All-Pairs Minimum Cut using $\tilde{O}(n^{7/4})$ Cut Queries

Yotam Kenneth-Mordoch* Robert Krauthgamer†

Abstract. We present the first non-trivial algorithm for the all-pairs minimum cut problem in the cut-query model. Given cut-query access to an unweighted graph $G = (V, E)$ with n vertices, our randomized algorithm constructs a Gomory-Hu tree of G, and thus solves the all-pairs minimum cut problem, using $\tilde{O}(n^{7/4})$ cut queries.

1 Introduction Optimization over graphs, such as finding a minimum s,t-cut, is a fundamental topic in computer science due to its wide range of applications from network design to combinatorial optimization and computational complexity. Accordingly, there exists a vast body of work on solving such problems efficiently in many different models of computation. One recently suggested model is the *cut-query model* [RSW18], where algorithms access the input, which is an unweighted graph $G = (V, E)$, only through cut queries: the query specifies $S \subseteq V$ and obtains in return the size of the corresponding cut, i.e. $\text{cut}_G(S) \coloneqq |E(S, V \setminus S)|$, where $E(S, T)$ is the set of edges with one endpoint in S and the other in T.

There are two main motivations for studying the cut-query model. The first one is scenarios where it is costly or even impossible to access graph edges directly, for example in genomic sequencing that models the structure of a genome as a graph. It is impossible to directly access the graph edges, but one can perform an experiment to compute the number of edges crossing a cut in the graph, and it is imperative to minimize the number of experiments [GK98, ABK+02, BGK05, CK08].[1] Another example is distributed systems, where many low-power devices with limited communication capabilities form a graph. A cut query can be implemented by a central server sending a message to all devices in a subset $S \subseteq V$ and asking them to report the number of their incident edges crossing the cut, which is more efficient than asking them to report all their neighbors, as in the standard neighbor-query model.

The second motivation is that graph-cut functions are a natural example of submodular functions, and hence the cut-query model is a natural model for studying submodular function minimization (SFM). A *submodular function* is a set function $f : 2^V \to \mathbb{R}$ that satisfies the diminishing returns property

$$\forall S \subseteq T \subseteq V, \forall v \in V \setminus T, \qquad f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T).$$

Denoting $n = |V|$, the minimum of a submodular function f can be found using $\tilde{O}(n^2)$ value queries to f [LSW15, MN21, Jia23], where we use the usual notation $\tilde{O}(\cdot)$ to hide polylogarithmic factors in n. Meanwhile, the known lower bound for SFM is only $\Omega(n)$ queries [GPRW20, LLSZ21], and $\Omega(n \log n)$ queries for deterministic algorithms [CGJS22]. The special case of graph cuts therefore poses an intriguing case study to either find stronger lower bounds or inspire better algorithms for SFM. In particular, while global minimum cut can be found using $O(n)$ queries, known algorithms for minimum s,t-cut use $O(n^{5/3})$ queries [RSW18, MN20, AEG+22, ASW25], leaving a substantial gap to the $\Omega(n)$ query lower bound (notice that it is a submodular minimization problem on $n - 2$ variables).

We study the all-pairs minimum cut problem, whose objective is to find a minimum s,t-cut for every pair of vertices $s, t \in V$ in an unweighted graph $G = (V, E)$. Recent advances reduce this problem to solving polylog n instances of minimum s,t-cut on contracted subgraphs of G, and the reduction itself only takes time that is linear

*Weizmann Institute of Science, Rehovot, Israel (yotam.kenneth@weizmann.ac.il).

†Weizmann Institute of Science, Rehovot, Israel (robert.krauthgamer@weizmann.ac.il). The Harry Weinrebe Professorial Chair of Computer Science. Work partially supported by the Israel Science Foundation grant #1336/23, by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center, and by a research grant from the Estate of Harry Schutzman.

[1] The access model used is additive queries, where given a set $S \subseteq V$ the query returns the number of edges in $E \cap S \times S$. It is easy to see that one can simulate a cut query using $O(1)$ additive queries.

in $|E|$ [AKT20, AKT21b, LPS21, AKT21a, AKT22, AKL+22, AKL+25].[2] It seems likely that these techniques extend to the cut-query model, yielding a reduction to polylog n instances of minimum s,t-cut also in the cut-query model. Since minimum s,t-cut is a submodular minimization problem (on $n-2$ variables), such a reduction would imply that the query complexity of the all-pairs minimum cut problem is at most polylog n times that of SFM. In that case, proving a truly super-linear (exceeding linear by more than polylogarithmic factors) lower bound for all-pairs minimum cut would surpass the currently known lower bounds for SFM. Conversely, our algorithmic results rule out establishing an $\tilde{\Omega}(n^2)$ lower bound for SFM via the all-pairs minimum cut problem in unweighted graphs.

Note that it is possible to recover the entire graph G using $O(n^2)$ cut queries [CK08, BM11, RSW18], after which the algorithm can solve any minimization problem without additional cut queries. Hence, our focus is on subquadratic query complexity.

Our main result is the first non-trivial algorithm for the all-pairs minimum cut problem in the cut-query model, improving upon the naive approach that recovers the entire graph using $O(n^2)$ cut queries. Our algorithm is based on efficiently constructing a Gomory-Hu tree [GH61], a well-known data structure that compactly represents the minimum s,t-cut for every pair of vertices $s,t \in V$ in a graph $G=(V,E)$.

THEOREM 1.1. *There exists a randomized algorithm for constructing a Gomory-Hu tree of an unweighted graph $G=(V,E)$ with n vertices, that uses $\tilde{O}(n^{7/4})$ cut queries and succeeds with probability $1-1/\operatorname{poly}(n)$.*

1.1 Technical Overview We now provide an overview of our algorithms and their technical components, and along the way explain our two main technical contributions. The first one is introducing a weaker version of a problem called isolating-cuts. The key point is that this relaxation strikes a good balance – it suffices for our intended application of constructing a Gomory-Hu tree, yet can be solved efficiently using cut queries. The second contribution is a new randomized contraction procedure, termed (τ,F)-star contraction. This procedure solves minimum s,t-cut, and is actually an adaptation of previous procedures that solve the easier task of global minimum cut.

Our algorithm for the all-pairs minimum cut problem constructs for G a *Gomory-Hu tree* [GH61], which is a weighted tree on the same vertex set V such that the minimum s,t-cut in the tree is also the minimum s,t-cut in G. The classical algorithm of Gomory and Hu [GH61] is based on iteratively refining the graph into a tree structure, using $n-1$ calls to a minimum s,t-cut procedure. Recent work [AKT21b, AKT21a, AKT22, LPS21] has managed to accelerate this refinement process using calls to a procedure that solves single-source minimum cut, and these offer more choices for which s,t-cuts to refine by. The refinement process then works effectively like a logarithmic-depth recursion, where the input graph is partitioned recursively, and the size of each part decreases by a constant factor each time.

Our algorithm follows this general approach, and one technical difference is that given a partition of the vertices, instead of solving single-source minimum cut in each part separately, we solve the problem for all the parts in parallel. This difference is crucial in the cut-query model, essentially because the bottleneck is information theoretic, and our parallel version aggregates information from the different parts. The following lemma presents the algorithm that constructs a Gomory-Hu tree using calls to single-source minimum cut; its proof is similar to the one in [AKT21a] and appears in Appendix D.

LEMMA 1.2. *Suppose that given cut-query access to an unweighted graph G on n vertices, a set of perturbation edges that guarantee unique minimum s,t-cuts, and a partition of the vertex set $V_1,\ldots,V_k$ with pivots $\{p_i \in V_i\}_{i\in[k]}$, one can return for every $i\in[k]$ and $v\in V_i$ the minimum p_i,v-cut S_v using $q(n)$ cut queries and with success probability $1-\rho_F(n)$.*

Then, one can compute a Gomory-Hu tree of an input graph on n vertices using $\tilde{O}(q(n)+n^{1.5})$ cut queries. The algorithm is randomized and succeeds with probability $1-O(\rho_F(n)\cdot\log^2 n)-1/\operatorname{poly}(n)$.

The main focus of our work is thus to solve the single-source minimum cut problem in the cut-query model. We present an efficient algorithm for this problem in the following lemma, whose proof appears in Section 5.

