the algorithm does not include any new edge.

When $k = \lambda$, there is no k-light vertex and Algorithm 1 reduces to a random contraction algorithm which was used by Karger to prove Theorem 2.1. Our 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.

To prove Lemma 2.8, we fix a k-projection F with $q(F) \leq \alpha k$. We will show that the following invariants are maintained by Algorithm 1 with good probability:

- (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 edge that is k-heavy (with respect to the initial graph) has connectivity at least k (in the remaining graph).

Algorithm 1. An algorithm for proving bound on cut projections.

procedure Contract($G,\ k,\ \alpha$):
input: A graph $G = (V, E)$, a parameter $k \geq \lambda$ where λ is the weight of a minimum cut in G, and an approximation factor α
RemoveLight($G,\ k$)
while there are more than $\lceil 2\alpha \rceil$ vertices remaining **do**
  Pick an edge e uniformly at random
  Contract e and remove any self-loops
  RemoveLight($G,\ k$)
end while
return the k-projection of a cut selected uniformly at random

function RemoveLight($G,\ k$):
input: A graph $G = (V, E)$, a parameter k
while there exists a k-light vertex v in G **do**
  Perform admissible splitting-off at v until v becomes an isolated vertex
  Remove v
end while
return G

In Algorithm 1, modifications to the graph are due to admissible splitting-offs, contraction of edges, and removal of self-loops. Clearly, removing self-loops does not affect the invariants. For the splitting-off operation, we note that

- (I1) is preserved because we only split-off k-light edges,
- (I2) is preserved because splitting-off never increases the size of any cut, and
- (I3) is preserved because we only split-off at a light vertex and the splitting-offs are admissible.

Finally, we consider edge contraction.

LEMMA 2.10. *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 $q(F)$ is the value of C. Note that (I1) and (I2) are preserved if the contracted edge e is not in cut C. Since e is picked uniformly at random, the probability that e is in C is at most $\frac{q(F)}{kr/2} \leq 2\alpha/r$. ∎

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

the execution of Algorithm 1, and F is the output, is at least

$$\begin{aligned}
&\left(1-\frac{2\alpha}{r_0}\right)\left(1-\frac{2\alpha}{r_1}\right)\dots\left(1-\frac{2\alpha}{\lceil 2\alpha\rceil+1}\right)2^{-(\lceil 2\alpha\rceil-1)}\\
&\geq \left(1-\frac{2\alpha}{n}\right)\left(1-\frac{2\alpha}{n-1}\right)\dots\left(1-\frac{2\alpha}{\lceil 2\alpha\rceil+1}\right)2^{-(\lceil 2\alpha\rceil-1)}\\
&\geq \left(1-\frac{\lceil 2\alpha\rceil}{n}\right)\left(1-\frac{\lceil 2\alpha\rceil}{n-1}\right)\dots\left(1-\frac{\lceil 2\alpha\rceil}{\lceil 2\alpha\rceil+1}\right)2^{-(\lceil 2\alpha\rceil-1)}\\
&= \frac{\lceil 2\alpha\rceil(\lceil 2\alpha\rceil-1)\dots 1}{n(n-1)\dots(n-\lceil 2\alpha\rceil+1)}\cdot 2^{-(\lceil 2\alpha\rceil-1)}\\
&\geq n^{-\lceil 2\alpha\rceil}.
\end{aligned}$$

This proves the bound for half-integral α. For extension to arbitrary α, we need to use generalized binomial coefficients. This generalization is exactly identical to the corresponding generalization for Karger's cut counting theorem, and the reader is referred to Corollary A.7 in [20] for the details.

This completes the proof of Lemma 2.8, which implies Theorem 2.3. Note that this theorem reduces to Karger's cut counting theorem by setting $k = \lambda$. Given the numerous applications of Karger's theorem (e.g., [5, 16, 20, 34]), we believe that our generalization may be of independent interest.

3. The general sparsification framework. In this section, we will prove Theorem 1.14. We reuse the notation defined in section 1.4.1. Recall that the Π-connectivity property ensures that every edge in F_i has connectivity at least π_i in the subgraph $G_i = (V, E_i)$. Also, the γ-overlap property ensures that for any cut C,

$$\sum_{i=0}^{\Lambda} \frac{e_i^{(C)} 2^{i-1}}{\pi_i} \leq \gamma \cdot e^{(C)},$$

where $e^{(C)}$ and $e_i^{(C)}$ denote the value of cut C in graphs G and G_i, respectively.

We also introduce some additional notation. For any cut C, let

$$F_i^{(C)} = F_i \cap C \quad \text{and} \quad E_i^{(C)} = E_i \cap C;$$

correspondingly, let

$$f_i^{(C)} = |F_i^{(C)}| \quad \text{and} \quad e_i^{(C)} = |E_i^{(C)}|.$$

Also, let $\widehat{f_i^{(C)}}$ be the sum of weights of all edges in $F_i^{(C)}$ that appear in the random graph G_ϵ. Note that

$$\mathbb{E}[\widehat{f_i^{(C)}}] = f_i^{(C)}.$$

We first prove a key lemma.

LEMMA 3.1. *For any fixed i, with probability at least $1 - 4/n^3$, every cut C in G satisfies*

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

Proof. If $f_i^{(C)} = 0$, then $\widehat{f_i^{(C)}} = 0$ and the lemma is trivial, so assume that $f_i^{(C)} > 0$. Thus, C contains some edge $e \in F_i$, which must be π_i-heavy in G_i due to the Π-connectivity property. It follows that

$$e_i^{(C)} \geq \pi_i.$$

Let $\mathcal{C}_{ij}$ be the set of all cuts C such that

$$\pi_i \cdot 2^j \leq e_i^{(C)} \leq \pi_i \cdot 2^{j+1} - 1 \quad \text{for} \quad j \geq 0.$$

