Deterministic Mincut in Almost-Linear Time*

Jason Li

jmli@cs.cmu.edu

Carnegie Mellon University

Pittsburgh, PA, USA

ABSTRACT

We present a deterministic (global) mincut algorithm for weighted, undirected graphs that runs in $m^{1+o(1)}$ time, answering an open question of Karger from the 1990s. To obtain our result, we de-randomize the construction of the *skeleton* graph in Karger's near-linear time mincut algorithm, which is its only randomized component. In particular, we partially de-randomize the well-known Benczur-Karger graph sparsification technique by random sampling, which we accomplish by the method of pessimistic estimators. Our main technical component is designing an efficient pessimistic estimator to capture the cuts of a graph, which involves harnessing the expander decomposition framework introduced in recent work by Goranci et al. (SODA 2021). As a side-effect, we obtain a structural representation of all approximate mincuts in a graph, which may have future applications.

CCS CONCEPTS

• **Theory of computation** → **Sparsification and spanners**; **Graph algorithms analysis**.

KEYWORDS

minimum cut, deterministic algorithms, de-randomization, graph sparsification

ACM Reference Format:

Jason Li. 2021. Deterministic Mincut in Almost-Linear Time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3406325.3451114

1 INTRODUCTION

The minimum cut of an undirected, weighted graph $G = (V, E, w)$ is a minimum weight subset of edges whose removal disconnects the graph. Finding the mincut of a graph is one of the central problems in combinatorial optimization, dating back to the work of Gomory and Hu [5] in 1961 who gave an algorithm to compute the mincut of an n-vertex graph using $n - 1$ max-flow computations. Since then, a large body of research has been devoted to obtaining faster algorithms for this problem. In 1992, Hao and Orlin [7] gave a clever amortization of the $n - 1$ max-flow computations to match the running time of a single max-flow computation. Using the "push-relabel" max-flow algorithm of Goldberg and Tarjan [4], they obtained an overall running time of $O(mn \log(n^2/m))$ on an n-vertex, m-edge graph. Around the same time, Nagamochi and Ibaraki [16] (see also [17]) designed an algorithm that bypasses max-flow computations altogether, a technique that was further refined by Stoer and Wagner [20] (and independently by Frank in unpublished work). This alternative method yields a running time of $O(mn + n^2 \log n)$. Before 2020, these works yielding a running time bound of $\widetilde{O}(mn)$ were the fastest *deterministic* mincut algorithms for weighted graphs.

Starting with Karger's contraction algorithm in 1993 [9], a parallel body of work started to emerge in *randomized* algorithms for the mincut problem. This line of work (see also Karger and Stein [11]) eventually culminated in a breakthrough paper by Karger [10] in 1996 that gave an $O(m \log^3 n)$ time *Monte Carlo* algorithm for the mincut problem. Note that this algorithm comes to within poly-logarithmic factors of the optimal $O(m)$ running time for this problem. In this paper, Karger asks whether we can also achieve near-linear running time using a *deterministic* algorithm. Even before Karger's work, Gabow [2] showed that the mincut can be computed in $O(m + \lambda^2 n \log(n^2/m))$ (deterministic) time, where λ is the value of the mincut (assuming integer weights). Note that this result obtains a near-linear running time if λ is a constant, but in general, the running time can be exponential. Indeed, for general graphs, Karger's question remains open after more than 20 years. However, some exciting progress has been reported in recent years for special cases of this problem. In a recent breakthrough, Kawarabayashi and Thorup [12] gave the first near-linear time deterministic algorithm for this problem *for simple graphs*. They obtained a running time of $O(m \log^{12} n)$, which was later improved by Henzinger, Rao, and Wang [8] to $O(m \log^2 n \log\log^2 n)$, and then simplified by Saranurak [18] at the cost of $m^{1+o(1)}$ running time. From a technical perspective, Kawarabayashi and Thorup's work introduced the idea of using low conductance cuts to find the mincut of the graph, a very powerful idea that we also exploit in this paper.

In 2020, the author, together with Debmalya Panigrahi [13], obtained the first improvement to deterministic mincut for weighted graphs since the 1990s, obtaining a running time of $O(m^{1+\epsilon})$ plus

*Most of this work was done while the author was an intern at Microsoft Research, Redmond.

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06.
https://doi.org/10.1145/3406325.3451114

polylogarithmic calls to a deterministic exact $s-t$ max-flow algorithm. Using the fastest deterministic algorithm for weighted graphs of Goldberg and Rao [3], their running time becomes $\widetilde{O}(m^{1.5})$.[1] Their algorithm was inspired by the conductance-based ideas of Kawarabayashi and Thorup and introduced expander decompositions into the scene. While it is believed that a near-linear time algorithm exists for $s-t$ max-flow—which, if deterministic, would imply a near-linear time algorithm for deterministic mincut—the best max-flow algorithms, even for unweighted graphs, is still $m^{4/3+o(1)}$ [14]. For the deterministic, weighted case, no improvement since Goldberg-Rao [3] is known.

The main result of this paper is a new deterministic algorithm for mincut that does not rely on $s-t$ max-flow computations and achieves a running time of $m^{1+o(1)}$, answering Karger's open question.

Theorem 1.1. *There is a deterministic mincut algorithm for weighted, undirected graphs that runs in $m^{1+o(1)}$ time.*

1.1 Our Techniques

Our approach differs fundamentally from the one in [13] that relies on $s-t$ max-flow computations. At a high level, we follow Karger's approach and essentially de-randomize the single randomized procedure in Karger's near-linear time mincut algorithm [10], namely the construction of the *skeleton* graph, which Karger accomplishes through the Benczur-Karger graph sparsification technique by random sampling. We remark that our de-randomization does not recover a full $(1+\epsilon)$-approximate graph sparsifier, but the skeleton graph that we obtain is sufficient to solve the mincut problem.

Let us first briefly review the Benczur-Karger graph sparsification technique, and discuss the difficulties one encounters when trying to de-randomize it. Given a weighted, undirected graph, the sparsification algorithm samples each edge independently with a probability depending on the weight of the edge and the global mincut of the graph, and then re-weights the sampled edge accordingly. In traditional graph sparsification, we require that every cut in the graph has its weight preserved up to a $(1+\epsilon)$ factor. There are exponentially many cuts in a graph, so a naive union bound over all cuts does not work. Benczur and Karger's main insight is to set up a more refined union bound, layering the (exponentially many) cuts in a graph by their weight. They show that for all $\alpha \geq 1$, there are only $n^{c\alpha}$ many cuts in a graph whose weight is roughly α times the mincut, and each one is preserved up to a $(1+\epsilon)$ factor with probability $1-n^{-c'\alpha}$, for some constants $c' \gg c$. In other words, they establish a union bound layered by the α-approximate mincuts of a graph, for each $\alpha \geq 1$.

One popular method to de-randomize random sampling algorithms is through *pessimistic estimators*, which is a generalization of the well-known method of conditional probabilities. For the graph sparsification problem, the method of pessimistic estimators can be implemented as follows. The algorithm considers each edge one by one in some arbitrary order, and decides on the spot whether to keep or discard each edge for the sparsifier. To make this decision, the algorithm maintains a *pessimistic estimator*, which is a real number in the range $[0,1)$ that represents an upper bound on the probability of failure should the remaining undecided edges each be sampled independently at random. In many cases, the pessimistic estimator is exactly the probability upper bound that one derives from analyzing the random sampling algorithm, except conditioned on the edges kept and discarded so far. The algorithm makes the choice—whether to keep or discard the current edge—based on whichever outcome does not increase the pessimistic estimator; such a choice must always exist for the pessimistic estimator to be valid. Once all edges are processed, the pessimistic estimator must still be a real number less than 1. But now, since there are no more undecided edges, the probability of failure is either 0 or 1. Since the pessimistic estimator is an upper bound which is less than 1, the probability of failure must be 0; in other words, the set of chosen edges is indeed a sparsifier of the graph.

In order for this de-randomization procedure to be efficient, the pessimistic estimator must be quickly evaluated and updated after considering each edge. Unfortunately, the probability union bound in the Benczur-Karger analysis involves all cuts in the graph, and is therefore an expression of exponential size and too expensive to serve as our pessimistic estimator. To design a more efficient pessimistic estimator, we need a more compact, easy-to-compute union bound over all cuts of the graph. We accomplish this by grouping all cuts of the graph into two types: small cuts and large cuts.

Small cuts. Recall that our goal is to preserve cuts in the graph up to a $(1+\epsilon)$ factor. Let us first restrict ourselves to all α-approximate mincuts of the graph for some $\alpha = n^{o(1)}$. There can be $n^{\Omega(\alpha)}$ many such cuts, so the naive union bound is still too slow. Here, our main strategy is to establish a *structural representation* of all α-approximate mincuts of a graph, with the goal of deriving a more compact "union bound" over all α-approximate cuts. This structure is built from an *expander hierarchy* of the graph, which is a hierarchical partitioning of the graph into disjoint expanders introduced by Goranci et al. [6]. The connection between expanders and the mincut problem has been observed before [12, 13]: in an expander with conductance ϕ, all α-approximate mincuts must have at most α/ϕ vertices on one side, so a compact representation is simply all cuts with at most α/ϕ vertices on one side. Motivated by this connection, we show that if the original graph is itself an expander, then it is enough to preserve all vertex degrees and all edge weights up to an additive $\epsilon'\lambda$ factor, where λ is the mincut of the graph and ϵ' depends on ϵ, α, ϕ. We present the unweighted expander case in Section 2 as a warm-up, which features all of our ideas except for the final expander decomposition step. To handle general graphs, we exploit the full machinery of the expander hierarchy [6].

Large cuts. For the large cuts—those that are not α-approximate mincuts—our strategy differs from the pessimistic estimator approach. Here, our aim is not to preserve each of them up to a $(1+\epsilon)$-factor, but a γ-factor for a different parameter $\gamma = n^{o(1)}$. This relaxation prevents us from obtaining a full $(1+\epsilon)$-approximate sparsification of the graph, but it still works for the mincut problem

[1] In this paper, $\widetilde{O}(\cdot)$ notation hides polylogarithmic factors in n, the number of vertices of the graph.

since the large cuts do not fall below the original mincut value. While a deterministic $(1+\epsilon)$-approximate sparsification algorithm in near-linear time is unknown, one exists for γ-approximation sparsification for some $\gamma = n^{o(1)}$ [1]. In our case, we actually need the sparsifier to be *uniformly weighted*, so we construct our own sparsifier in Section 3.2.2, again via the expander hierarchy. Note that if the original graph is an expander, then we can take any expander whose degrees are roughly the same; in particular, the sparsifier does not need to be a subgraph of the original graph. To summarize, for the large cuts case, we simply construct an γ-approximate sparsifier deterministically, bypassing the need to de-randomize the Benczur-Karger random sampling technique.