LEMMA 1.3 (Single-Source Minimum Cut). *Given cut-query access to an unweighted graph G on n vertices, a set of perturbation edges $\tilde{E}$ that guarantee unique minimum s,t-cuts, and a partition of the vertex set $V_1,\ldots,V_k$*

[2] Since these graphs are contracted subgraphs of G, they may have parallel edges, preventing the direct application of existing minimum s,t-cut algorithms in the cut-query model. For further discussion, see Subsection 1.1.

with pivots $\{p_i \in V_i\}_{i\in[k]}$*, one can compute for every* $i \in [k]$ *and* $v \in V_i$ *the minimum* p_i, v*-cut* S_v *using* $q(n) = \tilde{O}(n^{1.75})$ *cut queries. The algorithm is randomized and succeeds with probability* $1 - 1/\operatorname{poly}(n)$*.*

Theorem 1.1 follows immediately from Theorems 1.2 and 1.3. The main technical tool for solving the single-source minimum cut problem is procedure for (a relaxed version of) the isolating-cuts problem, which we define next.

DEFINITION 1.4. *In the* isolating-cuts problem*, the input is a graph* $G = (V, E)$ *and a set* $R \subseteq V$*, where* $|R| \geq 2$*, and the goal is to output for each* $v \in R$ *some* $S_v \subset V$ *that is a* $(v, R \setminus v)$*-minimum cut.*

It was shown in [LP20, AKT21b] how to reduce the isolating-cuts problem to $O(\log n)$ instances of minimum s, t-cut in contracted multigraphs. The best algorithm for computing minimum s, t-cuts in the cut-query model uses $\tilde{O}(n^{4/3} \cdot c^{1/3})$ cut queries, where c is the value of the cut [RSW18, ASW25],[3] but unfortunately in a contracted multigraph c can be as large as $\Omega(n^2)$. For example, take a random graph $G(n, 1/2)$ and contract two random vertex subsets of cardinality $n/4$ into two vertices s, t; the resulting multigraph is likely to have $\Theta(n)$ vertices and minimum s, t-cut of value $\Omega(n^2)$. Therefore, existing algorithms do not solve such minimum s, t-cut instances efficiently.

Our first technical contribution is to introduce a relaxed variant of the isolating-cuts problem, termed *weak isolating cuts*, that suffices for solving the single-source minimum cut problem yet can be solved efficiently in the cut-query model.[4] In this relaxation, the goal is to compute a minimum $(v, R \setminus v)$-cut only if it is also a minimum v, u-cut for some $u \in R$, and otherwise any $(v, R \setminus v)$-cut can be returned. We observe that the existing algorithms for single-source minimum cut work even when calls to isolating cuts are replaced with calls to our relaxed version. Our main technical result is thus to solve the relaxed problem directly, i.e., without going through minimum s, t-cut in contracted multigraphs, stated as follows.

LEMMA 1.5 (Weak Isolating Cuts). *Given cut-query access to an unweighted graph* $G = (V, E)$ *on* n *vertices, and also a set of vertices* $R \subseteq V$ *with maximum degree* $d = \max_{v\in R} d_G(v)$ *and a set of perturbation edges* $\tilde{E}$ *that guarantee unique minimum* s, t*-cuts, one can solve the weak isolating-cuts problem on* R *using* $\tilde{O}(\min\{nd, n^{7/4}\})$ *cut queries. The algorithm is randomized and succeeds with probability* $1 - 1/\operatorname{poly}(n)$*.*

The proof of this lemma, which appears in Section 3, is based on splitting the minimum $(v, R \setminus v)$-cuts into two sets, of friendly cuts and unfriendly cuts. Given a cut $C \subseteq V$, let $\operatorname{cr}_C(v)$ be the number of edges in $E(C, V \setminus C)$ that are incident to v, and let us omit the subscript when clear from context. Define the *friendliness ratio* of a vertex $v \in V$ (with respect to C) as $1 - \operatorname{cr}(v)/\deg(v)$, where $\deg(v)$ is the degree of v in G. The cut C is called α*-friendly* if every vertex $v \in V$ has friendliness ratio $1 - \operatorname{cr}(v)/\deg(v) \geq \alpha$, and otherwise it is called α*-unfriendly*. The case of friendly cuts is handled using a friendly cut sparsifier, which was first introduced in [AKT22] and is described next.[5]

Throughout, a *contraction* of $G = (V, E)$ is a graph H obtained from G by contracting some subsets of vertices. A *super-vertex* is a vertex in H formed by contracting multiple vertices in G. Observe that contracting two vertices $u, v \in V$ is equivalent to adding an edge of infinite capacity between them. This view is useful because now H has the same vertex set as G, and moreover cuts in H are at least as large as those in G (when comparing the same $S \subset V$). In the cut-query model, one can simulate access to a contraction H of the input graph G by replacing each super-vertex in H with the vertices in V that form it.

DEFINITION 1.6. *An* (α, w)-friendly cut sparsifier *of a graph* G *is a contraction* H *of* G *such that for every* α*-friendly cut* $C \subseteq V$ *of* G *with at most* w *edges we have* $\operatorname{cut}_H(C) = \operatorname{cut}_G(C)$*.*

We show through a straightforward extension of [AKT22] an algorithm in the cut-query model that recovers (i.e., reports explicitly) the edges of an (α, w)-friendly cut sparsifier of G using $\tilde{O}(\alpha^{-1} n\sqrt{w})$ cut queries. It can be shown that the contraction resulting from the procedure has at most $\tilde{O}(\alpha^{-1} n\sqrt{w})$ edges, and hence H is indeed sparser than G. We will eventually set $\alpha = n^{-1/4}$ and $w = n$, to obtain a sparsifier with $\tilde{O}(n^{7/4})$ edges. The main technical challenge in the algorithm is to find an expander decomposition of G, which we achieve using $\tilde{O}(n)$ cut queries. Afterwards, the friendly sparsifier is obtained by applying a contraction defined by the expander

[3] This bound is not written there explicitly. In fact, the bound stated in [ASW25] is $\tilde{O}(n^{5/3} \cdot W)$, where W is the maximum edge weight in G, which is only weaker because the value of a minimum s, t-cut is at most nW.

[4] Another relaxation of isolating cuts, aimed at solving global minimum cut, was introduced in [ASW25].

[5] A slight difference is that we consider a general parameter $\alpha = \alpha(n)$ instead of fixing $\alpha = 0.4$.

decomposition. We present an algorithm for constructing a friendly cut sparsifier in the following lemma, whose proof appears in Section 6.

LEMMA 1.7 (Friendly Cut Sparsifier). *Given cut-query access to an unweighted graph G on n vertices and parameters α and w, one can recover all the edges of a friendly (α, w)-cut sparsifier of G using $\tilde{O}(\alpha^{-1} n\sqrt{w})$ cut queries. The algorithm is randomized and succeeds with probability $1 - 1/\operatorname{poly}(n)$.*

The primary challenge in proving Lemma 1.5 is to handle unfriendly cuts. Our second technical contribution addresses this by introducing a new randomized contraction procedure, called *(τ, F)-star contraction*, which aims to contract most of the edges in the graph while preserving an unfriendly minimum s,t-cut for $s,t \in F$ (with constant probability, separately for each pair). Previous work used similar procedures for the easier task of finding a global minimum cut [GNT20, AEG+22, KK25], and their main idea is that the probability of preserving a fixed cut $C \subset V$ is approximately $\prod_{v \in V} \left(1 - \mathrm{cr}_C(v)/\deg(v)\right)$, so for instance if only $O(1)$ vertices have small friendliness ratio, then the cut C is preserved with rather large probability.[6] Unfortunately, a minimum s,t-cut might be $o(1)$-friendly and preserved with low probability $o(1)$.

Our key insight is that if $C \subseteq V$ is a minimum s,t-cut, then at most one vertex (in fact either s or t) is α-unfriendly, i.e., has friendliness ratio below α, and thus by excluding it from the contraction we can leverage the machinery developed previously for global minimum cut. A further complication is that the argument why (in effect) only $O(1)$ vertices have low friendliness ratio assumes that $|E(C, V \setminus C)|/\min_{v \in V} \deg(v) = O(1)$. This assumption clearly holds for a global minimum cut, however the minimum s,t-cut value can be much larger than the minimum degree. To overcome this, we show that if the contraction excludes vertices of degree $o(|E(C, V \setminus C)|)$, then again only $O(1)$ vertices have low friendliness ratio. Unfortunately, this poses another challenge, because now edges incident to the low-degree vertices are not contracted, and the number of such edges might be $\Omega(n \cdot |E(C, V \setminus C)|) = \Omega(n^2)$. Ultimately, we prove that one need only consider the $O(\alpha \cdot |E(C, V \setminus C)|)$ edges of the cut that are not incident to the α-unfriendly vertex, which we indeed exclude from the contraction due to its low friendliness ratio as discussed above. Therefore, we can indeed set the degree threshold (to exclude contraction) at $\alpha \cdot |E(C, V \setminus C)|$, and again argue that only $O(1)$ vertices have low friendliness ratio.