We will prove that with probability at least $1 - 2n^{-3\cdot 2^j}$, all cuts in $\mathcal{C}_{ij}$ satisfy the property of the lemma. The lemma then follows by using the union bound over j (keeping i fixed) since

$$2n^{-3} + 2n^{-6} + \cdots + 2n^{-3\cdot 2^j} + \cdots \leq 4n^{-3}.$$

Suppose $C \in \mathcal{C}_{ij}$. 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 $F_i^{(C)} - X_i^{(C)}$ retain their weight exactly in G_ϵ, it is sufficient to show that with probability at least $1 - 2n^{-2^{j+1}}$,

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

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 A.1 with the lower bound on probabilities

$$p = \frac{112\gamma \ln n}{0.38 \cdot 2^{i+1}\epsilon^2}.$$

There are two cases. The first case is that

$$x_i^{(C)} \leq \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma}.$$

Then, for any $X_i^{(C)}$ where $C \in \mathcal{C}_{ij}$, by Theorem A.1, we have

$$\begin{aligned}
\mathbb{P}\left[\left|x_i^{(C)} - \widehat{x_i^{(C)}}\right| > \left(\frac{\epsilon}{2}\right)\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma}\right] &< 2\exp\left(-\frac{0.38\epsilon^2}{4}\left(\frac{112\gamma \ln n}{0.38 \cdot 2^{i+1}\epsilon^2}\right)\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma}\right) \\
&\leq 2\exp\Big(-\frac{7 \cdot e_i^{(C)} \ln n}{\pi_i}\Big) \\
&\leq 2\exp(-7 \cdot 2^j \ln n),
\end{aligned}$$

since $e_i^{(C)} \geq \pi_i \cdot 2^j$ for any $C \in \mathcal{C}_{ij}$. The second case is that

$$x_i^{(C)} > \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma}. \tag{3.1}$$

Then, for any $X_i^{(C)}$ where $C \in \mathcal{C}_{ij}$, by Theorem A.1, we have

$$\begin{aligned}\mathbb{P}\left[\left|x_i^{(C)} - \widehat{x_i^{(C)}}\right| > \left(\frac{\epsilon}{2}\right) x_i^{(C)}\right] &< 2\exp\left(-\frac{0.38\epsilon^2}{4}\left(\frac{112\gamma \ln n}{0.38 \cdot 2^{i+1}\epsilon^2}\right) x_i^{(C)}\right) \\ &< 2\exp\Big(-\frac{7 \cdot e_i^{(C)} \ln n}{\pi_i}\Big) \qquad \text{(by (3.1))} \\ &\leq 2\exp(-7 \cdot 2^j \ln n),\end{aligned}$$

since $e_i^{(C)} \geq 2^j \pi_i$ by the assumption that $C \in \mathcal{C}_{ij}$. Thus, we have proved that

$$\begin{aligned}\mathbb{P}\left[\left|x_i^{(C)} - \widehat{x_i^{(C)}}\right| > \left(\frac{\epsilon}{2}\right) \max\left(\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma}, x_i^{(C)}\right)\right] &< 2e^{-7 \cdot 2^j \ln n} \\ &= 2n^{-7 \cdot 2^j}\end{aligned}$$

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 2.3 applied to graph G_i with $k = \pi_i$ and $\alpha k = \pi_i \cdot 2^{j+1} - 1$, the number of distinct $X_i^{(C)}$ sets for cuts $C \in \mathcal{C}_{ij}$ is at most

$$n^{2\alpha k/k} < n^{4 \cdot 2^j}.$$

Using the union bound over these distinct $X_i^{(C)}$ edge sets, we conclude that with probability at least $1 - 2n^{-3 \cdot 2^j}$, all cuts in $\mathcal{C}_{ij}$ satisfy the property of the lemma. ∎

We now use the above lemma to prove Theorem 1.14. Lemma 3.1 bounds the sampling error for a fixed i. Applying a naïve union bound would incur an error probability that depends on $\Lambda = \lfloor \lg \max_{e \in E}\{\lambda_e\} \rfloor$. However, we observe that since there are at most n^2 distinct edges in G, the number of nonempty sets F_i is also at most n^2. This allows us to use the union bound over these values of i only, and bound the overall error probability to at most $4/n$.

Let $e^{(C)}$ and $\widehat{e^{(C)}}$ be the weight of edges crossing a cut C in G and G_ϵ, respectively. As we just argued, the conclusion of Lemma 3.1 holds for every value of i with probability at least $1 - 4/n$. Therefore,

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

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

$$\begin{aligned}|\widehat{e^{(C)}} - e^{(C)}| &= \left|\sum_{i=0}^{\Lambda} \widehat{f_i^{(C)}} - \sum_{i=0}^{\Lambda} f_i^{(C)}\right| \\ &\leq \sum_{i=0}^{\Lambda} |\widehat{f_i^{(C)}} - f_i^{(C)}| \\ &\leq \frac{\epsilon}{2} \sum_{i=0}^{\Lambda} \max\left(\frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma}, f_i^{(C)}\right) \\ &\leq \frac{\epsilon}{2}\left(\sum_{i=0}^{\Lambda} \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma} + \sum_{i=0}^{\Lambda} f_i^{(C)}\right) \\ &\leq \epsilon \cdot e^{(C)},\end{aligned}$$

since

$$\sum_{i=0}^{\Lambda} \frac{e_i^{(C)} \cdot 2^{i-1}}{\pi_i \cdot \gamma} \leq e^{(C)}$$

by the γ-overlap property and

$$\sum_{i=0}^{\Lambda} f_i^{(C)} = e^{(C)}$$

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

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

where w_e is the multiplicity of edge e in G. The bound follows by substituting the value of p_e.

This completes the proof of Theorem 1.14.

4. Sparsification by edge compression. In this section, we present edge compression schemes using various connectivity parameters and apply the sparsification framework to show that they yield cut sparsifiers.

4.1. Compression using edge connectivities. First, we use the sparsification framework to show Theorem 1.15. Although the theorem statement makes no reference to NI forests, our proof will use them since they allow for a convenient application of the general framework. Is it also possible to use a direct proof that avoids NI forests and instead certifies connectivity of edges using the entire graph, but we omit the details for the sake of brevity.