Combining them together. Of course, this γ-approximate sparsifier destroys the guarantee of the small cuts, which need to be preserved $(1+\epsilon)$-approximately. Our strategy is to combine the small cut sparsifier and the large cut sparsifier together in the following way. We take the union of the small cut sparsifier with a "lightly" weighted version of the large cut sparsifier, where each edge in it is weighted by ϵ/γ times its normal weight. This way, each small cut of weight w suffers at most an additive $\gamma w \cdot \epsilon/\gamma = \epsilon w$ weight from the "light" large cut sparsifier, so we do not destroy the small cuts guarantee (up to replacing ϵ with 2ϵ). Moreover, each large cut of weight $w \geq \alpha\lambda$ is weighted by at least $w/\gamma \cdot \epsilon/\gamma \geq \alpha\lambda/\gamma \cdot \epsilon/\gamma = \alpha/\gamma^2 \cdot \epsilon\lambda$, where λ is the mincut of the original graph. Hence, as long as $\alpha \geq \gamma^2/\epsilon$, the large cuts have weight at least the mincut, and the property for large cuts is preserved.

Unbalanced vs. balanced. We remark that our actual separation between small cuts and large cuts is somewhat different; we use *unbalanced* and *balanced* instead to emphasize this distinction. Nevertheless, we should intuitively think of unbalanced cuts as having small weight and balanced as having large weight; rather, the line is not drawn precisely at a weight threshold of $\alpha\lambda$. The actual separation is more technical, so we omit it in this overview section.

1.2 Preliminaries

In this paper, all graphs are undirected, and n and m denote the number of vertices and edges of the input graph in question. All graphs are either unweighted or weighted multigraphs with polynomially bounded edge weights, i.e., in the range $[\frac{1}{\text{poly}(n)}, \text{poly}(n)]$. **We emphasize that even *weighted* graphs are multigraphs, which we find more convenient to work with.**

We begin with more standard notation. For an unweighted graph $G = (V, E)$ and vertices $u, v \in V$, let $\#(u, v)$ be the number of edges $e \in E$ with endpoints u and v. For a weighted graph $G = (V, E)$ and edge $e \in E$, let $w(e)$ be the weight of the edge, and for vertices $u, v \in V$, let $w(u, v)$ be the *sum* of the weights $w(e)$ of all (parallel) edges e between u and v. For disjoint sets of vertices $S, T \subseteq V$, define $E(S, T) \subseteq E$ as the set of edges with one endpoint in S and the other in T, and define $\partial S := E(S, V \setminus S)$. For a set $F \subseteq E$ of edges, denote its cardinality by $|F|$ if G is unweighted, and its total weight by $w(F)$ if G is weighted. Define the *degree* $\deg(v)$ of vertex $v \in V$ to be $|\partial(\{v\})|$ if G is unweighted, and $w(\partial(\{v\}))$ if G is weighted. For a set $S \subseteq V$, define $\mathbf{vol}(S) := \sum_{v \in S} \deg(v)$. A *cut* of G is the set of edges ∂S for some $\emptyset \subsetneq S \subsetneq V$, and the *mincut* of G is the cut ∂S in G that minimizes $|\partial S|$ or $w(\partial S)$ depending on if G is unweighted or weighted. When the graph G is ambiguous, we may add a subscript of G in our notation, such as $\#_G(u, v)$.

1.2.1 Karger's Approach.

In this section, we outline Karger's approach to his near-linear time randomized mincut algorithm and set up the necessary theorems for our deterministic result. Karger's algorithm has two main steps. First, it computes a small set of (unweighted) trees on vertex set V such that the mincut 2-*respects* one of the trees T, defined as follows:

Definition 1.2. *Given a weighted graph G and an unweighted tree T on the same set of vertices, a cut $\partial_G S$* 2-respects *the tree T if $|\partial_T S| \leq 2$.*

Karger accomplishes this goal by first *sparsifying* the graph into an unweighted *skeleton* graph using the well-known Benzcur-Karger sparsification by random sampling, and then running a *tree packing* algorithm of Gabow [2] on the skeleton graph.

Theorem 1.3 (Karger [10]). *Let G be a weighted graph, let m' and c' be parameters, and let H be an unweighted graph on the same vertices, called the* skeleton *graph, with the following properties:*

(a) *H has m' edges,*

(b) *The mincut of H is c', and*

(c) *The mincut in G corresponds (under the same vertex partition) to a $7/6$-approximate mincut in H.*

Given graphs G and H, there is a deterministic algorithm in $O(c'm' \log n)$ time that constructs $O(c')$ trees on the same vertices such that one of them 2-respects the mincut in G.

The second main step of Karger's algorithm is to compute the mincut of G given a tree that 2-respects the mincut. This step is deterministic and is based on dynamic programming.

Theorem 1.4 (Karger [10]). *Given a weighted, undirected graph G and a (not necessarily spanning) tree T on the same vertices, there is a deterministic algorithm in $O(m \log^2 n)$ time that computes the minimum-weight cut in G that 2-respects the tree T.*

Our main technical contribution is a deterministic construction of the skeleton graph used in Theorem 1.3. Instead of designing an algorithm to produce the skeleton graph directly, it is more convenient to prove the following, which implies a skeleton graph by the following claim.

Theorem 1.5. *For any $0 < \epsilon \leq 1$, we can compute, in deterministic $\epsilon^{-4} 2^{O(\log n)^{5/6} (\log\log n)^{O(1)}} m$ time, an unweighted graph H and some weight $W = \epsilon^4 \lambda / 2^{O(\log n)^{5/6} (\log\log n)^{O(1)}}$ such that*

(1) *For any mincut ∂S^* of G, we have $W \cdot |\partial_H S^*| \leq (1+\epsilon)\lambda$, and*

(2) *For any cut $\emptyset \subsetneq S \subsetneq V$ of G, we have $W \cdot |\partial_H S| \geq (1-\epsilon)\lambda$.*

Claim 1.6. *For $\epsilon = 0.01$, the graph H in Theorem 1.5 fulfills the conditions of Theorem 1.3 with $m' = m^{1+o(1)}$ and $c' = n^{o(1)}$.*

Proof. Since the algorithm of Theorem 1.5 takes $m^{1+o(1)}$ time, the output graph H must have $m^{1+o(1)}$ edges, fulfilling condition (a) of Theorem 1.3. For any mincut S^* of G, by property (1) of Theorem 1.5, we have $|\partial_H S^*| \le (1+\epsilon)\lambda/W \le n^{o(1)}$, fulfilling condition (b). For any cut $\emptyset \subsetneq S \subsetneq V$, by property (2), we have $|\partial_H S| \ge (1-\epsilon)\lambda/W$. In other words, S^* is a $(1+\epsilon)/(1-\epsilon)$-approximate mincut, which is a $7/6$-approximate mincut for $\epsilon = 0.01$, fulfilling condition (c). □

With the above three statements in hand, we now prove Theorem 1.1 following Karger's approach. Run the algorithm of Theorem 1.5 to produce a graph H which, by Claim 1.6, satisfies the conditions of Theorem 1.3. Apply Theorem 1.3 on G and the skeleton graph H, producing $n^{o(1)}$ many trees such that one of them 2-respects the mincut in G. Finally, run Theorem 1.4 on each tree separately and output the minimum 2-respecting cut found among all the trees, which must be the mincut in G. Each step requires $2^{O(\log n)^{5/6}(\log\log n)^{O(1)}} m$ deterministic time, proving Theorem 1.1.

Thus, the main focus for the rest of the paper is proving Theorem 1.5.

1.2.2 Spectral Graph Theory. Central to our approach are the well-known concepts of *conductance*, *expanders*, and the graph *Laplacian* from spectral graph theory.

Definition 1.7 (Conductance, expander)**.** *The* conductance *of a weighted graph G is*

$$\Phi(G) := \min_{\emptyset \subsetneq S \subsetneq V} \frac{w(E(S, V\setminus S))}{\min\{\mathbf{vol}(S), \mathbf{vol}(V\setminus S)\}}.$$

*For the conductance of an unweighted graph, replace $w(E(S, V\setminus S))$ by $|E(S, V\setminus S)|$. We say that G is a ϕ-*expander *if $\Phi(G) \ge \phi$.*

Definition 1.8 (Laplacian)**.** *The* Laplacian *L_G of a weighted graph $G = (V, E)$ is the $n \times n$ matrix, indexed by $V \times V$, where*

(a) *Each diagonal entry (v, v) has entry $\deg(v)$, and*

(b) *Each off-diagonal entry (u, v) $(u \ne v)$ has weight $-w(u, v)$ if $(u, v) \in E$ and 0 otherwise.*

The only fact we will use about Laplacians is the following well-known fact, that cuts in graphs have the following nice form:

Fact 1.9. *For any weighted graph $G = (V, E)$ with Laplacian L_G, and for any subset $S \subseteq V$, we have*

$$w(\partial S) = \mathbf{1}_S^T L_G \mathbf{1}_S,$$

where $\mathbf{1}_S \in \{0,1\}^V$ is the vector with value 1 at vertex v if $v \in S$, and value 0 otherwise. For unweighted graph G, replace $w(\partial S)$ with $|\partial S|$.

2 EXPANDER CASE

In this section, we prove Theorem 1.5 restricted to the case when G is an *unweighted expander*. Our aim is to present an informal, intuitive exposition that highlights our main ideas in a relatively simple setting. Since this section is not technically required for the main result, we do not attempt to formalize our arguments, deferring the rigorous proofs to the general case in Section 3.

Theorem 2.1. *Let G be an unweighted ϕ-expander multigraph. For any $0 < \epsilon \le 1$, we can compute, in deterministic $m^{1+o(1)}$ time, an unweighted graph H and some weight $W = \epsilon^3\lambda/n^{o(1)}$ such that*

(a) *For any mincut $\partial_G S^*$ of G, we have $W \cdot |\partial_H S^*| \le (1+\epsilon)\lambda$, and*

(b) *For any cut $\partial_G S$ of G, we have $W \cdot |\partial_H S| \ge (1-\epsilon)\lambda$.*

For the rest of this section, we prove Theorem 2.1.