Our contraction procedure is a variant of the τ-star contraction from [AEG+22, KK25], which is defined as follows. Let $H = \{v \in V \mid \deg(v) \geq \tau\}$ be the set of high-degree vertices, and sample a set $P \subseteq V$ of so-called *star vertices*, by including each vertex $v \in V$ independently with probability $p = \Theta(\log n/\tau)$. Then, for each high-degree non-star vertex $u \in H \setminus P$, sample an edge uniformly from $E(u, P)$ and contract it (unless $E(u, P) = \emptyset$). It can be shown that the resulting graph G' preserves a fixed global minimum cut with constant probability, and has $\tilde{O}((n/\tau)^2 + n\tau)$ edges with high probability [KK25]. Our variant introduces an additional feature, a prescribed set of "forbidden" vertices $F \subseteq V$ that are always excluded from H.

DEFINITION 1.8. (τ, F)-star contraction *is a procedure defined for an unweighted graph $G = (V, E)$ on n vertices, a threshold $\tau \in [n]$, and a vertex subset $F \subseteq V$, and it works as follows:*

1. *let $H = \{v \in V \setminus F \mid \deg(v) \geq \tau\}$ be the set of high-degree vertices not in F;*
2. *sample $P \subseteq V$ by including each $v \in V$ independently with probability $p = \Theta(\log n/\tau)$;*
3. *for each $u \in H \setminus P$, sample an edge uniformly from $E(u, P)$ and contract it.*

To solve the weak isolating-cuts problem, we apply (τ, F)-star contraction with parameters $\tau = \Theta(n\alpha)$ and $F = R$ (recall R is given in the input of the isolating-cuts problem). We now describe at a high level why this contraction preserves with constant probability a minimum s,t-cut for $s,t \in F$. Consider such a cut $C \subseteq V$ that it is α-unfriendly, and observe that our choice of τ implies $|E(C, V \setminus C)| \leq \min\{\deg(s), \deg(t)\} \leq n \leq \tau/(100\alpha)$. We wish to bound the probability that the cut is not preserved, i.e., at least one of its edges is contracted. Each vertex $v \in H \setminus P$ samples one incident edge to be contracted, and the probability that this edge belongs to the cut $E(C, V \setminus C)$ is $\mathrm{cr}_P(v)/d_P(v)$, where $\mathrm{cr}_P(v) \coloneqq |E(C, V \setminus C) \cap E(v, P)|$, and $d_P \coloneqq |E(v, P)|$. Therefore,

$$\Pr\left[\text{the cut } C \text{ is preserved} \mid P\right] \geq \prod_{v \in H} \left(1 - \frac{\mathrm{cr}_P(v)}{d_P(v)}\right),$$

[6] For a global minimum cut, the worst-case scenario for that probability can be shown to be when $O(1)$ vertices have friendliness ratio $1/2$ and the rest have friendliness ratio 1.

where by convention $\mathrm{cr}_P(v)/d_P(v) = 0$ whenever $d_P(v) = 0$. We bound this expression using the following proposition.

PROPOSITION 1.9 (Proposition 4.1 in [AEG+22]). *For integer $n \geq 1$ and $0 \leq a < 1 \leq b$, define*

$$\begin{aligned} F(a,b) := \min \quad & \prod_{i\in[n]} (1-x_i) \\ \textit{subject to} \quad & \sum_{i\in[n]} x_i = b, \\ & 0 \leq x_i \leq a, \qquad \forall i \in [n]. \end{aligned}$$

Then $F(a,b) \geq (1-a)^{\lceil b/a \rceil}$.

To apply this proposition we need upper bounds that hold with high constant probability for $\max_{v\in H} \mathrm{cr}_P(v)/d_P(v)$ and $\sum_{v\in H} \mathrm{cr}_P(v)/d_P(v)$. Recall that at most one vertex (in fact either s or t) is α-unfriendly, and assume without loss of generality it is s. We can further show (we skip the argument in this overview) that $\mathbb{E}_P\left[\mathrm{cr}_P(v)/d_P(v) \mid s \notin P\right] \leq 2\overline{\mathrm{cr}}_{-s}(v)/\deg(v)$, where $\overline{\mathrm{cr}}_{-s}(v)$ is the number of cut edges incident on v not including the edge (v,s) if it exists.

Before providing those upper bounds, let us argue that with high probability no edge incident to s is contracted. The probability that s is sampled into P is $p = \Theta(\log n/\tau) = o(1)$, where the last equality holds for a large enough τ, and we thus assume henceforth that $s \notin P$ (otherwise the algorithm fails). Observe that only edges in $E(H,P)$ can be contracted, and because s is neither in P nor in H (recall $s \in F$), it follows that no edge incident to s is contracted, as claimed.

Let us also bound $\sum_{v\in H} \overline{\mathrm{cr}}_{-s}(v)$, which will be used shortly. Since s is α-unfriendly and C is a minimum s,t-cut, we have $\mathrm{cr}(s) > (1-\alpha)\cdot\deg(s) \geq (1-\alpha)\cdot|E(C, V\setminus C)|$. Therefore, the number of edges in the cut C that are not incident to s is at most $\alpha\cdot|E(C,V\setminus C)|$, and furthermore $\sum_{v\in H}\overline{\mathrm{cr}}_{-s}(v) \leq 2\alpha\cdot|E(C,V\setminus C)|$ because the sum counts each of these edges at most twice. Altogether,

$$\mathop{\mathbb{E}}_{P}\left[\sum_{v\in H}\frac{\mathrm{cr}_P(v)}{d_P(v)}\right] \leq 2\sum_{v\in H}\frac{\overline{\mathrm{cr}}_{-s}(v)}{\deg(v)} \leq \frac{4\alpha\cdot|E(C,V\setminus C)|}{\tau} \leq \frac{1}{25},$$

where the first inequality uses that $\deg(v) \geq \tau$ for every $v \in H$, and the last one uses $|E(C,V\setminus C)| \leq \tau/(100\alpha)$ established above. Moreover, the summands above are non-negative, hence we can bound their maximum by their sum and obtain $\mathbb{E}_P\left[\max_{v\in H}\mathrm{cr}_P(v)/d_P(v)\right] \leq 1/25$. We can thus conclude, by Markov's inequality, that with constant probability these two quantities are indeed bounded.

We can therefore use Proposition 1.9 to conclude that the cut C is preserved with constant probability. Using arguments similar to the τ-star contraction, the number of edges in the contracted graph is bounded, with high probability, by $O((n/\tau)^2 + n\tau)$ (ignoring edges incident to F), and for our setting $\tau = \Theta(n\alpha)$ and $\alpha = n^{-1/4}$, this bound becomes $O(1/\alpha^2 + n^2\alpha) = \tilde{O}(n^{7/4})$. The following lemma states the guarantees of the (τ,F)-star contraction procedure more formally, and its proof appears in Section 4.

LEMMA 1.10 (Guarantees for (τ,F)-star contraction). *Let $G=(V,E)$ be an unweighted graph on n vertices, $F \subseteq V$ a subset of excluded vertices, and $\tau \geq \log^2 n$ a parameter. Now suppose $s,t \in F$ are such that $\min\{\deg(s),\deg(t)\} \leq \tau/(100\alpha)$ and that $C \subseteq V$ is a minimum s,t-cut that is α-unfriendly. Then a (τ,F)-star contraction of G produces a contraction G' such that with probability $2/5$ no edge of C is contracted, and with probability $1-\mathrm{poly}(n)$ the following hold:*

1. *The number of edges in G' is at most $O((n/\tau+|F|)^2 + n\tau)$.*
2. *The number of edges in G' that are incident to P is at most $O((n/\tau)^2 + n|F|/\tau)$.*

We conclude with a brief recap of the algorithm for solving the weak isolating-cuts problem on graph G and vertex subset R. The algorithm first sets $\alpha = n^{-1/4}$, $w = n$ and $\tau = \Theta(n\alpha) = \Theta(n^{3/4})$. It then constructs an (α,w)-friendly cut sparsifier H of G using Lemma 1.7, and uses its explicit description to find (without additional cut queries), all the minimum s,t-cuts (meaning for all $s,t \in R$) that are α-friendly. Next, to find the minimum s,t-cuts that are α-unfriendly, the algorithm applies the (τ,R)-star contraction procedure. By Lemma 1.10, each

minimum $(s, R \setminus s)$-cut that is also a minimum s,t-cut for some $t \in R$, is preserved in the contraction G' with constant probability, and this success probability is further amplified to $1 - 1/\operatorname{poly}(n)$ using $O(\log n)$ repetitions. Notice that since both steps are based on contractions, the algorithm's final output can be simply the minimum s,t-cut found throughout the steps and repetitions. The query complexity of both steps is $\tilde{O}(n^{7/4})$ by Lemmas 1.7 and 1.10.