For any edge $e = (u, v)$, set λ_e to the value $\kappa_e \leq k_e$. Let

$$\gamma = 3 + \lg n \quad \text{and} \quad \pi_i = 2^{i-1}.$$

Recall that F_i is defined as the set of all edges e with

$$2^i \leq \lambda_e \leq 2^{i+1} - 1.$$

For any $i \geq 1 + \lg n$, let G_i contain all edges in NI forests $T_{2^{i-1-\lg n}}, T_{2^{i-1-\lg n}+1}, \ldots, T_{2^{i+1}-1}$ 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.

LEMMA 4.1. *The γ-overlap property is satisfied by the above definitions.*

Proof. 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 $2 + \lg n$ different values of i. This proves the γ-overlap property. ∎

To prove the Π-connectivity property, we will use the following fact that follows from the definition of NI forests.

FACT 4.2. *Let $T_1, T_2, \ldots$ be an NI forest packing of a graph $G = (V, E)$. For any pair of vertices $u, v \in V$ and for any $i \geq 1$, u, v are at least $\min(k_{uv}, i)$-connected in the first i NI forests, i.e., in $T_1 \cup T_2 \cup \ldots \cup T_i$.*

Proof. Consider any cut separating u, v in G. Each such cut contains at least k_{uv} edges. For any NI forest T_j, $j \le i$, either there is at least one edge of the cut in T_j, or all edges in the cut are already contained in $T_1 \cup T_2 \cup \ldots \cup T_{j-1}$. It follows that $T_1 \cup T_2 \cup \ldots \cup T_i$ contains at least $\min(k_{uv}, i)$ edges in the cut. ∎

LEMMA 4.3. *The* Π*-connectivity property is satisfied by the above definitions.*

Proof. Note that since $\kappa_e = \lambda_e \le k_e$, the connectivity of any edge in F_i satisfies $k_e \ge 2^i$. Then, from Fact 4.2, any edge $e \in F_i$ is at least 2^i-heavy in the union of 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 the Π-connectivity property. ∎

Theorem 1.15 now follows from the above lemmas and Corollary 1.7 applied to Theorem 1.14. Note that Corollary 1.17 follows immediately from Lemma 1.5, Theorem 1.15, and Lemma 1.8; hence, we will not consider sampling by edge conductances separately.

4.2. Compression using edge strengths. Now, we use the sparsification framework to show the result of Benczúr and Karger on compression using edge strengths (Theorem 1.12), up to constant factors.

For any edge e, set λ_e to its strength s_e. Let

$$\gamma = 1 \quad \text{and} \quad \pi_i = 2^\Lambda \quad \text{for all } i,$$

where, as usual, $\Lambda = \lfloor \lg \max_{e \in E}\{\lambda_e\} \rfloor$. Let G_i contain all edges in F_r for all $r \ge i$, where each edge in F_r is replicated $2^{\Lambda - r}$ times. (Recall that replication of edges is allowed in G_i, which are only used in the analysis and not in the actual compression algorithm.)

Let us introduce the following notation. Suppose that C is a cut containing an edge $e \in F_i$. Then C_i denotes the corresponding cut in G_i (i.e., with the same bipartition of vertices). Recall that $f_i^{(C)}$ and $e_i^{(C)}$, respectively, denote the number of edges in $F_i \cap C$ and in C_i, respectively.

LEMMA 4.4. *The* γ*-overlap property is satisfied by the above definitions.*

Proof. The proof amounts to the following calculation:

$$\begin{aligned}
\sum_{i=0}^{\Lambda} \frac{e_i^{(C)} 2^{i-1}}{\pi_i} &= \sum_{i=0}^{\Lambda} \sum_{r=i}^{\Lambda} \frac{f_r^{(C)} 2^{\Lambda - r} 2^{i-1}}{2^\Lambda} \\
&= \sum_{i=0}^{\Lambda} \sum_{r=i}^{\Lambda} \frac{f_r^{(C)}}{2^{r-i+1}} \\
&= \sum_{r=0}^{\Lambda} \sum_{i=0}^{r} \frac{f_r^{(C)}}{2^{r-i+1}} \\
&= \sum_{r=0}^{\Lambda} f_r^{(C)} \sum_{i=0}^{r} \frac{1}{2^{r-i+1}} \\
&< \sum_{r=0}^{\Lambda} f_r^{(C)} \\
&= e^{(C)}.
\end{aligned}$$

∎

LEMMA 4.5. *The* Π*-connectivity property is satisfied by the above definitions.*

To prove this lemma, we use the following property of edge strengths [7].

LEMMA 4.6. *The strength of an edge does not decrease even if all edges with lower strength are removed from the graph.*

We need to show that the number of edges in C_i is at least 2^Λ to prove Lemma 4.5. Let the maximum edge strength in C be k_C, where

$$2^j \le k_C \le 2^{j+1} - 1$$

for some $j \ge i$. By Lemma 4.6, C_i contains at least 2^j distinct edges of G, each of which is replicated at least $2^{\Lambda - j}$ times. Thus, C_i contains at least 2^Λ edges. This completes the proof of Lemma 4.5.

Theorem 1.12 (with a different constant) now follows from the above lemmas and Lemma 1.6 applied to Theorem 1.14.

4.3. Compression using NI indices. Now, we use the sparsification framework to show Theorem 1.19.

For any edge $e = (u, v)$, set λ_e to its NI index ℓ_e. Let

$$\gamma = 2 \quad \text{and} \quad \pi_i = 2^{i-1}.$$

For any $i \ge 1$, define G_i to be the union of the set of edges in NI forests $T_{2^{i-1}}$, $T_{2^{i-1}+1}$, $\ldots$, T_{2^i-1} (call this set of edges Y_i) and all edges in F_i. (Note that Y_i may contain parallel edges.) Let G_0 only contain edges in F_0.