Consider an arbitrary cut $\partial_G S$. By Fact 1.9, we have

$$|\partial_G S| = \mathbf{1}_S^T L_G \mathbf{1}_S = \left(\sum_{v\in S} \mathbf{1}_v^T\right) L_G \left(\sum_{v\in S} \mathbf{1}_v\right) = \sum_{u,v\in S} \mathbf{1}_u^T L_G \mathbf{1}_v. \quad (1)$$

Suppose we can approximate each $\mathbf{1}_u^T L_G \mathbf{1}_v$ to an additive error of $\epsilon'\lambda$ for some small ϵ' (depending on ϵ); that is, suppose that our graph H and weight W satisfy

$$|\mathbf{1}_u^T L_G \mathbf{1}_v - W \cdot \mathbf{1}_u^T L_H \mathbf{1}_v| \le \epsilon'\lambda$$

for all $u, v \in V$. Then, by (1), we can approximate $|\partial_G S|$ up to an additive $|S|^2\epsilon'\lambda$, or a multiplicative $(1 + |S|^2\epsilon')$, which is good if $|S|$ is small. Similarly, if $|V \setminus S|$ is small, then we can replace S with $V \setminus S$ in (1) and approximate $|\partial_G S| = |\partial_G(V \setminus S)|$ to the same factor. Motivated by this observation, we define a set $S \subseteq V$ to be *unbalanced* if $\min\{\mathbf{vol}(S), \mathbf{vol}(V \setminus S)\} \le \alpha\lambda/\phi$ for some $\alpha = n^{o(1)}$ to be set later. Similarly, define a cut $\partial_G S$ to be unbalanced if the set S is unbalanced. Note that an unbalanced set S must have either $|S| \le \alpha/\phi$ or $|V \setminus S| \le \alpha/\phi$, since if we assume without loss of generality that $\mathbf{vol}(S) \le \mathbf{vol}(V \setminus S)$, then

$$|S|\lambda \le \sum_{v\in S} \deg(v) = \mathbf{vol}(S) \le \alpha\lambda/\phi, \quad (2)$$

where the first inequality uses that each degree cut $\partial(\{v\})$ has weight $\deg(v) \ge \lambda$. Moreover, since G is a ϕ-expander, the mincut $\partial_G S^*$ is unbalanced because, assuming without loss of generality that $\mathbf{vol}(S^*) \le \mathbf{vol}(V \setminus S^*)$, we obtain

$$\frac{|\partial_G(S^*)|}{\mathbf{vol}(S^*)} \ge \Phi(G) \ge \phi \implies \mathbf{vol}(S^*) \le 1/\phi \le \alpha\lambda/\phi.$$

To approximate all unbalanced cuts, it suffices by (1) and (2) to approximate each $\mathbf{1}_u^T L_G \mathbf{1}_v$ up to additive error $(\phi/\alpha)^2\epsilon\lambda$. When $u \ne v$, the expression $\mathbf{1}_u^T L_G \mathbf{1}_v$ is simply the negative of the number of parallel (u, v) edges in G. So, approximating $\mathbf{1}_u^T L_G \mathbf{1}_v$ up to additive error $\epsilon\lambda$ simply amounts to approximating the number of parallel (u, v) edges. When $u = v$, the expression $\mathbf{1}_v^T L_G \mathbf{1}_v$ is simply the degree of v, so approximating it amounts to approximating the degree of v.

Consider what happens if we randomly sample each edge with probability $p = \Theta(\frac{\alpha \log n}{\epsilon^2\phi\lambda})$ and weight the sampled edges by $\widehat{W} := 1/p$ to form the sampled graph $\widehat{H}$. For the terms $\mathbf{1}_u^T L_G \mathbf{1}_v$ $(u \ne v)$, we have $\#_G(u, v) \le \mathbf{vol}(S) \le \alpha\lambda/\phi$. Let us assume for simplicity that $\#_G(u, v) = \alpha\lambda/\phi$, which turns out to be the worst case. By

Chernoff bounds, for $\delta = \epsilon\phi/\alpha$,

$$\Pr\left[\left|\#_{\widehat{H}}(u,v) - p\cdot\#_G(u,v)\right| > \delta\cdot p\cdot\#_G(u,v)\right] \tag{3}$$
$$< 2\exp(-\delta^2\cdot p\cdot\#_G(u,v)/3)$$
$$= 2\exp\left(-\left(\frac{\epsilon\phi}{\alpha}\right)^2\cdot\Theta\left(\frac{\alpha\log n}{\epsilon^2\phi\lambda}\right)\cdot\frac{\alpha\lambda/\phi}{3}\right) \tag{4}$$
$$= 2\exp(-\Theta(\log n)),$$

which we can set to be much less than $1/n^2$. We then have the implication

$$\left|\#_{\widehat{H}}(u,v) - p\cdot\#_G(u,v)\right| \le \delta\cdot p\cdot\#_G(u,v)$$
$$\implies \left|\mathbf{1}_u^T(L_G - L_{\widehat{H}})\mathbf{1}_v\right| \le \delta\cdot\#_G(u,v) = \epsilon\phi/\alpha\cdot\alpha\lambda/\phi = \epsilon\lambda.$$

Similarly, for the terms $\mathbf{1}_v^T L_G\mathbf{1}_v$, we have $\deg(v) \le \mathbf{vol}(S) \le \alpha\lambda/\phi$, and the same calculation can be made.

From this random sampling analysis, we can derive the following pessimistic estimator. Initially, it is the sum of the quantities (4) for all (u,v) satisfying either $u = v$ or $(u,v)\in E$. This sum has $O(m)$ terms which sum to less than 1, so it can be efficiently computed and satisfies the initial condition of a pessimistic estimator. After some edges have been considered, the probability upper bounds (4) are modified to be conditional to the choices of edges so far, which can still be efficiently computed. At the end, for each unbalanced set S, the graph $\widehat{H}$ will satisfy

$$\left||\partial_G S| - \widehat{W}\cdot|\partial_{\widehat{H}}S|\right| \le \epsilon\lambda \implies (1-\epsilon)|\partial_G S| \le \widehat{W}\cdot|\partial_{\widehat{H}}S| \le (1+\epsilon)|\partial_G S|.$$

Since any mincut $\partial_G S^*$ is unbalanced, we fulfill condition (a) of Theorem 2.1. We also fulfill condition (b) for any cut with a side that is unbalanced. This concludes the unbalanced case; we omit the rest of the details, deferring the pessimistic estimator and its efficient computation to the general case, specifically Section 3.2.1.

Define a cut to be *balanced* if it is not unbalanced. For the balanced cuts, it remains to fulfill condition (b), which may not hold for the graph $\widehat{H}$. Our solution is to "overlay" a fixed expander onto the graph $\widehat{H}$, weighted small enough to barely affect the mincut (in order to preserve condition (a)), but large enough to force all balanced cuts to have weight at least λ. In particular, let $\widetilde{H}$ be an unweighted $\Theta(1)$-expander on the same vertex set V where each vertex $v\in V$ has degree $\Theta(\deg_G(v)/\lambda)$, and let $\widetilde{W} := \Theta(\epsilon\phi\lambda)$. We should think of $\widetilde{H}$ as a "lossy" sparsifier of G, in that it approximates cuts up to factor $O(1/\phi)$, not $(1+\epsilon)$.

Consider taking the "union" of the graph $\widehat{H}$ weighted by $\widehat{W}$ and the graph $\widetilde{H}$ weighted by $\widetilde{W}$. More formally, consider a weighted graph H' where each edge (u,v) is weighted by $\widehat{W}\cdot w_{\widehat{H}}(u,v) + \widetilde{W}\cdot w_{\widetilde{H}}(u,v)$. We now show two properties: (1) the mincut gains relatively little weight from $\widetilde{H}$ in the union H', and (2) any balanced cut automatically has at least λ total weight from $\widetilde{H}$.

(1) For a mincut $\partial_G S^*$ in G with $\mathbf{vol}_G(S^*) \le |\partial_G S^*|/\phi = \lambda/\phi$, the cut crosses

$$w(\partial_{\widetilde{H}}S^*) \le \mathbf{vol}_{\widetilde{H}}(S^*) \le \Theta(1)\cdot\mathbf{vol}_G(S^*)/\lambda \le \Theta(1/\phi)$$

edges in $\widetilde{H}$, for a total cost of at most $\Theta(1/\phi)\cdot\Theta(\epsilon\phi\lambda) \le \epsilon\lambda$.

(2) For a balanced cut $\partial_G S$, it satisfies $|\partial_G S| \ge \phi\cdot\mathbf{vol}_G(S) \ge \alpha\lambda$, so it crosses

$$w(\partial_{\widehat{H}}S) \ge \Theta(1)\cdot\mathbf{vol}_{\widehat{H}}(S) \ge \Theta(1)\cdot\mathbf{vol}_G(S)/\lambda \ge \Theta(\alpha/\phi)$$

many edges in $\widetilde{H}$, for a total cost of at least $\Theta(\alpha/\phi)\cdot\Theta(\epsilon\phi\lambda)$. Setting $\alpha := \Theta(\frac{1}{\epsilon})$, the cost becomes at least λ.

Therefore, in the weighted graph H', the mincut has weight at most $(1+O(\epsilon))\lambda$, and any cut has weight at least $(1-\epsilon)\lambda$. We can reset ϵ to be a constant factor smaller so that the factor $(1+O(\epsilon))$ becomes $(1+\epsilon)$.

To finish the proof of Theorem 2.1, it remains to extract an unweighted graph H and a weight W from the weighted graph H'. Since $\widehat{W} = \Theta(\frac{\epsilon^2\phi\lambda}{\alpha\log n}) = \Theta(\frac{\epsilon^3\phi\lambda}{\log n})$ and $\widetilde{W} = \Theta(\epsilon\phi\lambda)$, we can make $\widetilde{W}$ an integer multiple of $\widehat{W}$, so that each edge in H' is an integer multiple of $\widehat{W}$. We can therefore set $W := \widehat{W}$ and define the unweighted graph H so that $\#_H(u,v) = w_{H'}(u,v)/\widehat{W}$ for all $u,v\in V$.

3 GENERAL CASE

This section is dedicated to proving Theorem 1.5. For simplicity, we instead prove the following restricted version first, which has the additional assumption that the maximum edge weight in G is bounded. At the end of this section, we show why this assumption can be removed to obtain the full Theorem 1.5.

Theorem 3.1. *There exists a function $f(n) \le 2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}$ such that the following holds. Let G be a graph with mincut λ* ***and maximum edge weight at most*** *$\epsilon^4\lambda/f(n)$. For any $0 < \epsilon \le 1$, we can compute, in deterministic $2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}m$ time, an unweighted graph H and some weight $W \ge \epsilon^4\lambda/f(n)$ such that the two properties of Theorem 1.5 hold, i.e.,*

(1) *For any mincut S^* of G, we have $W\cdot|\partial_H S^*| \le (1+\epsilon)\lambda$, and*

(2) *For any cut $\emptyset \subsetneq S \subsetneq V$ of G, we have $W\cdot|\partial_H S| \ge (1-\epsilon)\lambda$.*

3.1 Expander Decomposition Preliminaries

Our main tool in generalizing the expander case is *expander decompositions*, which was popularized by Spielman and Teng [19] and is quickly gaining traction in the area of fast graph algorithms. The general approach to utilizing expander decompositions is as follows. First, solve the case when the input graph is an expander, which we have done in Section 2 for the problem described in Theorem 1.5. Then, for a general graph, *decompose* it into a collection of expanders with few edges between the expanders, solve the problem each expander separately, and combine the solutions together, which often involves a recursive call on a graph that is a constant-factor smaller. For our purposes, we use a slightly stronger variant than the usual expander decomposition that ensures *boundary-linkedness*, which will be important in our analysis. The following definition is inspired by [6]; note that our variant is weaker than the one in Definition 4.2 of [6] in that we only guarantee their property (2). In the full version of this paper, we include a full proof that is similar to the one in [6], and assuming a subroutine called WeightedBalCutPrune from [1].