1.2 Related Work

Gomory-Hu Tree construction Our algorithm for all-pairs minimum cut is based on constructing a Gomory-Hu tree [GH61]. It has long been known that a Gomory-Hu tree can be constructed using $n-1$ calls to minimum s,t-cut procedure [GH61]. In recent years, there has been significant progress on this problem, culminating in algorithms that make $O(\operatorname{polylog} n)$ calls to minimum s,t-cut procedures and require $\tilde{O}(m)$ time elsewhere [AKT20, LPS21, AKT21b, AKT21a, AKT22, AKL+22, ALPS23, AKL+25]. Over roughly the same time span, there has been significant progress on faster algorithms (in running time, not cut queries) for the minimum s,t-cut problem itself, and currently the best algorithm runs in $m^{1+o(1)}$ time [CKL+25].

Cut Query Algorithms The study of cut-query algorithms was initiated in [RSW18], motivated by the relation between cut queries and submodular minimization. and by now a few results are known for this model. For the global minimum cut problem, $O(n)$ randomized cut queries suffice in unweighted graphs [RSW18, AEG+22], matching the lower bound [GPRW20, LLSZ21], and $\tilde{O}(n)$ randomized cut queries suffice in weighted graphs [MN20]. For deterministic algorithms, $\tilde{O}(n^{5/3})$ cut queries suffice in unweighted graphs [ASW25]. For the easier problem of determining whether a graph is connected, it is known that $O(n)$ randomized queries suffice [CL23, LC24].

Yet another aspect of efficiency in this model is the round complexity, which refers to the number of parallel rounds of queries required to solve a problem. For example, for every $r \geq 1$, a global minimum cut can be found using $\tilde{O}(n^{1+1/r})$ cut queries in $O(r)$ rounds [KK25]. Additionally, there exists a one round randomized algorithm for connectivity that uses $\tilde{O}(n)$ queries [ACK21].

Meanwhile, for the minimum s,t-cut problem, the known algorithms use $\tilde{O}(n^{5/3})$ cut queries [RSW18, ASW25], leaving a substantial gap to the lower bound of $\Omega(n)$ queries. Another recent work has examined the problem of approximating the maximum cut using cut queries [PRW24].

1.3 Future Work

Gap Between All-Pairs Minimum Cut and Minimum s,t-Cut Existing algorithms for the all-pairs minimum cut problem achieve time complexity that is comparable, namely, within a polylogarithmic factor, of computing a single minimum s,t-cut [AKT20, LPS21, AKT21a, AKT22, AKL+22, ALPS23]. Our work leaves a substantial gap between these two problems in the cut-query model.

A promising direction for narrowing this gap is developing efficient algorithms for the weighted minimum s,t-cut problem in the cut-query model. An efficient algorithm for this problem would immediately yield an efficient solution to the isolating-cuts problem, and hence to the all-pairs minimum cut problem using existing techniques. Interestingly, there is a round-complexity lower bound for solving the minimum s,t-cut problem in weighted graphs, as an algorithm that runs in r rounds requires $\Omega(n^2/r^5)$ cut queries [ACK19].

Applicability to Other Computational Models One surprising aspect of cut-query algorithms is that they often translate to efficient algorithms in other computational models. For example, the global minimum cut algorithms of [RSW18, MN20] also yield efficient algorithms for finding a minimum cut in the streaming and sequential models. One reason for the wide applicability of cut-query algorithms is that they often use the following three-step approach: (1) apply structural analysis to contract the graph while preserving some desired property; (2) recover all the edges of the contracted graph; and (3) solve the problem on the smaller graph. This approach is often useful in other computational models since a contracted graph offers a succinct representation that facilitates faster computation. Our techniques for the all-pairs minimum cut employ this strategy and construct two succinct data structures, namely, a friendly cut sparsifier and a (τ, F)-star contraction, that together encode all minimum s,t-cuts using only $\tilde{O}(n^{7/4})$ edges. Hence, algorithms that construct these data structures efficiently in other computational models, such as dynamic, streaming, or parallel models, may yield new algorithms for the minimum s,t-cut problem in these settings.

2 Preliminaries

Cut Query Primitives Throughout the proof we use the following cut query primitives to find the weight of edges between two sets of vertices, $w(E(S,T)) \coloneqq \sum_{e \in E} w_e$. Notice that when the graph is unweighted then

$w(E(S,T)) = |E(S,T)|$.

CLAIM 2.1. *Let $S, T \subseteq V$ be two disjoint sets. It is possible to find $w(E(S,T))$ using $O(1)$ non-adaptive cut queries.*

Proof. Observe that $\text{cut}_G(S \cup T) = \text{cut}_G(S) + \text{cut}_G(T) - 2w(E(S,T))$. Hence, we can find $w(E(S,T))$ using three queries, one for each of $\text{cut}_G(S)$, $\text{cut}_G(T)$, and $\text{cut}_G(S \cup T)$. ∎

CLAIM 2.2. *Given an unweighted graph G on n vertices, and two disjoint sets $S, T \subseteq V$, one can sample an edge uniformly from $E(S,T)$ using $O(\log n)$ cut queries.*

Proof. At each stage the algorithm randomly partitions the vertex set T into two sets of roughly equal size T_1, T_2, and then find $|E(S,T_i)|$ using $O(1)$ cut queries by Claim 2.1. It then recurses on either T_1 or T_2 with probability proportional to $|E(S,T_i)|$, i.e. T_1 is chosen with probability $|E(S,T_1)|/(|E(S,T_1)|+|E(S,T_2)|)$. Notice that this process terminates in $O(\log n)$ rounds and yields an endpoint $t \in T$ of the edge, but does not recover a corresponding endpoint $s \in S$. To find the other endpoint, the algorithm repeats the process, but now partitioning S to find an edge in $E(S,t)$. It is straightforward to see that this indeed yields a uniform edge of $E(S,T)$ in $O(\log n)$ cut queries. This concludes the proof. ∎

Edge Perturbation Throughout the paper, we use a perturbation technique to ensure that the minimum s,t-cuts in the graph are unique. We do this by adding an edge set $\tilde{E}$ of all possible edges $(u,v) \in \binom{V}{2}$, where each edge gets a random weight $w(u,v)$ sampled uniformly from $\{1/n^{10}, \ldots, n^7/n^{10}\}$. Since those edges are defined externally, and are not a part of the graph, we can access them freely without having to use any cut queries. Note that this is slightly different from the perturbation technique used in previous work, where the perturbation is applied only to edges that exist in the graph. This is impossible to use in the cut-query model, as algorithms do not have access to the edges of the graph, and hence they cannot perturb existing edges. However, our method adds noise more broadly, and hence only increases the probability that the minimum cuts in the perturbed graph are unique.

CLAIM 2.3 ([BENW16, AKT20, AKT21a]). *Given an unweighted graph $G = (V,E)$ with n vertices, if one adds a random perturbation sampled uniformly from $\{1/n^{10}, \ldots, n^7/n^{10}\}$ to all edges in G, then with probability at least $1 - n^{-3}$ every minimum s,t-cut in the perturbed graph $\tilde{G} = (V, E \cup \tilde{E})$ is unique.*

COROLLARY 2.4. *Let $\tilde{E} = \binom{n}{2}$ be the set of all possible edges in G, each with weight sampled uniformly from $\{1/n^{10}, \ldots, n^7/n^{10}\}$. Then, with probability at least $1 - n^{-3}$ every minimum s,t-cut in the perturbed graph $\tilde{G} = (V, E \cup \tilde{E})$ is unique.*

We note that in general we perform most algorithms directly on the graph G, and then add the perturbation edges $\tilde{E}$ only for computing the minimum s,t-cuts. Throughout the paper, when we write that an algorithm is given a perturbation edge set $\tilde{E}$, we assume that the edges guarantee unique minimum s,t-cuts and that the algorithm fails otherwise. Finally, note that given a contracted graph $G' = (V', E')$ of G and a cut $S \subseteq V'$ we have that $\text{cut}_{\tilde{G}'}(S) = \text{cut}_{\tilde{G}}(S)$, where $\tilde{G}, \tilde{G}'$ are the perturbed versions of G, G' respectively. Hence, if no edge of the unique minimum s,t-cut in $\tilde{G}$, $C \subseteq V$, is contracted in $\tilde{G}'$ then C is also the unique minimum s,t-cut in $\tilde{G}'$.