LEMMA 4.7. *The* γ*-overlap property is satisfied by the above definitions.*

Proof. For any $i \ne j$,

$$F_i \cap F_j = Y_i \cap Y_j = \emptyset.$$

Thus, each edge appears in G_i for at most two different values of i, proving the γ-overlap property. ∎

LEMMA 4.8. *The* Π*-connectivity property is satisfied by the above definitions.*

Proof. For any edge $e \in F_i$, the endpoints of e are connected in each of $T_{2^{i-1}}$, $T_{2^{i-1}+1}, \ldots, T_{2^i-1}$ by definition of an NI forest packing. It follows that e is 2^{i-1}-heavy in G_i, thereby proving the Π-connectivity property. ∎

Theorem 1.19 now follows from the above lemmas and Lemma 1.9 applied to Theorem 1.14.

5. Cut sparsification algorithm. Recall that for graphs with polynomial edge weights, an implementation of edge compression using NI indices has a running time of $O(m)$ and produces a cut sparsifier containing $O(n \log^2 n/\epsilon^2)$ edges in expectation. In this section, we give a more refined algorithm for unweighted input graphs that will have the same time complexity, but will produce cut sparsifiers containing $O(n \log n/\epsilon^2)$ edges in expectation. This algorithm proves Theorem 1.22.

Before formally describing the algorithm, let us give some intuition about it. Let us abstractly view compression using NI indices as an iterative algorithm that finds a set of edges F_i in iteration i (these are the edges in NI forests $T_{2^i}, T_{2^i+1}, \ldots, T_{2^{i+1}-1}$ and are sampled with probability $\Theta(\log n/2^i)$) with the following properties:

- (P1) Each edge in F_i has connectivity of $\Theta(2^i)$ in F_{i-1}.
- (P2) The number of edges in F_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 F_i to have connectivity of $\Theta(2^i)$ in $H_{i-1} = (V, E_{i-1})$ where $E_{i-1} = F_{i-1} \cup F_i \cup \ldots$. 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 F_{i-1} from any connected component $\mathbf{C}$ of H_{i-1} is $O(2^i)$ times the number of components into which $\mathbf{C}$ decomposes in H_i. This stricter condition ensures that the expected number of edges in G_ϵ decreases to $\Theta(n \log n/\epsilon^2)$.

We also need to give a linear-time construction of F_i's satisfying the above properties. Iteration i runs on each component of H_i separately; we describe the algorithm for any one component $\mathbf{C}$. First, $(2^i + 1)$ NI forests $T_1, T_2, \ldots, T_{2^i+1}$ 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 F_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 F_i and retain the remaining edges for iteration $i + 1$. One may verify that the modified versions of properties (P1) and (P2) described above are satisfied by the F_i's constructed by this algorithm.

This algorithm, with a preprocessing 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 $O(n \log n/\epsilon^2)$. 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 F_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 2 (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 sparsification framework that we developed for independent compression of edges.

Now, we will formally describe our sparsification algorithm. The algorithm (Algorithm 2) 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 an NI forest packing 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 $E - F_0$.

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

Algorithm 2. The cut 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 $\rho = 1014 \ln n / 0.38\epsilon^2$.
if $m \leq 2\rho n$ **then**
  $G_\epsilon = G$ and terminate
end if
Construct NI forests $T_1, T_2, \ldots$ for G.
Set $i = 0$.
Set $X_0 = E$.
Set $F_0 = \cup_{1 \leq j \leq 2\rho} T_j$.
Set $Y_0 = X_0 - F_0$.
Add each edge in F_0 to G_ϵ with weight 1.

OuterLoop:
if $|Y_i| \leq 2\rho n$ **then**
  Add each edge in Y_i to G_ϵ with weight 2^{i-1} and terminate
end if
Sample each edge in Y_i with probability $1/2$ to construct X_{i+1}.
Increment i by 1.
Set $E_c = X_i$.
Set $V_c = V$.
Set $k_i = \rho \cdot 2^{i+1}$.

InnerLoop:
if $|E_c| \leq 2k_i|V_c|$ **then**
  Set $F_i = E_c$; $Y_i = X_i - E_c$.
  for all $e \in F_i$ **do**
    Set $\lambda_e = \rho \cdot 4^i$.
  end for
  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**.
end if

for all F_i created in the previous loops **do**
  for all edge $e \in F_i$ **do**
    Set $p_e = \min\left(1, \frac{16128}{1521} \cdot \frac{1}{4^i}\right)$.
    Generate r_e from **Binomial**$(2^i, p_e)$.
    if $r_e > 0$ **then**
      Add edge e to G_ϵ with weight r_e/p_e.
    end if
  end for
end for

with weight 2^{i-1} each, and terminate the algorithm.
- Otherwise, all edges in Y_{i-1} are sampled with probability $1/2$; call the sample X_i and let $G_i = (V, X_i)$.
- We identify a set of edges $F_i \subseteq X_i$ with 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 $Y_i = X_i - F_i$.
  - Each edge in Y_i is k_i-heavy in G_i.
- Each edge $e \in F_i$ is assigned the sampling probability

$$p_i \;=\; p_e \;=\; \min\left(\frac{112 \cdot (32/3)\ln n}{0.38\lambda_e(\epsilon/3)^2}, 1\right) \;=\; \min\left(\frac{16128}{1521} \cdot \frac{1}{4^i}, 1\right),$$

since $\lambda_e = \rho \cdot 4^i$ and $\rho = \frac{1014 \ln n}{0.38\epsilon^2}$.

The final phase consists of replacing each edge in F_i with 2^i parallel edges, and then compressing each edge independently with probability p_i. (Recall that in the interest of time complexity of the compression procedure, we will generate a Binomial random variable to represent the weight of the edge in the sparsifier.) The weighted graph formed by this compression procedure is the sparsifier G_ϵ.

We now give a short description of the subroutine that constructs the set F_i in the second phase of the algorithm. This subroutine 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 an NI forest packing for G_c and contract all edges in the $(k_i + 1)$st forest, where $k_i = \rho \cdot 2^{i+1}$, to obtain a new G_c. We terminate this iterative process when

$$|E_c| \leq 2k_i|V_c|.$$

The set of edges E_c that finally achieves this property forms F_i.