Theorem 3.2 (Boundary-linked expander decomposition). *Let $G = (V, E)$ be a graph and let $r \geq 1$ be a parameter. There is a deterministic algorithm in $m^{1+O(1/r)} + \widetilde{O}(m/\phi^2)$ time that, for any parameters $\beta \leq (\log n)^{-O(r^4)}$ and $\phi \leq \beta$, partitions $V = V_1 \uplus \cdots \uplus V_k$ such that*

(1) *Each vertex set V_i satisfies*

$$\min_{\emptyset \subsetneq S \subsetneq V_i} \frac{w(\partial_{G[V_i]} S)}{\min \left\{ \begin{array}{l} \mathbf{vol}_{G[V_i]}(S) + \frac{\beta}{\phi} w(E_G(S, V \setminus V_i)), \\ \mathbf{vol}_{G[V_i]}(V_i \setminus S) + \frac{\beta}{\phi} w(E_G(V_i \setminus S, V \setminus V_i)) \end{array} \right\}} \geq \phi. \tag{5}$$

*Informally, we call the graph $G[V_i]$ together with its boundary edges $E_G(V_i, V \setminus V_i)$ a β-*boundary-linked *ϕ-expander.*[2] *In particular, for any S satisfying*

$$\mathbf{vol}_{G[V_i]}(S) + \frac{\beta}{\phi} w(E_G(S, V \setminus V_i)) \leq \mathbf{vol}_{G[V_i]}(V_i \setminus S) + \frac{\beta}{\phi} w(E_G(V_i \setminus S, V \setminus V_i)),$$

we simultaneously obtain

$$\frac{w(\partial_{G[V_i]} S)}{\mathbf{vol}_{G[V_i]}(S)} \geq \phi$$

and

$$\frac{w(\partial_{G[V_i]} S)}{\frac{\beta}{\phi} w(E_G(S, V \setminus V_i))} \geq \phi \iff \frac{w(\partial_{G[V_i]} S)}{w(E_G(S, V \setminus V_i))} \geq \beta.$$

The right-most inequality is where the name "boundary-linked" comes from.

(2) *The total weight of "inter-cluster" edges, $w(\partial V_1 \cup \cdots \cup \partial V_k)$, is at most $(\log n)^{O(r^4)} \phi \mathbf{vol}(V)$.*

Note that for our applications, it's important that the boundary-linked parameter β is much larger than ϕ. This is because in our recursive algorithm, the approximation factor will blow up by roughly $1/\beta$ per recursion level, while the instance size shrinks by roughly ϕ.

In order to capture recursion via expander decompositions, we now define a *boundary-linked expander decomposition sequence* $\{G^i\}$ on the graph G in a similar way to [6]. Compute a boundary-linked expander decomposition for β and $\phi \leq \beta$ to be determined later, contract each expander,[3] and recursively decompose the contracted graph until the graph consists of a single vertex. Let $G^0 = G$ be the original graph and $G^1, G^2, \ldots, G^L$ be the recursive contracted graphs. Note that each graph G^i has minimum degree at least λ, since any degree cut in any G^i induces a cut in the original graph G. Each time we contract, we will keep edge identities for the edges that survive, so that $E(G^0) \supseteq E(G^1) \supseteq \cdots \supseteq E(G^L)$. Let U^i be the vertices of G^i.

[2] For unweighted graphs, [6] uses the notation $G[V_i]^{\beta/\phi}$ to represent a graph where each (boundary) edge in $E(V_i, V \setminus V_i)$ is replaced with β/ϕ many self-loops at the endpoint in V_i. With this definition, (5) is equivalent to saying that $G[V_i]^{\beta/\phi}$ is a ϕ-expander.

[3] Since we are working with weighted multigraphs, we do *not* collapse parallel edges obtained from contraction into single edges.

For the rest of Section 3.1, fix an expander decomposition sequence $\{G^i\}$ of G. For any subset $\emptyset \subsetneq S \subsetneq V$, we now define an *decomposition sequence* of S as follows. Let $S^0 = S$, and for each $i > 0$, construct S^{i+1} as a subset of the vertices of G^{i+1}, as follows. Take the expander decomposition of G^i, which partitions the vertices U^i of G^i into, say, $U^i_1, \ldots, U^i_{k_i}$. Each of the U^i_j gets contracted to a single vertex u_j in G^i. For each U^i_j, we have a choice whether to add u_j to S^i or not. This completes the construction of S^i. Define the "difference" $D^i_j = U_j \setminus S^i$ if $u_j \in S^i$, and $D^i_j = U_j \cap S^i$ otherwise. The sets S^i, U^i_j, and D^i_j define the decomposition sequence of S.

We now prove some key properties of the boundary-linked expander decomposition sequence in the context of graph cuts, which we will use later on. First, regardless of the choice whether to add each u_j to S^i, we have the following lemma relating the sets D^i_j to the original set S.

Lemma 3.3. *For any decomposition sequence $\{S^i\}$ of S,*

$$\partial_G S \subseteq \bigcup_{i=0}^{L} \bigcup_{j \in [k_i]} \partial_{G^i} D^i_j.$$

Proof. Observe that

$$(\partial_{G^i} S^i) \triangle (\partial_{G^i} S^{i+1}) \subseteq \bigcup_{j \in [k_i]} \partial_{G^i} D^i_j. \tag{6}$$

In particular,

$$\partial_{G^i} S^i \subseteq \partial_{G^i} S^{i+1} \cup \bigcup_{j \in [k_i]} \partial_{G^i} D^i_j.$$

Iterating this over all i,

$$\partial_G S \subseteq \bigcup_{i=0}^{L} \bigcup_{j \in [k_i]} \partial_{G^i} D^i_j.$$

□

We now define a specific decomposition sequence of S, by setting up the rule whether or not to include each u_j in S^i. For each U^i_j, if

$$\mathbf{vol}_{G^i[U^i_j]}(S^i \cap U^i_j) + \frac{\beta}{\phi} w(E_{G^i}(S^i \cap U^i_j, U^i \setminus U^i_j)) \geq \mathbf{vol}_{G^i[U^i_j]}(U^i_j \setminus S^i) + \frac{\beta}{\phi} w(E_{G^i}(U^i_j \setminus S^i, U^i \setminus U^i_j)),$$

then add u_j to S^i; otherwise, do not add u_j to S^i. This ensures that

$$\mathbf{vol}_{G^i[U^i_j]}(U^i_j \setminus D^i_j) + \frac{\beta}{\phi} w(E_{G^i}(U^i_j \setminus D^i_j, U^i \setminus U^i_j)) \geq \mathbf{vol}_{G^i[U^i_j]}(D^i_j) + \frac{\beta}{\phi} w(E_{G^i}(D^i_j, U^i \setminus U^i_j)). \tag{7}$$

Since $G^i[U^i_j]$ is a β-boundary-linked ϕ-expander, by our construction, we have, for all i, j,

$$\frac{w(\partial_{G^i[U^i_j]} D^i_j)}{\mathbf{vol}_{G^i[U^i_j]}(D^i_j)} \geq \phi \tag{8}$$

and

$$\frac{w(\partial_{G^i}[U_j^i]D_j^i)}{w(E_{G^i}(D_j^i, U^i \setminus U_j^i))} \geq \beta. \tag{9}$$

For this specific construction of $\{S^i\}$, called the *canonical* decomposition sequence of S, we have the following lemma, which complements Lemma 3.3.

Lemma 3.4. *Let $\{S^i\}$ be any decomposition sequence of S satisfying (9) for all i, j. Then,*

$$\sum_{i=0}^{L} \sum_{j\in[k_i]} w(\partial_{G^i} D_j^i) \leq \beta^{-O(L)} w(\partial_G S).$$

Proof. By (9),

$$w(E_{G^i}(D_j^i, U^i \setminus U_j^i)) \leq \frac{1}{\beta} \cdot w(\partial_{G^i}[U_j^i]D_j^i).$$

The edges of $\partial_{G^i}[U_j^i]D_j^i$ are inside $\partial_{G^i} S^i$ and are disjoint over distinct j, so in total,

$$\sum_{j\in[k_i]} w(\partial_{G^i} D_j^i) \leq \sum_{j\in[k_i]} \frac{1}{\beta} \cdot w(\partial_{G^i}[U_j^i]D_j^i) \leq \frac{1}{\beta} \cdot w(\partial_{G^i} S^i).$$

From (6), we also obtain

$$\partial_{G^{i+1}} S^{i+1} \subseteq \partial_{G^i} S^i \cup \bigcup_{j\in[k_i]} \partial_{G^i} D_j^i.$$

Therefore,

$$w(\partial_{G^{i+1}} S^{i+1}) \leq w(\partial_{G^i} S^i) + w\left(\bigcup_{j\in[k_i]} \partial_{G^i} D_j^i\right) \leq \left(1 + \frac{1}{\beta}\right) \cdot w(\partial_{G^i} S^i).$$

Iterating this over all $i \in [L]$, we obtain

$$w(\partial_{G^i} S^i) \leq \left(1 + \frac{1}{\beta}\right)^i \cdot w(\partial_G S).$$

Thus,

$$\begin{aligned}\sum_{i=0}^{L} \sum_{j\in[k_i]} w(\partial_{G^i} D_j^i) &\leq \sum_{i=0}^{L} \frac{1}{\beta} \cdot w(\partial_{G^i} S^i) \\ &\leq \sum_{i=0}^{L} \frac{1}{\beta} \cdot \left(1 + \frac{1}{\beta}\right)^i \cdot w(\partial_G S) \\ &= \beta^{-O(L)} w(\partial_G S).\end{aligned}$$

□

3.2 Unbalanced Case

In this section, we generalize the notion of *unbalanced* from Section 2 to the general case, and then prove a $(1+\epsilon)$-approximate sparsifier of the unbalanced cuts.

Fix an expander decomposition sequence $\{G^i\}$ of G for the Section 3.2. For a given set $\emptyset \subsetneq S \subsetneq V$, let $\{S^i\}$ be the canonical decomposition sequence of S, and define D_j^i as before, so that they satisfy (8) and (9) for all i, j. We generalize our definition of *unbalanced* from the expander case as follows, for some $\tau = n^{o(1)}$ to be specified later.