Cut Sparsification Our algorithm uses two types of cut sparsifiers. The first, the Nagamochi-Ibaraki (henceforth NI) sparsifier [NI92] is used to handle small cuts. Informally, this sparsifier is a subgraph that preserves the value of all cuts of value at most k.

DEFINITION 2.5. *A k-*NI sparsifier *of an unweighted graph $G = (V, E_G)$ with parameter $k \in [n]$ is a subgraph $H = (V, E_H)$ that satisfies*

$$\forall C \subseteq V, \qquad \min\{\text{cut}_G(C), k\} = \min\{\text{cut}_H(C), k\}.$$

LEMMA 2.6 (Lemma 2.1 of [NI92]). *Let $G = (V,E)$ be an unweighted graph with n vertices, and for all $k \in [n]$ let T_k be a maximal spanning forest of $G \setminus (\cup_{j<k} T_j)$. Then, $H = (V, \cup_{j=1}^{k} T_j)$ is a k-NI sparsifier of G.*

It is known that a k-NI sparsifier can be computed in $O(m)$ time for every $k \in [n]$ [NI92]. We show how to construct such a sparsifier using $\tilde{O}(kn)$ cut queries.

CLAIM 2.7. *Given an unweighted graph $G = (V,E)$ with n vertices, one can construct a k-NI sparsifier H of G using $\tilde{O}(kn)$ cut queries.*

Proof. We show how to construct a maximal spanning forest T of a given graph G using $\tilde{O}(n)$ cut queries. Then, removing the edges in T from G and repeating the process k times yields a Nagamochi-Ibaraki sparsifier that preserves all cuts of value at most k using $\tilde{O}(kn)$ cut queries.

To construct a maximal spanning forest T of G, use the following procedure. Begin by initializing T to have no edges. At each step pick some connected component C of T, find some edge $e \in E(C, V \setminus C)$ using Claim 2.2, and add it to T. Notice that the number of connected components in T decreases by one in each iteration, and hence the procedure terminates after at most $n-1$ iterations. This concludes the proof of the claim. ∎

The second type of cut sparsifier we use is defined as follows.

DEFINITION 2.8. *A* $(1 \pm \epsilon)$-cut sparsifier *a graph* G *is a weighted subgraph* H *that satisfies*

$$\forall S \subseteq V, \qquad (1-\epsilon) \cdot \mathrm{cut}_G(S) \leq \mathrm{cut}_H(S) \leq (1+\epsilon) \cdot \mathrm{cut}_G(S).$$

THEOREM 2.9 ([RSW18]). *Given an unweighted graph* $G = (V, E)$ *with* n *vertices and quality parameter* $\epsilon \in (0, 1)$*, one can construct a* $(1+\epsilon)$*-cut sparsifier* H *of* G *using* $\tilde{O}(\epsilon^{-2} n)$ *cut queries. The algorithm is randomized and succeeds with probability* $1 - \mathrm{poly}(n)$*.*

3 Isolating Cuts

In this section we prove a slightly extended version of Lemma 1.5, stated as follows.

LEMMA 3.1. *Given cut-query access to an unweighted graph* $G = (V, E)$ *on* n *vertices, a set of vertices* $R \subseteq V$ *with maximum degree* $d = \max_{v \in R} d_G(v)$*, and a set of perturbation edges* $\tilde{E}$ *that guarantee unique minimum* s, t*-cuts, one can solve the weak isolating-cuts problem using* $\tilde{O}(\min\{nd, n^{7/4}\})$ *cut queries. The algorithm is randomized and succeeds with probability* $1 - 1/\mathrm{poly}(n)$*.*

Furthermore, if the algorithm is given in addition all the edges incident to vertices of degree at most $100n^{3/4}$*, an* $(n^{-1/4}, n)$*-friendly cut sparsifier, and an* $n^{3/4}$*-NI sparsifier, then it can solve the weak isolating-cuts problem using* $\tilde{O}(n^{1/2} + |R| n^{1/4})$ *cut queries.*

The proof requires the following claim, whose proof appears at the end of this section.

CLAIM 3.2. *Let* $G = (V, E)$ *be a perturbed unweighted graph,* $R \subseteq V$ *be a set of vertices, and* $G' = (V, E \setminus (R \times R))$*. For* $v \in R$*, denote by* S_v^G *the minimum* $(v, R \setminus v)$*-cut in* G*. Then* $S_v^G = S_v^{G'}$*.*

Proof of Lemma 3.1. The algorithm for finding the weak isolating cuts is as follows. If $d \leq 100n^{3/4}$, known by a preliminary round of $O(n)$ queries, solve the (stronger) isolating-cuts problem by constructing a d-NI sparsifier of G using Claim 2.7, perturb it with $\tilde{E}$, and then find each minimum $(v, R \setminus v)$-cut in the perturbed sparsifier. Otherwise, construct an $(n^{-1/4}, n)$-friendly cut sparsifier H of G using Lemma 1.7, and let $\tilde{H}$ be the perturbed version of H with added edges $\tilde{E}$. For each vertex $v \in R$, let $S_v^f \subseteq V$ be the minimum $(v, R \setminus v)$-cut in the perturbed sparsifier $\tilde{H}$.

Then, the algorithm performs $r = O(\log n)$ repetitions of $(100n^{3/4}, R)$-star contraction on G to obtain graphs $\{G_i\}_{i=1}^r$. Denote by $G_i' = (V, E_i')$ the graph obtained from G_i by removing all edges internal to R. We now explain how to recover the edges of G_i'. Given two vertex sets $S, T \subseteq V$ one can recover a single edge from $E(S, T)$ using $O(\log n)$ cut queries by Claim 2.2. Therefore, the entire edge set of the graph of G_i' can be recovered using $\tilde{O}(|E_i'|)$ cut queries, by iteratively recovering the sets $E(R, V \setminus R)$ and $E(V \setminus R, V \setminus R)$ one edge at a time. It then perturbs each G_i' to get a graph $\tilde{G}_i'$ with unique minimum s, t-cuts, and sets $S_v^i \subseteq V$ to be the minimum $(v, R \setminus v)$-cut in $\tilde{G}_i'$. Finally, the algorithm returns $S_v \leftarrow \arg\min_{A \in \{S_v^f\} \cup \{S_v^i\}_i} \mathrm{cut}_G(A)$.

We now argue that the algorithm returns every minimum $(v, R \setminus v)$-cut that is also a minimum v, u-cut for some $u \in R$. Fix one such cut $C \subseteq V$. Begin by noting that $|E_H(C, V \setminus C)|, |E_{G_i}(C, V \setminus C)| \geq |E_G(C, V \setminus C)|$ since these graphs are contractions of G, therefore it suffices to argue that $C \in \{S_v^f\} \cup \{S_v^i\}_{i=1}^r$ with high probability. If C is $n^{-1/4}$-friendly then the algorithm will find it in H. Otherwise, i.e. C is $n^{-1/4}$-unfriendly, it is preserved in G_i with constant probability by Lemma 1.10 since $\tau/(100\alpha) = n \geq \deg(s), \deg(t)$ where the equality is from $\tau = 100n^{3/4}, \alpha = n^{-1/4}$. Repeating this $O(\log n)$ times, we find that with probability at least $1 - n^{-10}$, C is preserved in at least one of the graphs G_i. Using a union bound over all vertices $w \in R$, the probability that at least one of the graphs G_i preserves the minimum $(w, R \setminus w)$-cut, if it is also a minimum w, u-cut for some $u \in R$, is at least $1 - n^{-10}|R| \geq 1 - n^{-9}$. Finally, by Claim 3.2 the minimum $(v, R \setminus v)$-cut in $\tilde{G}_i$ is equal to the minimum $(v, R \setminus v)$-cut in $\tilde{G}_i'$ and hence we can find it in the perturbed graph $\tilde{G}_i'$. Therefore, using a union bound with the probability that the construction of the friendly sparsifier succeeds, the algorithm returns the minimum $(v, R \setminus v)$-cut with probability at least $1 - 1/\mathrm{poly}(n)$.