5.1. 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 Λ be the maximum value of i for which $F_i \neq \emptyset$; let

$$S = \left(\cup_{i=0}^{\Lambda} 2^i F_i\right) \cup 2^\Lambda Y_\Lambda$$

and $G_S = (V, S)$. Then, we prove the following two theorems, which together yield Theorem 5.1 using the union bound. (Observe that since $\epsilon < 1$, $(1 + \epsilon/3)^2 \leq 1 + \epsilon$ and $(1 - \epsilon/3)^2 \geq 1 - \epsilon$.)

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. ∎

We will now prove Theorem 5.2. First, we state a property of edge sampling. Let $R \subseteq Q$ be subsets of edges such that R is π-heavy in (V, Q). Suppose each edge $e \in R$ is sampled with probability p, and if selected, given a weight of $1/p$ to form a set of weighted edges $\widehat{R}$. Now, for any cut C in G, let

$$R^{(C)} = R \cap C, \quad Q^{(C)} = Q \cap C, \quad \text{and} \quad \widehat{R^{(C)}} = \widehat{R} \cap C,$$

respectively; also let the total weight of edges in $R^{(C)}$, $Q^{(C)}$, and $\widehat{R^{(C)}}$ be $r^{(C)}$, $q^{(C)}$, and $\widehat{r^{(C)}}$, respectively. Then the following lemma holds.

LEMMA 5.5. *For any $\delta \in (0, 1]$ satisfying $\delta^2 p\pi \geq \frac{6 \ln n}{0.38}$,*

$$|r^{(C)} - \widehat{r^{(C)}}| \leq \delta q^{(C)}$$

for all cuts C, with probability at least $1 - 4/n^2$.

Proof. Let $\mathcal{C}_j$ be the set of all cuts C such that

$$2^j \cdot \pi \leq r^{(C)} \leq 2^{j+1} \cdot \pi - 1$$

for each $j \geq 0$. We will prove that with probability at least $1 - 2n^{-2^{j+1}}$, all cuts in $\mathcal{C}_j$ satisfy the property of the lemma. Then, the lemma follows by using the union bound over j since

$$2n^{-2} + 2n^{-4} + \cdots + 2n^{-2^{j+1}} + \cdots \leq 4n^{-2}.$$

We now prove the property for cuts $C \in \mathcal{C}_j$. Since each edge $e \in R^{(C)}$ is sampled with probability p in obtaining $\widehat{R^{(C)}}$, we can use Theorem A.1 with sampling probability p. Then, for any $R^{(C)}$ where $C \in \mathcal{C}_j$, by Theorem A.1, we have

$$\begin{aligned}
\mathbb{P}\left[\left|\widehat{r^{(C)}} - r^{(C)}\right| > \delta q^{(C)}\right] &< 2e^{-0.38 \cdot \delta^2 \cdot p \cdot q^{(C)}} \\
&\leq 2e^{-0.38 \cdot \delta^2 \cdot p \cdot \pi \cdot 2^j} \\
&\leq 2e^{-6 \cdot 2^j \ln n} \\
&= 2n^{-6 \cdot 2^j},
\end{aligned}$$

since $q^{(C)} \geq \pi \cdot 2^j$ for any $C \in \mathcal{C}_j$. Since each edge in $R^{(C)}$ is π-heavy in (V, Q), Theorem 2.3 ensures that the number of distinct $R^{(C)}$ sets for cuts $C \in \mathcal{C}_j$ is at most

$$n^{2\left(\frac{\pi \cdot 2^{j+1}}{\pi}\right)} = n^{4 \cdot 2^j}.$$

Using the union bound over these distinct $R^{(C)}$ edge sets, we conclude that with probability at least $1 - 2n^{-2^{j+1}}$, all cuts in $\mathcal{C}_j$ satisfy the property of the lemma. ∎

To obtain Corollary 5.6, we use the following settings in Lemma 5.5:

$$R = Y_i, \quad Q = X_i, \quad \widehat{R} = 2X_{i+1}, \quad \delta = \frac{\epsilon/13}{2^{i/2}}, \quad p = 1/2, \quad \text{and} \quad \pi = \rho \cdot 2^{i+1}.$$

COROLLARY 5.6. *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* $F_i \cap C$*.*

Next, we show the following fact.

FACT 5.7. *Let* $x \in (0, 1]$ *and* $r_i = 13 \cdot 2^{i/2}$*. Then, for any* $k \geq 0$*,*

$$\prod_{i=0}^{k}(1 + x/r_i) \leq 1 + x/3,$$
$$\prod_{i=0}^{k}(1 - x/r_i) \geq 1 - x/3.$$

Proof. We prove by induction on k. For $k = 0$, the property trivially holds. Suppose the property holds for $k - 1$. Then,

$$\begin{aligned}
\prod_{i=0}^{k}(1 + x/r_i) &= \prod_{i=0}^{k}\left(1 + \frac{x}{13 \cdot 2^{i/2}}\right) \\
&= (1 + x/13) \cdot \prod_{i=1}^{k}\left(1 + \frac{x/\sqrt{2}}{13 \cdot 2^{(i-1)/2}}\right) \\
&\leq (1 + x/13) \cdot (1 + x/(3\sqrt{2})) \\
&\leq 1 + x/3 \\
\prod_{i=0}^{k}(1 - x/r_i) &= \prod_{i=0}^{k}\left(1 - \frac{x}{13 \cdot 2^{i/2}}\right) \\
&= (1 - x/13) \cdot \prod_{i=1}^{k}\left(1 - \frac{x/\sqrt{2}}{13 \cdot 2^{(i-1)/2}}\right) \\
&\geq (1 - x/13) \cdot (1 - x/(3\sqrt{2})) \\
&\geq 1 - x/3.
\end{aligned}$$

∎

We now use Fact 5.7 and Corollary 5.6 to prove the following lemma.

LEMMA 5.8. *Let*

$$S_j = \left(\cup_{i=j}^{\Lambda} 2^{i-j} F_i\right) \cup 2^{\Lambda - j} Y_\Lambda$$

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. 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

$$X_i^{(C)} = X_i \cap C, \quad Y_i^{(C)} = Y_i \cap C, \quad \text{and} \quad F_i^{(C)} = F_i \cap C,$$

and let their respective sum of weights be $x_i^{(C)}$, $y_i^{(C)}$, and $f_i^{(C)}$.