Definition 3.5. *The set $S \subseteq V$ is* τ-unbalanced *if for each level i, $\sum_{j\in[k_i]} \mathbf{vol}_{G^i}(D_j^i) \leq \tau\lambda/\phi$. A cut ∂S is τ-unbalanced if the set S is τ-unbalanced.*

Note that if G is originally an expander, then in the first expander decomposition of the sequence, we can declare the entire graph as a single expander; in this case, the expander decomposition sequence stops immediately, and the definition of τ-unbalanced becomes equivalent to that from the expander case. We now claim that for an appropriate value of τ, any mincut is τ-unbalanced.

Claim 3.6. *For $\tau \geq \beta^{-\Omega(L)}$, any mincut ∂S^* of G is τ-unbalanced.*

Proof. Consider the canonical decomposition sequence of S, and define D_j^i as usual. For each level i and index $j \in [k_i]$,

$$\begin{aligned}\mathbf{vol}_{G^i}(D_j^i) &= \mathbf{vol}_{G^i[U_j^i]}(D_j^i) + w(E_{G^i}(D_j^i, U^i \setminus U_j^i)) \\ &\overset{(8)}{\leq} \frac{1}{\phi} w(\partial_{G^i[U_j^i]} D_j^i) + w(E_{G^i}(D_j^i, U^i \setminus U_j^i)) \\ &\leq \frac{1}{\phi} w(\partial_{G^i} D_j^i).\end{aligned}$$

Summing over all $j \in [k_i]$ and applying Lemma 3.4,

$$\begin{aligned}\sum_{j\in[k_i]} \mathbf{vol}_{G^i}(D_j^i) &\leq \sum_{j\in[k_i]} \frac{1}{\phi} w(\partial_{G^i} D_j^i) \\ &= \frac{1}{\phi} \cdot \sum_{j\in[k_i]} w(\partial_{G^i} D_j^i) \\ &\overset{\text{Lem.3.4}}{\leq} \frac{1}{\phi} \cdot \beta^{-O(L)} w(\partial_G S^*) \leq \frac{\tau\lambda}{\phi},\end{aligned}$$

so S^* is τ-unbalanced. □

Let us now introduce some notation exclusive to this section. For each vertex $v \in U^i$, let $\overline{v} \subseteq V$ be its "pullback" on the original set V, defined as all vertices in V that get contracted into v in graph G^i in the expander sequence. For each set D_j^i, let $\overline{D_j^i} \subseteq V$ be the pullback of D_j^i, defined as $\overline{D_j^i} = \bigcup_{v\in D_j^i} \overline{v}$. We can then write

$$\mathbf{1}_S = \sum_{i,j} \pm \mathbf{1}_{\overline{D_j^i}} = \sum_{i,j} \sum_{v\in D_j^i} \pm \mathbf{1}_{\overline{v}},$$

where the $\pm$ sign depends on whether $D_j^i = U_j^i \setminus S^i$ or $D_j^i = U_j^i \cap S^i$. Then,

$$\begin{aligned}w(\partial_G S) = \mathbf{1}_S^T L_G \mathbf{1}_S &= \sum_{i,j,k,l} \pm \mathbf{1}_{\overline{D_j^i}}^T L_G \mathbf{1}_{\overline{D_l^k}} \\ &= \sum_{i,j,k,l} \sum_{u\in D_j^i, v\in D_l^k} \pm \mathbf{1}_u^T L_G \mathbf{1}_{\overline{v}}.\end{aligned} \tag{10}$$

Claim 3.7. *For an τ-unbalanced set S, there are at most $((L+1)\tau/\phi)^2$ nonzero terms in the summation (10).*

Proof. Each vertex $v \in D^i_j$ has degree at least λ in G^i, since it induces a cut (specifically, its pullback $\overline{v} \subseteq V$) in the original graph G. Therefore,

$$\tau\lambda/\phi \geq \sum_{j\in[k_i]} \mathbf{vol}_{G^i}(D^i_j) \geq \sum_{j\in[k_i]} |D^i_j| \cdot \lambda,$$

so there are at most τ/ϕ many choices for j and $u \in D^i_j$ given a level i. There are at most $L+1$ many choices for i, giving at most $(L+1)\tau/\phi$ many combinations of i, j, u. The same holds for combinations of k, l, v, hence the claim. □

The main goal of this section is to prove the following lemma.

Lemma 3.8. *There exists a constant $C > 0$ such that given any weight $W \leq \frac{C\epsilon\phi\lambda}{\tau \ln(Lm)}$, we can compute, in deterministic $\widetilde{O}(L^2 m)$ time,[4] an unweighted graph H such that for all levels i, k and vertices $u \in U^i, v \in U^k$ satisfying $\deg_{G^i}(u) \leq \tau\lambda/\phi$ and $\deg_{G^k}(v) \leq \tau\lambda/\phi$,*

$$\left|\mathbf{1}_{\overline{u}}^T L_G \mathbf{1}_{\overline{v}} - W \cdot \mathbf{1}_{\overline{u}}^T L_H \mathbf{1}_{\overline{v}}\right| \leq \epsilon\lambda. \tag{11}$$

Before we prove Lemma 3.8, we show that it implies a sparsifier of τ-unbalanced cuts, which is the lemma we will eventually use to prove Theorem 3.1:

Lemma 3.9. *There exists a constant $C > 0$ such that given any weight $W \leq \frac{C\epsilon\phi\lambda}{\tau \ln(Lm)}$, we can compute, in deterministic $\widetilde{O}(L^2 m)$ time, an unweighted graph H such that for each τ-*unbalanced *cut S,*

$$\left|w(\partial_G S) - W \cdot w(\partial_H S)\right| \leq \left(\frac{(L+1)\tau}{\phi}\right)^2 \cdot \epsilon\lambda.$$

Proof. Let $C > 0$ be the same constant as the one in Lemma 3.8. Applying (10) to $\partial_H S$ as well, we have

$$w(\partial_G S) - W \cdot w(\partial_H S) = \sum_{i,j,k,l} \sum_{u\in D^i_j, v\in D^k_l} \pm(\mathbf{1}_{\overline{u}}^T L_G \mathbf{1}_{\overline{v}} - W \cdot \mathbf{1}_{\overline{u}}^T L_H \mathbf{1}_{\overline{v}}),$$

so that

$$\left|w(\partial_G S) - W \cdot w(\partial_H S)\right| \leq \sum_{i,j,k,l} \sum_{u\in D^i_j, v\in D^k_l} \left|\mathbf{1}_{\overline{u}}^T L_G \mathbf{1}_{\overline{v}} - W \cdot \mathbf{1}_{\overline{u}}^T L_H \mathbf{1}_{\overline{v}}\right|.$$

By Claim 3.7, there are at most $((L+1)\tau/\phi)^2$ nonzero terms in the summation above. In order to apply Lemma 3.8 to each such term, we need to show that $\deg_{G^i}(u) \leq \tau\lambda/\phi$ and $\deg_{G^k}(v) \leq \tau\lambda/\phi$. Since S is an τ-unbalanced cut, we have

$$\deg_{G^i}(u) \leq \mathbf{vol}_{G^i}(D^i_j) \leq \sum_{j\in[k_i]} \mathbf{vol}_{G^i}(D^i_j) \leq \tau\lambda/\phi,$$

and similarly for $\deg_{G^k}(v)$. Therefore, by Lemma 3.8,

$$\left|w(\partial_G S) - W \cdot w(\partial_H S)\right| \leq \left(\frac{(L+1)\tau}{\phi}\right)^2 \cdot \epsilon\lambda,$$

as desired. □

[4] outside of computing the boundary-linked expander decomposition sequence

The rest of Section 3.2 is dedicated to proving Lemma 3.8.

Expand out $L_G = \sum_{e\in E} L_e$, where L_e is the Laplacian of the graph consisting of the single edge e of the same weight, so that $\mathbf{1}_{\overline{u}}^T L_e \mathbf{1}_{\overline{v}} \in \{-w(e), w(e)\}$ if exactly one endpoint of e is in $\overline{u}$ and exactly one endpoint of e is in $\overline{v}$, and $\mathbf{1}_{\overline{u}}^T L_e \mathbf{1}_{\overline{v}} = 0$ otherwise. Let $E_{\overline{u},\overline{v},+}$ denote the edges $e \in E$ with $\mathbf{1}_{\overline{u}}^T L_e \mathbf{1}_{\overline{v}} = w(e)$, and $E_{\overline{u},\overline{v},-}$ denote those with $\mathbf{1}_{\overline{u}}^T L_e \mathbf{1}_{\overline{v}} = -w(e)$.

3.2.1 Random Sampling Procedure. Consider the Benzcur-Karger random sampling procedure, which we will de-randomize in this section. Let $\widehat{H}$ be a subgraph of G with each edge $e \in E$ sampled independently with probability $w(e)/W$, which is at most 1 by the assumption of Theorem 3.1. Intuitively, the parameter $W \geq \lambda/f(n)$ is selected so that with probability close to 1, (11) holds over all i, k, u, v.

We now introduce our concentration bounds for the random sampling procedure, namely the classical multiplicative Chernoff bound. We state a form that includes bounds on the moment-generating function $\mathbb{E}[e^{tX}]$ obtained in the standard proof.