It remains to analyze the query complexity of the algorithm. If $d \leq 100n^{3/4}$, then constructing the NI sparsifier uses $\tilde{O}(dn)$ cut queries by Claim 2.7. Otherwise, recovering the $(n^{-1/4}, n)$-friendly cut sparsifier H takes $O((1/n^{-1/4})n\sqrt{n}) = O(n^{7/4})$ cut queries by Lemma 1.7. Then, performing the star contraction takes $\tilde{O}(n)$ cut queries since each vertex samples at most one neighboring vertex. To bound the number of queries needed to recover the edges of G'_i, notice that each G'_i has at most $O\Big((n^{1/2} + |R|n^{1/4}) + n^{7/4}\Big)$ edges by Lemma 1.10. Therefore, recovering each G'_i takes $\tilde{O}(n^{7/4})$ cut queries using Claim 2.2. Hence, the overall query complexity of the algorithm is $\tilde{O}(\min\{nd, n^{7/4}\})$.

Finally, if the algorithm is given all edges incident to vertices of degree at most $n^{3/4}$, an $(n^{-1/4}, n)$-friendly cut sparsifier, and an $n^{3/4}$-NI sparsifier then it only needs to recover the edges incident to P. By Lemma 1.10, there are $\tilde{O}((n^2/\tau^2 + |R|(n/\tau))) = \tilde{O}(n^{1/2} + |R|n^{1/4})$ such edges, where we used $\tau = \Theta(n^{3/4})$. Since recovering a single edge takes $O(\log n)$ cut queries using Claim 2.2, the overall query complexity of the algorithm is $\tilde{O}(n^{1/2} + |R|n^{1/4})$. ∎

It remains to prove Claim 3.2.

Proof of Claim 3.2. Let C be a $(v, R \setminus v)$-cut in G. The edge set of the cut can be decomposed as

$$E(C, V \setminus C) = E(v, R) \cup E(v, V \setminus (R \cup C)) \cup E(C \setminus \{v\}, V \setminus C).$$

The second and third sets, $E(v, V \setminus (R \cup C))$ and $E(C \setminus \{v\}, V \setminus C)$ are the same in G and in G'. Furthermore, $E(v, R)$ does not depend on C and hence for every two $(v, R \setminus v)$-cuts C_1, C_2 we have $\mathrm{cut}_G(C_1) > \mathrm{cut}_G(C_2)$ if and only if $\mathrm{cut}_{G'}(C_1) > \mathrm{cut}_{G'}(C_2)$ since $E(v, R)$ contributes the same additive quantity to both sides. Therefore, the minimizer over all $(v, R \setminus v)$-cuts in G is the same as the minimizer over all $(v, R \setminus v)$-cuts in G' (and vice versa). ∎

4 (τ, F)-Star Contraction In this section we prove Lemma 1.10, which provides the guarantees of the (τ, F)-star contraction procedure. We will need the following claim, whose proof is similar to that of [AEG+22, Proposition 4.5] and appears in Subsection 4.1.

Claim 4.1. *Let $G = (V, E)$ be a simple n-vertex graph, $C \subseteq E$ be some non-trivial cut of G, and $U(C) \coloneqq \{v \in V \mid v \text{ is } \alpha\text{-unfriendly}\}$. Then, for any $v \in V$,*

$$\mathbb{E}\left[\mathrm{cr}_P(v)/d_P(v) \mid U(C) \cap P = \emptyset\right] \leq \frac{\overline{\mathrm{cr}}_{-U(C)}(v)}{\deg(v) - |U(C)|},$$

where $\overline{\mathrm{cr}}_{-U(C)}(v)$ is the number of edges incident to v that are in $E(C, V \setminus C)$ but not incident to any vertex in $U(C)$.

Proof of Lemma 1.10. Throughout the proof set $p \coloneqq 800 \cdot \log n/\tau$ and fix some α-unfriendly minimum s,t-cut $C \subseteq V$ of G such that $s, t \in F$ and $\min\{\deg(s), \deg(t)\} \geq \tau/(100\alpha)$. We begin by showing the correctness guarantee of the procedure, i.e. that no edges in $E(C, V \setminus C)$ is contracted with probability $2/5$. Denote a set P as good, if it satisfies $\max_{v \in H} \mathrm{cr}_P(v)/d_P(v) \leq 1/2$ and $\sum_{v \in H} \mathrm{cr}_P(v)/d_P(v) \leq 1/2$. If a set P is good, then,

$$\begin{aligned}\Pr\left[\text{the cut } C \text{ is preserved}\right] &\geq \Pr\left[\text{the cut } C \text{ is preserved} \mid P \text{ is good}\right] \cdot \Pr\left[P \text{ is good}\right] \\ &\geq \left(1 - \frac{1}{2}\right)^{\lceil (1/2)/(1/2) \rceil} \cdot \Pr\left[P \text{ is good}\right] = \frac{1}{2} \cdot \Pr\left[P \text{ is good}\right],\end{aligned}$$

where the first inequality is from the law of total probability, and the second is from Proposition 1.9. Therefore, we now focus on showing that P is good with high constant probability.

We start by bounding the probability that $\sum_{v \in H} \mathrm{cr}_P(v)/d_P(v) > 1/2$. Observe that since C is a minimum s,t-cut, at most one vertex (namely s or t) is α-unfriendly. Henceforth, assume without loss of generality that s is α-unfriendly. The probability that s is sampled into P is $p = \Theta(\log n/\tau) = \Theta(1/\log n)$. For the rest of the proof, assume that $\{s\} \not\subseteq P$, and that the algorithm fails otherwise. Notice that under this assumption no edge incident to s is contracted since the procedure only contracts edges in $H \times P$, and $s \notin P, H$ as H does not contain any vertex in F. By Claim 4.1 we have,

$$\mathop{\mathbb{E}}_{P}\left[\sum_{v \in H} \frac{\mathrm{cr}_P(v)}{d_P(v)} \middle| \{s\} \cap P = \emptyset\right] = \sum_{v \in H} \mathop{\mathbb{E}}_{P}\left[\frac{\mathrm{cr}_P(v)}{d_P(v)} \middle| \{s\} \cap P = \emptyset\right] \leq \sum_{v \in H} \frac{\overline{\mathrm{cr}}_{-s}(v)}{\deg(v) - \mathbb{1}_{\{(s,v) \in E\}}}. \tag{4.1}$$

Therefore, we need to bound $\sum_{v\in H}\overline{\mathrm{cr}}_{-s}(v)$. Notice that $\sum_{v\in H}\overline{\mathrm{cr}}_{-s}(v)$ is at most twice the number of edges in $E(C, V\setminus C)$ that are not incident to s. Observe that $|E(C, V\setminus C)| \le \min\{\deg(s), \deg(t)\}$ as C is a minimum s,t-cut and hence,

$$\sum_{v\in H}\overline{\mathrm{cr}}_{-s}(v) = 2(|E(C, V\setminus C)| - |E(\{s\}, V\setminus C)|) = 2(|E(C, V\setminus C)| - \mathrm{cr}(s))$$
$$\le 2(\deg(s) - (1-\alpha)\deg(s)) \le 2\alpha\deg(s) \le \tau/50,$$

where the first inequality is from the friendliness ratio of s and the last is from $\deg(s) \le \tau/(100\alpha)$ by the theorem statement. Furthermore, $\deg(v) \ge \tau$ for every $v \in H$, and therefore

$$\sum_{v\in H}\frac{\overline{\mathrm{cr}}_{-s}(v)}{\deg(v) - \mathbb{1}_{\{(s,v)\in E\}}} \le \frac{\tau/50}{\tau - 1} \le \frac{1}{20},$$

for every large enough n (and hence τ). Plugging back into Equation (4.1) we have that,

$$\mathop{\mathbb{E}}_{P}\left[\sum_{v\in H}\frac{\mathrm{cr}_P(v)}{d_P(v)}\middle|\{s\}\cap P = \emptyset\right] \le \sum_{v\in H}\frac{\overline{\mathrm{cr}}_{-s}(v)}{\deg(v) - \mathbb{1}_{\{(s,v)\in E\}}} \le \frac{1}{20},$$

and by Markov's inequality we have,

$$\Pr_{P}\left[\sum_{v\in H}\frac{\mathrm{cr}_P(v)}{d_P(v)} \ge \frac{1}{2}\right] \le \frac{1}{10}.$$

Using a union bound on the event that $\{s\} \not\subseteq P$ we have that with probability at least $4/5$, $\sum_{v\in H}\mathrm{cr}_P(v)/d_P(v) \le 1/2$. Notice that when this event occurs then $\max_{v\in H}\mathrm{cr}_P(v)/d_P(v) \le 1/2$ as well since every term in the sum is non-negative. Therefore, the set P is good with probability at least $4/5$. This concludes the proof of the correctness guarantee of the procedure, i.e., that no edge in $E(C, V\setminus C)$ is contracted with probability at least $2/5$.