Since $\Lambda \leq n-1$, we can use the union bound on Corollary 5.6 to conclude that with probability at least $1 - 4/n$, for every $0 \leq i \leq \Lambda$ and for all cuts C,

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

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

$$\begin{aligned}
s_j^C &= 2^{\Lambda-j}y_\Lambda^{(C)} + 2^{\Lambda-j}f_\Lambda^{(C)} + 2^{\Lambda-1-j}f_{\Lambda-1}^{(C)} + \cdots + f_j^{(C)} \\
&= 2^{\Lambda-j}x_\Lambda^{(C)} + 2^{\Lambda-1-j}f_{\Lambda-1}^{(C)} + \cdots + f_j^{(C)} \\
&\quad \text{since } y_\Lambda^{(C)} + f_\Lambda^{(C)} = x_\Lambda^{(C)} \\
&= 2^{\Lambda-1-j}(2x_\Lambda^{(C)} + f_{\Lambda-1}^{(C)}) + (2^{\Lambda-2-j}f_{\Lambda-2}^{(C)} + \cdots) \\
&\leq (1+\epsilon/r_{\Lambda-1})2^{\Lambda-1-j}x_{\Lambda-1}^{(C)} + (2^{\Lambda-2-j}f_{\Lambda-2}^{(C)} + \cdots) \\
&\leq (1+\epsilon/r_{\Lambda-1})(2^{\Lambda-1-j}x_{\Lambda-1}^{(C)} + 2^{\Lambda-2-j}f_{\Lambda-2}^{(C)} + \cdots) \\
&\cdots \\
&\leq (1+\epsilon/r_{\Lambda-1})(1+\epsilon/r_{\Lambda-2})\ldots(1+\epsilon/r_j)x_j^{(C)} \\
&\leq (1+(\epsilon 2^{-j/2})/r_{\Lambda-1-j})(1+(\epsilon 2^{-j/2})/r_{\Lambda-2-j})\ldots \\
&\quad \ldots(1+(\epsilon 2^{-j/2})/r_0)x_j^{(C)} \quad \text{since } r_{j+i} = r_i \cdot 2^{j/2} \\
&\leq (1+(\epsilon/3)2^{-j/2})x_j^{(C)} \quad \text{by Fact 5.7.}
\end{aligned}$$

Similarly, we can show that

$$s_j^C \geq (1-(\epsilon/3)2^{-j/2})x_j^{(C)}. \qquad \square$$

Finally, we observe that Theorem 5.2 follows from Lemma 5.8 if we set $j = 0$.

Now, we will prove Theorem 5.3. First, observe that edges $F_0 \cup 2^\Lambda Y_\Lambda$ 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 $\psi(i)$'s define doubling intervals on a function of i, which will be useful later in invoking the sparsification framework.) Note that for any j, $\psi(i) = j$ for at most one value of i. Then, for any $j \geq 1$,

$$R_j = F_i \quad \text{if } j = \psi(i)$$

$$\text{and} \quad R_j = \emptyset \quad \text{if there is no } i \text{ such that } j = \psi(i).$$

We set

$$\gamma = 32/3, \quad \pi_j = \rho \cdot 4^\Lambda, \quad \text{and for any } j \geq 1,\ Q_j = (V, W_j),$$

where

$$W_j = \cup_{i-1\le r\le\Lambda} 4^{\Lambda-r+1}2^r F_r \quad \text{if } R_j \neq \emptyset \text{ and } j = \psi(i),$$
$$\text{and } W_j = \emptyset \quad \text{if } R_j = \emptyset.$$

The following lemma ensures Π-connectivity.

LEMMA 5.9. *With probability at least* $1 - 4/n$*, every edge* $e \in F_i = R_{\psi(i)}$ *for each* $i \ge 1$ *is* π*-heavy in* $Q_{\psi(i)}$*, where* $\pi = \rho \cdot 4^\Lambda$.

Proof. Consider any edge $e \in F_i$. Since $F_i \subseteq Y_{i-1}$, Lemma 5.4 ensures that e is $\rho \cdot 2^i$-heavy in $G_{i-1} = (V, X_{i-1})$, and therefore $\rho \cdot 2^{2i-1}$-heavy in $(V, 2^{i-1}X_{i-1})$. Since $\epsilon \le 1$, Lemma 5.8 ensures that with probability at least $1 - 4/n$, the weight of each cut in $(V, 2^{i-1}X_{i-1})$ is preserved up to a factor of 2 in $Z_i = (V, \cup_{i-1\le r\le\Lambda} 2^r F_r)$. Thus, e is $\rho \cdot 4^{i-1}$-heavy in Z_i.

Consider any cut C containing $e \in F_i$. We need to show that the weight of this cut in $Q_{\psi(i)}$ is at least $\rho \cdot 4^\Lambda$. Let the maximum λ_a of an edge a in C be $\rho \cdot 4^{k_C}$, for some $k_C \ge i$. By the above proof, a is $\rho \cdot 4^{k_C-1}$-heavy in Z_{k_C}. Then, the total weight of edges crossing cut C in $Q_{\psi(k_C)}$ is at least

$$\rho \cdot 4^{k_C-1} \cdot 4^{\Lambda-k_C+1} = \rho \cdot 4^\Lambda.$$

Since $k_c \ge i$, $\psi(k_C) \ge \psi(i)$ and $Q_{\psi(k_C)}$ is a subgraph of $Q_{\psi(i)}$. Therefore, the total weight of edges crossing cut C in $Q_{\psi(i)}$ is at least $\rho \cdot 4^\Lambda$. ∎

We now prove the γ-overlap property. For any cut C, let $f_i^{(C)}$ and $w_i^{(C)}$, respectively, denote the total weight of edges in $F_i \cap C$ and $W_{\psi(i)} \cap C$, respectively. Further, let the number of edges in $\cup_{i=0}^{\Lambda} 2^i F_i \cap C$ be $f^{(C)}$. Then, we have the following bound:

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

Using Theorem 1.14 with $\gamma = 32/3$, we conclude the proof of Theorem 5.3.