Lemma 3.10 (Multiplicative Chernoff bound)**.** *Let $X_1, \ldots, X_N$ be independent random variables that take values in $[0,1]$, and let $X = \sum_{i=1}^N X_i$ and $\mu = \mathbb{E}[X] = \sum_{i=1}^N p_i$. Fix a parameter δ, and define*

$$t^u = \ln(1+\delta) \qquad \text{and} \qquad t^l = \ln\left(\frac{1}{1-\delta}\right). \tag{12}$$

Then, we have the following upper and lower tail bounds:

$$\Pr[X > (1+\delta)\mu] \leq e^{-t^u(1+\delta)\mu}\,\mathbb{E}[e^{t^u X}] \leq e^{-\delta^2\mu/3}, \tag{13}$$

$$\Pr[X < (1-\delta)\mu] \leq e^{t^l(1-\delta)\mu}\,\mathbb{E}[e^{-t^l X}] \leq e^{-\delta^2\mu/3}. \tag{14}$$

We now describe our de-randomization by pessimistic estimators. Let $F \subseteq E$ be the set of edges for which a value $X_e \in \{0,1\}$ has already been set, so that F is initially $\emptyset$. For each i, k, vertices $u \in U^i, v \in U^k$, and sign $\circ \in \{+,-\}$ such that $E_{\overline{u},\overline{v},\circ} \neq \emptyset$, we first define a "local" pessimistic estimator $\Phi_{\overline{u},\overline{v},\circ}(\cdot)$, which is a function on the set of pairs (e, X_e) over all $e \in F$. The algorithm computes a 3-approximation $\widetilde{\lambda} \in [\lambda, 3\lambda]$ to the mincut with the $\widetilde{O}(m)$-time $(2+\epsilon)$-approximation algorithm of Matula [15], and sets

$$\mu_{\overline{u},\overline{v},\circ} = \frac{w(E_{\overline{u},\overline{v},\circ})}{W} \qquad \text{and} \qquad \delta_{\overline{u},\overline{v},\circ} = \frac{\epsilon\widetilde{\lambda}}{6w(E_{\overline{u},\overline{v},\circ})}. \tag{15}$$

Following (12), we define

$$t^u_{\overline{u},\overline{v},\circ} = \ln(1+\delta_{\overline{u},\overline{v},\circ}) \qquad \text{and} \qquad t^l_{\overline{u},\overline{v},\circ} = \ln\left(\frac{1}{1-\delta_{\overline{u},\overline{v},\circ}}\right), \tag{16}$$

and following the middle expressions (the moment-generating functions) in (13) and (14), we define

$$\begin{aligned}
&\Phi_{\overline{u},\overline{v},\circ}(\{(e,X_e) : e \in F\}) \\
&= e^{-t^u_{\overline{u},\overline{v},\circ}(1+\delta_{\overline{u},\overline{v},\circ})\mu_{\overline{u},\overline{v},\circ}} \prod_{e\in E_{\overline{u},\overline{v},\circ}\cap F} e^{t^u_{\overline{u},\overline{v},\circ} X_e} \prod_{e\in E_{\overline{u},\overline{v},\circ}\setminus F} \mathbb{E}[e^{t^u_{\overline{u},\overline{v},\circ} X_e}] \\
&\quad + e^{t^l_{\overline{u},\overline{v},\circ}(1-\delta_{\overline{u},\overline{v},\circ})\mu_{\overline{u},\overline{v},\circ}} \prod_{e\in E_{\overline{u},\overline{v},\circ}\cap F} e^{-t^l_{\overline{u},\overline{v},\circ} X_e} \prod_{e\in E_{\overline{u},\overline{v},\circ}\setminus F} \mathbb{E}[e^{-t^l_{\overline{u},\overline{v},\circ} X_e}].
\end{aligned}$$

Observe that if we are setting the value of $X_{e'}$ for a new edge $e' \in E_{\overline{u},\overline{v},\circ} \setminus F$, then by linearity of expectation, there is an assignment $X_{e'} \in \{0,1\}$ for which $\Phi_{\overline{u},\overline{v},\circ}(\cdot)$ does not decrease:

$$\Phi_{\overline{u},\overline{v},\circ}(\{(e,X_e) : e \in F\} \cup (e', X_{e'})) \le \Phi_{\overline{u},\overline{v},\circ}(\{(e,X_e) : e \in F\}).$$

Since the X_e terms are independent, we have that for any $t \in \mathbb{R}$ and $E' \subseteq E$,

$$\mathbb{E}\left[e^{t\sum_{e\in E'} X_e}\right] = \prod_{e\in E'} \mathbb{E}[e^{tX_e}].$$

By the independence above and the second inequalities in (13) and (14), the initial "local" pessimistic estimator $\Phi_{\overline{u},\overline{v},\circ}(\emptyset)$ satisfies

$$\begin{aligned}\Phi_{\overline{u},\overline{v},\circ}(\emptyset) &\le 2\exp\left(-\frac{\delta^2_{\overline{u},\overline{v},\circ}\mu_{\overline{u},\overline{v},\circ}}{3}\right)\\ &= 2\exp\left(-\frac{(\epsilon\widetilde{\lambda}/(6w(E_{\overline{u},\overline{v},\circ})))^2 \cdot w(E_{\overline{u},\overline{v},\circ})/W\cdot}{3}\right)\\ &= 2\exp\left(-\frac{\epsilon\widetilde{\lambda}^2}{108w(E_{\overline{u},\overline{v},\circ})W}\right).\end{aligned}$$

We would like the above expression to be less than 1. To upper bound $w(E_{\overline{u},\overline{v},\circ})$, note first that every edge $e \in E_{\overline{u},\overline{v},\circ}$ must, under the contraction from G all the way to G^i, map to an edge incident to u in G^i, which gives $w(E_{\overline{u},\overline{v},\circ}) \le \deg_{G^i}(u)$. Moreover, since $\deg_{G^i}(u) \le \tau\lambda/\phi$ by assumption, we have

$$w(E_{\overline{u},\overline{v},\circ}) \le \deg_{G^i}(u) \le \tau\lambda/\phi \tag{17}$$

so that

$$\begin{aligned}\Phi_{\overline{u},\overline{v},\circ}(\emptyset) \le 2\exp\left(-\frac{\epsilon\widetilde{\lambda}^2}{108(\tau\lambda/\phi)W}\right) &\le 2\exp\left(-\frac{\epsilon\lambda^2}{108(\tau\lambda/\phi)W}\right)\\ &= 2\exp\left(-\frac{\epsilon\phi\lambda}{108\tau W}\right).\end{aligned}$$

Assume that

$$W \le \frac{\epsilon\phi\lambda}{108\tau\ln\left(16(L+1)^2m\right)}, \tag{18}$$

which satisfies the bounds in Lemma 3.8, so that

$$\Phi_{\overline{u},\overline{v},\circ}(\emptyset) \le 2\exp\left(-\frac{\epsilon\phi\lambda}{108\tau W}\right) \le \frac{1}{8(L+1)^2m}.$$

Our actual, "global" pessimistic estimator $\Phi(\cdot)$ is simply the sum of the "local" pessimistic estimators:

$$\Phi(\{(e,X_e) : e \in F\}) = \sum_{\substack{i,k,\\ u\in U^i, v\in U^k,\\ \circ\in\{+,-\}}} \Phi_{\overline{u},\overline{v},\circ}(\{(e,X_e) : e \in F\}).$$

The initial pessimistic estimator $\Phi(\emptyset)$ satisfies

$$\Phi(\emptyset) = \sum_{\substack{i,k,\\ u\in U^i, v\in U^k,\\ \circ\in\{+,-\}}} \Phi_{\overline{u},\overline{v},\circ}(\emptyset) \le \sum_{\substack{i,k,\\ u\in U^i, v\in U^k,\\ \circ\in\{+,-\}}} \frac{1}{8(L+1)^2m} \overset{\text{Clm.3.12}}{\le} 4(L+1)^2m \cdot \frac{1}{8(L+1)^2m} = \frac{1}{2}.$$

Again, if we are setting the value of X_f for a new edge $f \in E \setminus F$, then by linearity of expectation, there is an assignment $X_f \in \{0,1\}$ for which $\Phi(\cdot)$ does not decrease:

$$\Phi(\{(e,X_e) : e \in F\} \cup (f, X_f)) \le \Phi(\{(e,X_e) : e \in F\}).$$

Therefore, if we always select such an assignment X_e, then once we have iterated over all $e \in E$, we have

$$\Phi(\{(e,X_e) : e \in E\}) \le \Phi(\emptyset) \le \frac{1}{2} \le 1. \tag{19}$$

This means that for each $i,k,u \in U^i, v \in U^k$, and sign $\circ \in \{+,-\}$,

$$\begin{aligned}&\Phi_{\overline{u},\overline{v},\circ}(\{(e,X_e) : e \in E\})\\ &= e^{-t^u_{\overline{u},\overline{v},\circ}(1+\delta_{\overline{u},\overline{v},\circ})\mu_{\overline{u},\overline{v},\circ}} \prod_{e\in E_{\overline{u},\overline{v},\circ}} e^{t^u_{\overline{u},\overline{v},\circ}X_e}\\ &\quad + e^{t^l_{\overline{u},\overline{v},\circ}(1-\delta_{\overline{u},\overline{v},\circ})\mu_{\overline{u},\overline{v},\circ}} \prod_{e\in E_{\overline{u},\overline{v},\circ}} e^{-t^l_{\overline{u},\overline{v},\circ}X_e}\\ &\le 1.\end{aligned}$$

In particular, each of the two terms is at most 1. Recalling from definition (15) that $\mu_{\overline{u},\overline{v},\circ} = w(E_{\overline{u},\overline{v},\circ})/W$ and $\delta_{\overline{u},\overline{v},\circ} = \epsilon\widetilde{\lambda}/(6w(E_{\overline{u},\overline{v},\circ}))$, we have

$$\sum_{e\in E_{\overline{u},\overline{v},\circ}} X_e \le (1+\delta_{\overline{u},\overline{v},\circ})\mu_{\overline{u},\overline{v},\circ} = \frac{w(E_{\overline{u},\overline{v},\circ})}{W} + \frac{\epsilon\widetilde{\lambda}}{6W}$$

and

$$\sum_{e\in E_{\overline{u},\overline{v},\circ}} X_e \ge (1-\delta_{\overline{u},\overline{v},\circ})\mu_{\overline{u},\overline{v},\circ} = \frac{w(E_{\overline{u},\overline{v},\circ})}{W} - \frac{\epsilon\widetilde{\lambda}}{6W}.$$

Therefore,

$$\begin{aligned}\left|\mathbf{1}_{\overline{u}}^T L_G \mathbf{1}_{\overline{v}} - W\cdot\mathbf{1}_{\overline{u}}^T L_{\widehat{H}}\mathbf{1}_{\overline{v}}\right| &\le \sum_{\circ\in\{+,-\}}\left|w(E_{\overline{u},\overline{v},\circ}) - W\cdot\sum_{e\in E_{\overline{u},\overline{v},\circ}} X_e\right|\\ &\le \frac{\epsilon\widetilde{\lambda}}{6} + \frac{\epsilon\widetilde{\lambda}}{6} = \frac{\epsilon\widetilde{\lambda}}{3} \le \epsilon\lambda,\end{aligned}$$

fulfilling (11).

It remains to consider the running time. We first bound the number of i,k,u,v such that either $E_{\overline{u},\overline{v},+} \neq \emptyset$ or $E_{\overline{u},\overline{v},-} \neq \emptyset$; the others are irrelevant since $\mathbf{1}_{\overline{u}}^T L_G \mathbf{1}_{\overline{v}} = \mathbf{1}_{\overline{u}}^T L_{\widehat{H}}\mathbf{1}_{\overline{v}} = 0$.