To conclude the proof we provide the size guarantee. Partition the vertices of G' into P, F and $L = V\setminus(P\cup F)$. Recall that the algorithm samples each vertex into P independently with probability p. By the Chernoff bound (Theorem A.1), the probability that $d_P(v) \ge 0.9\deg(v)\cdot p$ is at least,

$$\Pr\left[d_P(v) \ge 720\deg(v)\log n/\tau\right] \ge 1 - \exp(-(1/10)^2\deg(v)800\log n/(2\tau)) \ge 1 - 1/n^4,$$

where the last inequality is from $\deg(v) \ge \tau$ as $v \in H$. Therefore, $d_P(v) \ge 0.9\cdot p\deg(v)$ for all $v \in H$ with probability at least $1 - 1/n^3$. Therefore, every vertex in H is contracted to some $p \in P$ with high probability. Hence, every vertex $v \in L$ that was not contracted has degree at most τ. For the rest of the proof we assume that this event holds.

The total number of edges in the graph is bounded by the number of edges internal to $P\cup F$ plus the total number of edges incident to L. Since, $|P| \le \tilde{O}(n/\tau)$ with high probability the number of edges internal to $P\cup F$ is at most $\tilde{O}\big((n/\tau + |F|)^2\big)$. Furthermore, since the degree of the vertices of L is bounded by τ, the total number of edges incident on L is at most $O(n\tau)$. Hence, the total number of edges in G' is at most $\tilde{O}\Big((n/\tau + |F|)^2 + n\tau\Big)$ with high probability. Discounting the edges in $F\times F$ we have that the number of edges in G' is at most $\tilde{O}\Big((n/\tau)^2 + |F|n/\tau + n\tau\Big)$ with high probability following the same argument. Similarly, discounting both the edges in $F\times F$ and the edges incident to low degree vertices, we have that the number of edges in G' is at most $\tilde{O}\Big((n/\tau)^2 + |F|n/\tau\Big)$ with high probability. ∎

4.1 Proof of Claim 4.1

We now prove Claim 4.1, using similar arguments to [AEG+22, Proposition 4.5].

Proof. Begin by noting that by convention when $d_P(v) = 0$ then the expectation is trivially 0 and the claim trivially holds. For the rest of the proof condition on $d_P(v) > 0$. Denote by $X_1, \ldots, X_{\mathrm{cr}(v)}, Y_1, \ldots, Y_{\deg(v)-\mathrm{cr}(v)}$ the random variables indicating if the edges incident to v are sampled into P. The variables X_i correspond to the edges incident to v that are in C, and the variables Y_i correspond to the edges incident to v that are not in C.

Let $X = \sum_i X_i$ and $Y = X + \sum_i Y_i$. Notice that the theorem statement is equivalent to showing that,

$$\mathbb{E}\left[X/Y \mid Y > 0, U(C) \cap P = \emptyset\right] \leq \frac{\overline{\mathrm{cr}}_{-U(C)}(v)}{\deg(v) - |U(C)|}.$$

Denote the number of edges incident to v that are also incident to $U(C)$ by a. Now fix $Y = b$ for some $0 < b \leq \deg(v) - a$ (since we condition on no vertex of $U(C)$ being sampled into P) and notice that,

$$\mathbb{E}\left[X/Y \mid Y = b, U(C) \cap P = \emptyset\right] = \frac{1}{b} \cdot \mathbb{E}\left[X \mid Y = b, U(C) \cap P = \emptyset\right].$$

Notice that the expectation is upper bounded by the case when all the edges of v that are incident to $U(C)$ are not part of the cut C, i.e. they all correspond to variables in $\{Y_i\}_i$ and not in $\{X_i\}_i$. In this case, that X is a hypergeometric random variable corresponding to making b draws from $\deg(v) - a$ edges, of which $\overline{\mathrm{cr}}_{-U(C)}(v)$ are marked. Therefore, $\mathbb{E}\left[X \mid Y = b, U(C) \cap P = \emptyset\right] \leq b \cdot \frac{\overline{\mathrm{cr}}_{-U(C)}(v)}{\deg(v)-a}$. Hence, for every b we have that, $\mathbb{E}\left[X/Y \mid Y = b, U(C) \cap P = \emptyset\right] \leq \frac{\overline{\mathrm{cr}}_{-U(C)}(v)}{\deg(v)-|U(C)|}$, where we used that $a \leq |U(C)|$.

Since this is true for all $b \geq 0$, we find that $\mathbb{E}\left[\mathrm{cr}_P(v)/d_P(v) \mid U(C) \cap P = \emptyset\right] \leq \frac{\overline{\mathrm{cr}}_{-U(C)}(v)}{\deg(v)-|U(C)|}$. This concludes the proof. ∎

5 Single-Source Minimum Cut In this section we prove Theorem 1.3, which is copied here for convenience.

LEMMA 5.1 (Single-Source Minimum Cut). *Given cut-query access to an unweighted graph G on n vertices, a set of perturbation edges $\tilde{E}$ that guarantee unique minimum s,t-cuts, and a partition of the vertex set $V_1, \ldots, V_k$ with pivots $\{p_i \in V_i\}_{i\in[k]}$, one can compute for every $i \in [k]$ and $v \in V_i$ the minimum p_i, v-cut S_v using $q(n) = \tilde{O}(n^{1.75})$ cut queries. The algorithm is randomized and succeeds with probability $1 - 1/\operatorname{poly}(n)$.*

The lemma is based on Algorithm 5.2. An important subroutine of the algorithm is Algorithm 5.1, which handles the case of 0.4-unfriendly minimum p_i, v-cuts. Throughout this section we denote the value of the minimum s,t-cut in the graph G by $\lambda_{s,t}(G)$. The following claim, whose proof is provided at the end of this section, details the guarantees of the unfriendly minimum cut algorithm.

CLAIM 5.2. *Given an unweighted graph $G = (V, E)$, a partition of the vertices into sets $V_1, \ldots, V_k$, and a set of pivots $\{p_i\}_{i=1}^k$ such that $p_i \in V_i$, for every $i \in [k], v \in V_i$ Algorithm 5.1 returns the minimum p_i, v-cut if it is 0.4-unfriendly, and returns some larger cut otherwise. The algorithm uses $\tilde{O}(n^{1.75})$ cut queries and succeeds with high probability.*

Algorithm 5.1 Unfriendly Minimum s,t-Cuts

Input: An unweighted graph $G = (V, E)$, vertex partition $V_1, \ldots, V_k$, a set of pivots $\{p_i\}_{i=1}^k$, and perturbation edges $\tilde{E}$.
procedure SINGLE-SOURCE-UNFRIENDLY-CUTS($G, \{V_i\}, \{p_i\}, \tilde{E}$)
  $\epsilon, \delta \leftarrow 1/100, r \leftarrow O(\log n)$
  $G' \leftarrow$ a quality $(1 \pm \epsilon)$-cut sparsifier of G using Theorem 2.9
  for $i \in [k]$ **do**
    $\tilde{c}(v) \leftarrow 2\lambda_{p_i,v}(G')/(1-\epsilon)$ for all $v \in V_i$
  for $i \in [k]$ **do**
    for $j \in [r]$ **do**
      $T_j \leftarrow \{v \in V_i \mid \tilde{c}(v) \geq (1+\delta)^j\} \cup \{p_i\}$
      $\{S_v^j\}_{v\in T_j} \leftarrow$ Isolating-Minimum-Cuts($G_i, T_j, \tilde{E}$)
    $S_v \leftarrow \arg\min_{S\in\{S_v^1,\ldots,S_v^r, V\setminus S_{p_i}^1,\ldots,V\setminus S_{p_i}^r\}} \mathrm{cut}_{\tilde{G}}(S)$ for all $v \in V_i$
  return $\{S_v\}_{v\in V}$.