5.2. Size of the sparsifier. 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 $\sum_{e \in F_i} \frac{w_e}{\lambda_e}$ (where w_e is the number of parallel copies of e in the Binomial sampling step) 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

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

since we can have at most n singleton components. It follows from Theorem 1.14 that the expected number of edges in G_ϵ is $O(n \log n/\epsilon^2)$.

5.3. 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. We will show that all of 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 $Y_i \subset X_i$, $\mathbb{E}[|X_i|] = \mathbb{E}[|Y_{i-1}|]/2 \leq \mathbb{E}[|X_{i-1}|]/2$, 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., [17]).

We are now left to prove that all, except the first step, of iteration i in phase 2 takes $O(|X_i| + n \log n)$ time. Each iteration of the else block takes $O(|V_c| \log n + |E_c|)$ time for the current $G_c = (V_c, E_c)$. So, the last invocation of the else block takes at most $O(|X_i| + n \log n)$ time. In any other invocation, $|E_c| = \Omega(|V_c| \log n)$ and hence the time spent is $O(|E_c|)$. Now, consider an iteration that begins with $|E_c| > 2k_i \cdot |V_c|$. Note that E_c for the next iteration (denoted by E_c') comprises only edges in the first k_i NI forests constructed in the current iteration. Hence,

$$|E_c'| \leq k_i \cdot |V_c| < |E_c|/2.$$

Since $|E_c|$ decreases by a factor of 2 from one invocation of the else block to the next, the total time over all invocations of the else block is $O(|X_i| + n \log n)$.

6. Conclusion. In this paper, we gave a general sampling framework for cut sparsification and used it to show that various sampling schemes produce cut sparsifiers. In addition, we gave two algorithms for cut sparsification both of which run in $O(m)$ time and produce cut sparsifiers containing an expected $O(n \log n/\epsilon^2)$ edges for unweighted graphs and $O(n \log^2 n/\epsilon^2)$ edges for weighted graphs. For weighted graphs, using previously known algorithms for postprocessing, we can obtain cut sparsifiers with an expected $O(n \log n/\epsilon^2)$ edges in $O(m) + O(n \log^5 n)$ time. Several problems are left open by our work. For example, can the running time of the sparsification algorithm be improved to $O(m)$ even for weighted graphs? An interesting

combinatorial question is to remove the additional factor of $\log n$ in the sampling probability used in Theorem 1.15, i.e., sampling using edge connectivities.

Appendix A. A variant of Chernoff bounds. Our sparsification techniques in sections 3 and 5 require the following form of Chernoff bounds for a set of random variables with nonuniform sampling probabilities but uniform expectation.

THEOREM A.1. *Let $X_1, X_2, \ldots, X_n$ be n independent random variables such that X_i takes value $1/p_i$ with probability p_i and 0 otherwise. Then, for any p such that $p \le p_i$ for each i, any $\epsilon \in (0,1)$, and any $N \ge n$,*

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

We will derive Theorem A.1 using the following familiar form of the Chernoff bound, whose proof proceeds as in standard references [29, 30].

THEOREM A.2. *Let $Y_1, \ldots, Y_n$ be n independent random variables such that Y_i takes values in $[0,1]$. Let $\mu = \sum_{i=1}^{n} \mathbb{E}[Y_i]$ and $c = 2\ln(2) - 1 > 0.38$. Then, for all $\delta > 0$,*

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} Y_i - \mu\right| > \delta\mu\right] \;\le\; 2\exp(-c \cdot \min(1,\delta)\delta\mu).$$

Proof of Theorem A.1. Define $Y_i = p \cdot X_i$, so that $Y_i \in [0,1]$ and $\mathbb{E}[Y_i] = p$. Then $\mu = \sum_{i=1}^{n} \mathbb{E}[Y_i] = pn$. We apply Theorem A.2 with $\delta = \epsilon N/n$, obtaining

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} X_i - n\right| > \epsilon N\right] \;=\; \mathbb{P}\left[\left|\sum_{i=1}^{n} Y_i - \mu\right| > \delta\mu\right] \;\le\; 2\exp\big(-c \cdot \min(1,\delta)\delta\mu\big).$$

If $\delta \le 1$, then the bound is

$$2\exp\big(-c\delta^2\mu\big) \;=\; 2\exp(-c\epsilon^2 pN^2/n) \;<\; 2\exp(-0.38\epsilon^2 pN)$$

since $N \ge n$. If $\delta > 1$, then the bound is

$$2\exp\big(-c\delta\mu\big) \;=\; 2\exp(-c\epsilon pN) \;<\; 2\exp(-0.38\epsilon^2 pN)$$

since $\epsilon \in [0,1]$. ∎

Acknowledgments. The authors would like to thank David Karger and Telikepalli Kavitha for helpful discussions, Navin Goyal for pointing out the Rayleigh monotonicity principle, and anonymous reviewers for their helpful suggestions.

REFERENCES

[1] K. J. AHN AND S. GUHA, *Graph sparsification in the semi-streaming model*, in Automata, Languages and Programming. Part II, Lecture Notes in Comput. Sci. 5556, Springer, Berlin, 2009, pp. 328–338.

[2] K. J. AHN, S. GUHA, AND A. MCGREGOR, *Analyzing graph structure via linear measurements*, in Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2012, pp. 459–467.

[3] K. J. AHN, S. GUHA, AND A. MCGREGOR, *Graph sketches: Sparsification, spanners, and subgraphs*, in Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, ACM, New York, 2012, pp. 5–14.

[4] A. Andoni, J. Chen, B. Qin, R. Krauthgamer, D. P. Woodruff, and Q. Zhang, *On sketching quadratic forms*, in Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, ACM, New York, 2016, pp. 311–319.

[5] 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 Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2010, pp. 379–389.

[6] J. D. Batson, D. A. Spielman, and N. Srivastava, *Twice-Ramanujan sparsifiers*, in Proceedings of the 2009 ACM International Symposium on Theory of Computing, ACM, New York, 2009, pp. 255–262.