Claim 3.11. *For each pair of vertices x,y, there are at most $(L+1)^2$ many selections of i,k and $u \in U^i, v \in U^k$ such that $x \in \overline{u}$ and $y \in \overline{v}$.*

Proof. For each level i, there is exactly one vertex $u \in U^i$ with $x \in \overline{u}$, and for each level k, there is exactly one vertex $v \in U^k$ with $y \in \overline{v}$. This makes $(L+1)^2$ many choices of i,k total, and unique choices for u,v given i,k. □

Claim 3.12. *For each edge $e \in E$, there are at most $4(L+1)^2$ many selections of i,k and $u \in U^i, v \in U^k$ such that $e \in E_{\overline{u},\overline{v},+} \cup E_{\overline{u},\overline{v},-}$.*

Proof. If $e \in E_{\overline{u},\overline{v},+} \cup E_{\overline{u},\overline{v},-}$, then exactly one endpoint of e is in $\overline{u}$ and exactly one endpoint of e is in $\overline{v}$. There are four possibilities as to which endpoint is in $\overline{u}$ and which is in $\overline{v}$, and for each, Claim 3.11 gives at most $(L+1)^2$ choices. □

Claim 3.13. *There are at most $4(L+1)^2 m$ many choices of i, k, u, v such that either $E_{\overline{u},\overline{v},+} \neq \emptyset$ or $E_{\overline{u},\overline{v},-} \neq \emptyset$.*

Proof. For each such choice, charge it to an arbitrary edge $(x, y) \in E_{\overline{u},\overline{v},+} \cup E_{\overline{u},\overline{v},-}$. Each edge is charged at most $4(L+1)^2$ times by Claim 3.12, giving at most $4(L+1)^2 m$ total charges. □

By Claim 3.12, each new edge $e \in E \setminus F$ is in at most $4(L+1)^2$ many sets $E_{\overline{u},\overline{v},\circ}$, and therefore affects at most $4(L+1)^2$ many terms $\Phi_{\overline{u},\overline{v},\circ}(\{(e, X_e) : e \in F\})$. The algorithm only needs to re-evaluate these terms with the new variable X_e set to 0 and with it set to 1, and take the one with the smaller new $\Phi(\cdot)$. This takes $O(L^2)$ arithmetic operations.

How long do the arithmetic operations take? We compute each exponential in $\Phi(\cdot)$ with $c \log n$ bits of precision after the decimal point for some constant $c > 0$, which takes polylog(n) time. Each one introduces an additive error of $1/n^c$, and there are poly(n) exponential computations overall, for a total of $1/n^c \cdot \text{poly}(n) \leq 1/2$ error for a large enough $c > 0$. Factoring in this error, the inequality (19) instead becomes

$$\Phi(\{(e, X_e) : e \in E\}) \leq \Phi(\emptyset) + \frac{1}{2} \leq \frac{1}{2} + \frac{1}{2} = 1,$$

so the rest of the bounds still hold.

This concludes the proof of Lemma 3.8.

3.2.2 Balanced Case. Similar to the expander case, we treat balanced cuts by "overlaying" a "lossy", $n^{o(1)}$-approximate sparsifier of G top of the graph $\widehat{H}$ obtained from Lemma 3.9. In the expander case, this sparsifier was just another expander, but for general graphs, we need to do more work. At a high level, we compute an expander decomposition sequence, and on each level, we replace each of the expanders with a fixed expander (like in the expander case). Due to the technical proof and lack of novel ideas, we defer the proof to the full version.

Theorem 3.14. *Let G be an weighted multigraph with mincut λ whose edges have weight at most $O(\lambda)$. For any parameters $\widetilde{\lambda} \in [\lambda, 3\lambda]$ and $\Delta \geq 2^{O(\log n)^{5/6}}$, we can compute, in deterministic $2^{O(\log n)^{5/6}(\log\log n)^{O(1)}} m + O(\Delta m)$ time, an unweighted multigraph H such that $W \cdot H$ is a γ-approximate cut sparsifier of G, where $\gamma \leq 2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}$ and $W = \widetilde{\lambda}/\Delta$. (The graph H does not need to be a subgraph of G.) Moreover, the algorithm does not need to know the mincut value λ.*

3.2.3 Combining Them Together. We now combine the unbalanced and balanced cases to prove Theorem 3.1, restated below.

Theorem 3.1. *There exists a function $f(n) \leq 2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}$ such that the following holds. Let G be a graph with mincut λ **and maximum edge weight at most** $\epsilon^4 \lambda / f(n)$. For any $0 < \epsilon \leq 1$, we can compute, in deterministic $2^{O(\log n)^{5/6}(\log\log n)^{O(1)}} m$ time, an unweighted graph H and some weight $W \geq \epsilon^4 \lambda / f(n)$ such that the two properties of Theorem 1.5 hold, i.e.,*

(1) *For any mincut S^* of G, we have $W \cdot |\partial_H S^*| \leq (1+\epsilon)\lambda$, and*

Par.	Value
λ	Mincut of G
$\widetilde{\lambda}$	3-approximation of λ
ϵ	Given as input
r	$(\log n)^{1/6}$
β	$(\log n)^{-O(r^4)}$ from Theorem 3.2
ϕ	$(\log n)^{-r^5}$
L	$O(\frac{\log n}{r^5})$
γ	$2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}$ from Theorem 3.14
Δ	$2^{\Theta(\log n)^{5/6}}$ from Theorem 3.14
τ	$\beta^{-cL}\gamma^2/\epsilon$ for large enough constant $c > 0$
ϵ'	$\frac{1}{2}(\frac{\phi}{(L+1)\tau})^2\epsilon$
$\widehat{W}$	$\min\{\frac{C\epsilon'\phi\widetilde{\lambda}}{\tau \ln(Lm)}, \frac{\widetilde{\lambda}}{\Delta}\}$ where $C > 0$ is the constant from Lemma 3.9
$\widetilde{W}$	$\frac{\epsilon}{2\gamma} \cdot \frac{\widetilde{\lambda}}{\Delta}$

Figure 1: The parameters in the proof of Theorem 3.1.

(2) *For any cut $\emptyset \subsetneq S \subsetneq V$ of G, we have $W \cdot |\partial_H S| \geq (1-\epsilon)\lambda$.*

Our high-level procedure is similar to the one from the expander case. For the τ-unbalanced cuts, we use Lemma 3.9. For the balanced cuts, we show that their size must be much larger than λ, so that even on a γ-approximate weighted sparsifier guaranteed by Theorem 3.14, their weight is still much larger than λ. We then "overlay" the γ-approximate weighted sparsifier with a "light" enough weight onto the sparsifier of τ-unbalanced cuts. The weight is light enough to barely affect the mincuts, but still large enough to force any balanced cut to increase by at least λ in weight.

Claim 3.15. *If a cut S is balanced, then $w(\partial_G S) \geq \beta^{O(L)}\tau\lambda$.*

Proof. Consider the level i for which $\sum_{j\in[k_i]} \mathbf{vol}_{G^i}(D_j^i) > \tau\lambda/\phi$. For each $j \in [k_i]$, we have

$$\begin{aligned}
\mathbf{vol}_{G^i}(D_j^i) &= \mathbf{vol}_{G^i[U_j^i]}(D_j^i) + w(E_{G^i}(D_j^i, U^i \setminus U_j^i)) \\
&\overset{(8)}{\leq} \frac{1}{\phi} w(\partial_{G^i[U_j^i]} D_j^i) + w(E_{G^i}(D_j^i, U^i \setminus U_j^i)) \\
&\leq \frac{1}{\phi}\left(w(\partial_{G^i[U_j^i]} D_j^i) + w(E_{G^i}(D_j^i, U^i \setminus U_j^i))\right) \\
&= \frac{1}{\phi} w(\partial_{G^i} D_j^i),
\end{aligned}$$

so summing over all $j \in [k_i]$,

$$\sum_{j\in[k_i]} \frac{1}{\phi} w(\partial_{G^i} D_j^i) \geq \sum_{j\in[k_i]} \mathbf{vol}_{G^i}(D_j^i) > \frac{\tau\lambda}{\phi}.$$

By Lemma 3.4, it follows that

$$w(\partial_G S) \geq \beta^{O(L)} \sum_{j\in[k_i]} w(\partial_G^i D_j^i) \geq \beta^{O(L)}\tau\lambda.$$

□

We now set some of our parameters; see Figure 1 for a complete table of the parameters in our proof. For $r := (\log n)^{1/6}$, let $\beta := (\log n)^{-O(r^4)}$ and $\phi := (\log n)^{-r^5}$, so that by Theorem 3.2, the total weight of inter-cluster edges, and therefore the total weight of the

next graph in the expander decomposition sequence, shrinks by factor $(\log n)^{O(r^4)}\phi = (\log n)^{-\Omega(r^5)}$. Since edge weights are assumed to be polynomially bounded, this shrinking can only happen $O(\frac{\log n}{r^5})$ times, so $L \le O(\frac{\log n}{r^5})$.

Let $\widetilde{\lambda} \in [\lambda, 3\lambda]$ be a 3-approximation to the mincut, computable in $\widetilde{O}(m)$ time [15], Let $\epsilon' := \frac{1}{2}(\frac{\phi}{(L+1)\tau})^2\epsilon$ for parameter τ that we set later, and let $\widehat{H}$ be the sparsifier of τ-unbalanced cuts from Lemma 3.9 for this value of ϵ' (instead of ϵ) and the following value of $\widehat{W} \le \frac{C\epsilon'\phi\lambda}{\tau \ln(Lm)}$ (taking the place of W):

$$\widehat{W} := \min\left\{\frac{C\epsilon'\phi\widetilde{\lambda}}{3\tau\ln(Lm)}, \frac{\widetilde{\lambda}}{\Delta}\right\} = \min\left\{\Omega\left(\frac{\epsilon\phi^3\widetilde{\lambda}}{\tau^3L^2\ln(Lm)}\right), \frac{\widetilde{\lambda}}{\Delta}\right\}.$$

Let $\widetilde{H}$ be the unweighted graph from Theorem 3.14 applied to $\widetilde{\lambda}$ and Δ, so that $\widetilde{\lambda}/\Delta \cdot \widetilde{H}$ is a γ-approximate cut sparsifier for $\gamma := 2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}$. Define $\widetilde{W} := \frac{\epsilon}{2\gamma} \cdot \frac{\widetilde{\lambda}}{\Delta}$, and let H' be the "union" of the graph $\widehat{H}$ weighted by $\widehat{W}$ and the graph $\widetilde{H}$ weighted by $\widetilde{W}$. More formally, consider a weighted graph H' where each edge (u, v) is weighted by $\widehat{W} \cdot w_{\widehat{H}}(u, v) + \widetilde{W} \cdot w_{\widetilde{H}}(u, v)$.

For an τ-unbalanced cut ∂S, the addition of the graph $\widetilde{H}$ weighted by $\widetilde{W}$ increases its weight by

$$\widetilde{W} \cdot w(\partial_{\widetilde{H}} S) = \frac{\epsilon}{2\gamma} \cdot \left(\frac{\lambda}{\Delta} w(\partial_{\widetilde{H}} S)\right) \le \frac{\epsilon}{2\gamma} \cdot \gamma w(\partial_G S) = \frac{\epsilon}{2} w(\partial_G S),$$

so that

$$\begin{aligned}
&\left|w(\partial_G S) - \left(\widehat{W} \cdot w(\partial_{\widehat{H}} S) + \widetilde{W} \cdot w(\partial_{\widetilde{H}} S)\right)\right| \\
&\le \left|w(\partial_G S) - \widehat{W} \cdot w(\partial_{\widehat{H}} S)\right| + \widetilde{W} \cdot w(\partial_{\widetilde{H}} S^*) \\
&\le \left(\frac{(L+1)\tau}{\phi}\right)^2 \cdot \epsilon'\lambda + \frac{\epsilon}{2} w(\partial_G S) \\
&= \frac{\epsilon\lambda}{2} + \frac{\epsilon}{2} w(\partial_G S) \\
&\le \epsilon w(\partial_G S).
\end{aligned}$$

In particular, any τ-unbalanced cut satisfies

$$(1-\epsilon)\lambda \le \widehat{W} \cdot w(\partial_{\widehat{H}} S) + \widetilde{W} \cdot w(\partial_{\widetilde{H}} S) \le (1+\epsilon)\lambda. \tag{20}$$

Next, we show that all balanced cuts have weight at least λ in the graph $\widetilde{H}$ weighted by $\widetilde{W}$. This is where we finally set $\tau := \beta^{-cL}\gamma^2/\epsilon$ for large enough constant $c > 0$. For a balanced cut S,

$$\begin{aligned}
\widetilde{W} \cdot w(\partial_{\widetilde{H}} S) = \frac{\epsilon}{2\gamma} \cdot \left(\frac{\lambda}{\Delta} w(\partial_{\widetilde{H}} S)\right) &\ge \frac{\epsilon}{2\gamma} \cdot \left(\frac{1}{\gamma} w(\partial_G S)\right) \\
&\overset{\text{Clm.3.15}}{\ge} \frac{\epsilon}{\gamma^2} \cdot \beta^{O(L)}\tau\lambda \ge \lambda.
\end{aligned}$$

Moreover, by Claim 3.6 for this value of $\tau \ge \beta^{-O(L)}$, the mincut ∂S^* is τ-unbalanced, and therefore has weight at least $(1-\epsilon)\lambda$ in H' by (20).

Therefore, H' preserves the mincut up to factor ϵ and has mincut at least $(1-\epsilon)\lambda$. It remains to make all edge weights the same on this sparsifier. Since $\widetilde{W} = \frac{\epsilon}{2\gamma} \cdot \frac{\widetilde{\lambda}}{\Delta}$ and the only requirement for Δ from Theorem 3.14 is that $\Delta \ge 2^{O(\log n)^{5/6}}$, we can increase or decrease Δ by a constant factor until either $\widetilde{W}/\widehat{W}$ or $\widehat{W}/\widetilde{W}$ is an integer. Then, we can let $W := \min\{\widehat{W}, \widetilde{W}\}$ and define the unweighted graph H so that $\#_H(u, v) = w_{H'}(u, v)/W$ for all $u, v \in V$. Therefore, our final weight W is

$$\begin{aligned}
W = \min\{\widehat{W}, \widetilde{W}\} &= \min\left\{\Omega\left(\frac{\epsilon\phi^3\widetilde{\lambda}}{\tau^3L^2\ln(Lm)}\right), \frac{\widetilde{\lambda}}{\Delta}, \frac{\epsilon}{2\gamma} \cdot \frac{\widetilde{\lambda}}{\Delta}\right\} \\
&\ge \epsilon^4 2^{-O(\log n)^{5/6}(\log\log n)^{O(1)}}\lambda,
\end{aligned}$$

so we can set $f(n) := 2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}$, as desired.

Finally, we bound the running time. The expander decomposition sequence (Theorem 3.2) takes time $m^{1+O(1/r)} + \widetilde{O}(m/\phi^2)$, the unbalanced case (Theorem 3.2) takes time $\widetilde{O}(L^2 m)$, and the balanced case takes time $2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}m$. Altogether, the total is $2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}m$, which concludes the proof of Theorem 3.1.

3.3 Removing the Maximum Weight Assumption

Let $f(n) = 2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}$ be the function from Theorem 3.1. In this section, we show how to use Theorem 3.1, which assumes that the maximum edge weight in G is at most $\epsilon^4\lambda/f(n)$, to prove Theorem 1.5, which makes no assumption on edge weights.

First, we show that we can assume without loss of generality that the maximum edge weight in G is at most 3λ. To see why, the algorithm can first compute a 3-approximation $\widetilde{\lambda} \in [\lambda, 3\lambda]$ to the mincut with the $\widetilde{O}(m)$-time $(2+\epsilon)$-approximation algorithm of Matula [15], and for each edge in G with weight more than $\widetilde{\lambda}$, reduce its weight to $\widetilde{\lambda}$. Let the resulting graph be $\widetilde{G}$. We now claim the following:

Claim 3.16. *Suppose an unweighted graph H and some weight W satisfy the two properties of Theorem 1.5 for $\widetilde{G}$. Then, they also satisfy the two properties of Theorem 1.5 for G.*

Proof. The only cuts that change value between G and $\widetilde{G}$ are those with an edge of weight more than $\widetilde{\lambda}$, which means their value must be greater than $\widetilde{\lambda} \ge \lambda$. In particular, since G and $\widetilde{G}$ have the same mincuts and the same mincut values, both properties of Theorem 1.5 also hold when the input graph is G. □

For the rest of the proof, we work with $\widetilde{G}$ instead of G. Define $\widetilde{W} := \epsilon^4\widetilde{\lambda}/(3f(n))$, which satisfies $\widetilde{W} \le \epsilon^4\lambda/f(n)$. For each edge e in $\widetilde{G}$, split it into $\lceil w(e)/\widetilde{W}\rceil$ parallel edges of weight at most $\widetilde{W}$ each, whose sum of weights equals $w(e)$; let the resulting graph be $\widehat{G}$. Apply Theorem 3.1 on $\widehat{G}$, which returns an unweighted graph H and weight $W \ge \epsilon^4\lambda/f(n)$ such that the two properties of Theorem 1.5 hold for $\widehat{G}$. Clearly, the cuts are the same in $\widetilde{G}$ and $\widehat{G}$: we have $w(\partial_{\widetilde{G}} S) = w(\partial_{\widehat{G}} S)$ for all $S \subseteq V$. Therefore, the two properties also hold for $\widetilde{G}$, as desired.

We now bound the size of G' and the running time. Since $w(e) \le \widetilde{\lambda}$, we have $\lceil w(e)/\widetilde{W}\rceil \le \lceil 3f(n)/\epsilon^4\rceil$, so each edge splits into at most $O(f(n)/\epsilon^4)$ edges and the total number of edges is $\widehat{m} \le O(f(n)/\epsilon^4) \cdot$

m. Therefore, Theorem 3.1 takes time $2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}\widehat{m} = \epsilon^{-4}2^{O(\log n)^{5/6}(\log\log n)^{O(1)}}m$, concluding the proof of Theorem 1.5.

ACKNOWLEDGEMENTS

I am indebted to Sivakanth Gopi, Janardhan Kulkarni, Jakub Tarnawski, and Sam Wong for their supervision and encouragement on this project while I was a research intern at Microsoft Research, as well as providing valuable feedback on the manuscript. I also thank Thatchaphol Saranurak for introducing me to the boundary-linked expander decomposition framework [6].

REFERENCES

[1] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. 2019. A Deterministic Algorithm for Balanced Cut with Applications to Dynamic Connectivity, Flows, and Beyond. *arXiv preprint arXiv:1910.08025* (2019).

[2] Harold N Gabow. 1995. A matroid approach to finding edge connectivity and packing arborescences. *J. Comput. System Sci.* 50, 2 (1995), 259–273.

[3] Andrew V Goldberg and Satish Rao. 1998. Beyond the flow decomposition barrier. *Journal of the ACM (JACM)* 45, 5 (1998), 783–797.

[4] Andrew V. Goldberg and Robert Endre Tarjan. 1988. A new approach to the maximum-flow problem. *J. ACM* 35, 4 (1988), 921–940. https://doi.org/10.1145/48014.61051

[5] Ralph E Gomory and Tien Chung Hu. 1961. Multi-terminal network flows. *J. Soc. Indust. Appl. Math.* 9, 4 (1961), 551–570.

[6] Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. 2020. The Expander Hierarchy and its Applications to Dynamic Graph Algorithms. *arXiv preprint arXiv:2005.02369* (2020).

[7] Jianxiu Hao and James B Orlin. 1992. A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 165–174.

[8] Monika Henzinger, Satish Rao, and Di Wang. 2017. Local Flow Partitioning for Faster Edge Connectivity. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. 1919–1938. https://doi.org/10.1137/1.9781611974782.125

[9] David R Karger. 1993. Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm.. In *SODA*, Vol. 93. 21–30.

[10] David R. Karger. 2000. Minimum cuts in near-linear time. *J. ACM* 47, 1 (2000), 46–76. https://doi.org/10.1145/331605.331608

[11] David R Karger and Clifford Stein. 1996. A new approach to the minimum cut problem. *Journal of the ACM (JACM)* 43, 4 (1996), 601–640.

[12] Ken-ichi Kawarabayashi and Mikkel Thorup. 2018. Deterministic edge connectivity in near-linear time. *Journal of the ACM (JACM)* 66, 1 (2018), 1–50.

[13] Jason Li and Debmalya Panigrahi. 2020. Deterministic Min-cut in Poly-logarithmic Max-flows. In *FOCS*.

[14] Yang P Liu and Aaron Sidford. 2020. Faster Divergence Maximization for Faster Maximum Flow. *arXiv preprint arXiv:2003.08929* (2020).

[15] David W Matula. 1993. A linear time 2+ ε approximation algorithm for edge connectivity. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*. 500–504.

[16] Hiroshi Nagamochi and Toshihide Ibaraki. 1992. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM Journal on Discrete Mathematics* 5, 1 (1992), 54–66.

[17] Hiroshi Nagamochi and Toshihide Ibaraki. 1992. A Linear-Time Algorithm for Finding a Sparse k-Connected Spanning Subgraph of a k-Connected Graph. *Algorithmica* 7, 5&6 (1992), 583–596. https://doi.org/10.1007/BF01758778

[18] Thatchaphol Saranurak. 2021. A Simple Deterministic Algorithm for Edge Connectivity. In *Symposium on Simplicity in Algorithms (SOSA)*. SIAM, 80–85.

[19] Daniel A. Spielman and Shang-Hua Teng. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*. ACM, 81–90.

[20] Mechthild Stoer and Frank Wagner. 1997. A simple min-cut algorithm. *Journal of the ACM (JACM)* 44, 4 (1997), 585–591.