[7] A. A. Benczúr and D. R. Karger, *Approximating s-t minimum cuts in $\tilde{o}(n^2)$ time*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, ACM, New York, 1996, pp. 47–55.

[8] A. A. Benczúr and D. R. Karger, *Randomized approximation schemes for cuts and flows in capacitated graphs*, SIAM J. Comput., 44 (2015), pp. 290–319, https://doi.org/10.1137/070705970.

[9] B. Bollobás, *Modern Graph Theory*, Grad. Texts in Math. 184, Springer-Verlag, New York, 1998.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.

[11] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*, Carus Math. Monogr. 22, Mathematical Association of America, Washington, DC, 1984.

[12] W. S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi, *A general framework for graph sparsification*, in Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, ACM, New York, 2011, pp. 71–80.

[13] W. S. Fung and N. J. A. Harvey, *Graph Sparsification by Edge-connectivity and Random Spanning Trees*, preprint, https://arxiv.org/abs/1005.0265, 2010.

[14] A. Goel, M. Kapralov, and S. Khanna, *Graph Sparsification via Refinement Sampling*, preprint, https://arxiv.org/abs/1004.4915, 2010.

[15] A. Goel, M. Kapralov, and I. Post, *Single Pass Sparsification in the Streaming Model with Edge Deletions*, preprint, https://arxiv.org/abs/1203.4900, 2012.

[16] M. X. Goemans, N. J. A. Harvey, K. Jain, and M. Singh, *A Randomized Rounding Algorithm for the Asymmetric Traveling Salesman Problem*, preprint, https://arxiv.org/abs/0909.0941, 2009.

[17] V. Kachitvichyanukul and B. W. Schmeiser, *Binomial random variate generation*, Comm. ACM, 31 (1988), pp. 216–222.

[18] M. Kapralov and R. Panigrahy, *Spectral sparsification via random spanners*, in Proceedings of the 3rd Annual Innovations in Theoretical Computer Science Conference, ACM, New York, 2012, pp. 393–398.

[19] D. R. Karger, *Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm*, in Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993), ACM, New York, 1993, pp. 21–30.

[20] D. R. Karger, *Random sampling in cut, flow, and network design problems*, Math. Oper. Res., 24 (1999), pp. 383–413.

[21] D. R. Karger and C. Stein, *A new approach to the minimum cut problem*, J. ACM, 43 (1996), pp. 601–640.

[22] J. A. Kelner and A. Levin, *Spectral sparsification in the semi-streaming setting*, in Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2011, pp. 440–451.

[23] A. Kolla, Y. Makarychev, A. Saberi, and S.-H. Teng, *Subgraph sparsification and nearly optimal ultrasparsifiers*, in Proceedings of the 2010 ACM International Symposium on Theory of Computing, ACM, New York, 2010, pp. 57–66.

[24] I. Koutis, A. Levin, and R. Peng, *Improved spectral sparsification and numerical algorithms for SDD matrices*, in Proceedings of the 29th Annual International Symposium on Theoretical Aspects of Computer Science, Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2012, pp. 266–277.

[25] I. Koutis, G. L. Miller, and R. Peng, *Approaching optimality for solving SDD linear systems*, in Proceedings of the 51st Annual Symposium on Foundations of Computer Science—FOCS 2010, IEEE Computer Society, Los Alamitos, CA, 2010, pp. 235–244.

[26] I. Koutis, G. L. Miller, and R. Peng, *A nearly-$m\log n$ time solver for SDD linear systems*, in Proceedings of the 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011, IEEE Computer Society, Los Alamitos, CA, 2011, pp. 590–598.

[27] Y.-T. Lee and H. Sun, *Constructing linear-sized spectral sparsification in almost-linear time*, in Proceedings of the 56th Annual Symposium on Foundations of Computer Science—FOCS 2011, IEEE Computer Society, Los Alamitos, CA, 2015, pp. 250–269.
[28] W. Mader, *A reduction method for edge-connectivity in graphs*, Ann. Discrete Math., 3 (1978), pp. 145–164.
[29] C. McDiarmid, *Concentration*, in Probabilistic Methods for Algorithmic Discrete Mathematics, M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, eds., Springer, Berlin, Heidelberg, 1998, pp. 195–248.
[30] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, 1997.
[31] H. Nagamochi and T. Ibaraki, *Computing edge-connectivity in multigraphs and capacitated graphs*, SIAM J. Discrete Math., 5 (1992), pp. 54–66, https://doi.org/10.1137/0405004.
[32] H. Nagamochi and T. Ibaraki, *A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph*, Algorithmica, 7 (1992), pp. 583–596.
[33] D. Peleg and A. A. Schäffer, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116.
[34] S. Rao and S. Zhou, *Edge disjoint paths in moderately connected graphs*, SIAM J. Comput., 39 (2010), pp. 1856–1887, https://doi.org/10.1137/080715093.
[35] D. A. Spielman and N. Srivastava, *Graph sparsification by effective resistances*, SIAM J. Comput., 40 (2011), pp. 1913–1926, https://doi.org/10.1137/080734029.
[36] D. A. Spielman and S.-H. Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, in Proceedings of the 36th Annual ACM Symposium on Theory of Computing, ACM, New York, 2004, pp. 81–90.
[37] D. A. Spielman and S.-H. Teng, *Spectral sparsification of graphs*, SIAM J. Comput., 40 (2011), pp. 981–1025, https://doi.org/10.1137/08074489X.
[38] J. A. Tropp, *User-friendly tail bounds for sums of random matrices*, Found. Comput. Math., 12 (2012), pp. 389–434.
[39] Z. A. Zhu, Z. Liao, and L. Orecchia, *Spectral sparsification and regret minimization beyond matrix multiplicative updates*, in Proceedings of the 2015 ACM Symposium on Theory of Computing, ACM, New York, 2015, pp. 237–245.
[40] A. Zouzias, *A matrix hyperbolic cosine algorithm and applications*, in Automata, Languages, and Programming. Part I, Lecture Notes in Comput. Sci. 7391, Springer, Heidelberg, 2012, pp. 846–858.