Algorithm 5.2 Single-Source Minimum s, t-Cuts

Input: An unweighted graph $G = (V, E)$, vertex partition $V_1, \ldots, V_k$, a set of pivots $\{p_i\}_{i=1}^k$, and perturbation edges $\tilde{E}$.
procedure SINGLE-SOURCE-MIN-CUTS($G, \{V_i\}, \{p_i\}, \tilde{E}$)
  $H \leftarrow$ a friendly $(2/5, n)$-cut sparsifier of G
  $\tilde{H} \leftarrow$ perturb H with edges $\tilde{E}$ such that the minimum s, t-cuts are unique
  $\{S_v^f\}_{v \in V} \leftarrow$ compute the minimum p_i, v-cuts in $\tilde{H}$ for all $i \in [k]$ and $v \in V_i$
  $\{S_v^u\}_{v \in V} \leftarrow$ SINGLE-SOURCE-UNFRIENDLY-CUTS($G, \{V_i\}, \{p_i\}, \tilde{E}$)
  $\tilde{G} \leftarrow$ perturb G with edges $\tilde{E}$ such that the minimum s, t-cuts are unique
  for $v \in V$ **do**
    $S_v \leftarrow \arg\min_{S \in \{S_v^f, S_v^u\}} \mathrm{cut}_{\tilde{G}}(S)$
  return $\{S_v\}_{v \in V}$ ▷ return the cut value as well

Proof of Theorem 1.3. We begin by proving the correctness of Algorithm 5.2. Fix some vertex $v \in V_i$ and a pivot $p_i \in V_i$. Notice that by Lemma 1.7, we have $\lambda_{p_i,v}(H) \geq \lambda_{p_i,v}(G)$ and similarly by the guarantees of Claim 5.2, we have $\mathrm{cut}_G(S_v^u) \geq \lambda_{p_i,v}(G)$. Furthermore, the minimum p_i, v-cut S_v is either 0.4-friendly, in which case it is found in H by Lemma 1.7, or it is 0.4-unfriendly, in which case it is found by Claim 5.2. Hence, the minimum between the cuts S_v^f and S_v^u yields the unique minimum p_i, v-cut.

To analyze the query complexity of the algorithm, observe that constructing the $(0.4, n)$-friendly cut sparsifier H requires $\tilde{O}(n^{3/2})$ cut queries by Lemma 1.7. Then, perturbing H with edges $\tilde{E}$, and finding the minimum cuts requires no cut queries. By Claim 5.2, the unfriendly minimum cuts procedure uses $\tilde{O}(n^{1.75})$ cut queries. Therefore, the overall query complexity of the algorithm is $\tilde{O}(n^{1.75})$. Finally, the algorithm succeeds with high probability since both Claim 5.2 and Lemma 1.7 succeed with high probability, and the perturbation edges $\tilde{E}$ are chosen such that the minimum cuts in the perturbed graph are unique with high probability. ∎

5.1 Proof of Claim 5.2 The proof of the claim is similar to that of Theorem 1.4 in [AKT22], and requires the following lemmas.

LEMMA 5.3. *If the minimum p, v-cut, $S \subseteq V$ such that $v \in S$, is 0.4-unfriendly, and v is 0.4-unfriendly, then for all $v' \in V \setminus \{v\}$, $\lambda_{p,v'}(G) \leq 0.8 \cdot \lambda_{p,v}(G)$.*

LEMMA 5.4. *If the minimum p, v-cut, $S \subseteq V$ such that $v \in S$, is 0.4-unfriendly, and p is 0.4-unfriendly, then for all $v' \in (V \setminus S) \setminus \{p\}$, $\lambda_{p,v'}(G) \leq 0.8 \cdot \lambda_{p,v}(G)$.*

Proof of Claim 5.2. Throughout the proof, assume that the sparsifier construction succeeds and that the algorithm fails otherwise. Fix some vertex $v \in V_i \setminus \{p\}$ and a pivot $p_i \in V_i$. Begin by noting that the minimum p_i, v-cut returned by the algorithm is at least the value of the minimum p_i, v-cut in G by Lemma 3.1, even if the minimum p_i, v-cut is friendly. We show that if the minimum p_i, v-cut is 0.4-unfriendly, then the algorithm return it with high probability. Since by Lemma 3.1 the weak isolating-cuts procedure returns a cut that is at least the value of the minimum p_i, v, it suffices to show that Algorithm 5.1 finds the minimum p_i, v-cut in at least one of its iterations.

Denote the unique minimum p_i, v-cut by S_v. Notice that if $T_j \cap (V_i \setminus S_v) = \{p_i\}$ for some iteration j then the minimum $v, T_j \setminus v$ is the minimum p_i, v-cut, and similarly if $T_j \cap S_v = \{v\}$. Therefore, it remains to show that one of these events occurs in some iteration j. Let j^* be the index such that $(1+\delta)^{j^*} \leq \lambda_{p_i,v}(G') < (1+\delta)^{j^*+1}$. We now split the proof into two cases, depending on whether the p_i or v is 0.4-unfriendly. Notice that at most one of them can be 0.4-unfriendly since we consider a minimum p_i, v-cut.

Case 1: p_i is 0.4-unfriendly. In this case, by Lemma 5.4, for all $u \in (V_i \setminus S_v) \setminus \{p_i\}$, $\lambda_{p_i,u}(G) \leq 0.8 \cdot \lambda_{p_i,v}(G)$. Therefore, $\tilde{c}(u) \leq (1+3\epsilon)\lambda_{p_i,u}(G) \leq 0.8 \cdot (1+3\epsilon)\lambda_{p_i,v}(G) \leq \lambda_{p_i,v}/(1+\delta) \leq (1+\delta)^j$. Therefore, $u \notin T_j$ and we obtain the desired event that $T_j \cap (V_i \setminus S_v) = \{p_i\}$.

Case 2: v is 0.4-unfriendly. In this case, by Lemma 5.3, for all $u \in S_v \setminus \{v\}$, $\lambda_{p_i,u}(G) \leq 0.8 \cdot \lambda_{p_i,v}(G)$. Therefore, $\tilde{c}(u) \leq (1+3\epsilon)\lambda_{p_i,u}(G) \leq 0.8 \cdot (1+3\epsilon)\lambda_{p_i,v}(G) \leq \lambda_{p_i,v}/(1+\delta) \leq (1+\delta)^j$. Therefore, $u \notin T_j$ and we obtain the desired event that $T_j \cap S_v = \{v\}$.

Hence, the algorithm succeeds in finding the minimum p_i, v-cut as long as all the weak isolating-cuts instances succeed. Notice that the algorithm solves $k \cdot r = \tilde{O}(n)$ instances of the weak isolating-cuts problem, and since

each instance succeeds with high probability, the overall success probability of the algorithm is high.

To conclude the proof we analyze the query complexity of the algorithm. The first step of the algorithm is to construct a quality $(1 \pm \epsilon)$-cut sparsifier G' of G using Theorem 2.9, which requires $\tilde{O}(n)$ cut queries. Then, the algorithm solves $k \cdot r$ instances of the weak isolating-cuts problem. To do so efficiently, it recovers the following data from the graph once, and then passes it to all the instances. First, it recovers all the edges incident to vertices of degree at most $n^{3/4}$ using $\tilde{O}(n^{7/4})$ cut queries using Claim 2.2. The algorithm then constructs a $(n^{-1/4}, n)$-friendly cut sparsifier H of G using Lemma 1.7, which requires $\tilde{O}(n^{7/4})$ cut queries. Finally, it constructs an $n^{7/4}$-NI sparsifier of G using Claim 2.7, which requires $\tilde{O}(n^{7/4})$ cut queries. Hence, the query complexity of each weak isolating-cuts instance is at most $\tilde{O}(n^{1/2} + |T_j|n^{1/4}) \leq \tilde{O}(n^{1/2} + |V_i|n^{1/4})$. Since $\sum_i |V_i| = n$ and $k \leq n$, the total query complexity of all weak isolating-cut instances is at most $\tilde{O}(n^{3/2})$. Therefore, the overall query complexity of the algorithm is $\tilde{O}(n^{1.75})$. ∎

6 Friendly Cut Sparsifiers

In this section we prove Lemma 1.7, showing how to construct a friendly cut sparsifier using $\tilde{O}(\alpha^{-1}n\sqrt{w})$ cut queries. We begin by describing the expander decomposition procedure, which is the main technical tool used for constructing the friendly cut sparsifier.

6.1 Expander Decomposition

We begin by introducing the notion of expander decomposition and several useful facts about it. Let $G = (V, E)$ be some graph, and assume we are given a demand vector $\mathbf{d} \in \mathbb{R}_+^{|V|}$. Then, G is a said to be a $(\phi, \mathbf{d})$-*expander* if,

$$\forall S \subseteq V, \qquad \Phi_G^{\vec{d}}(S) \coloneqq \frac{\mathrm{cut}_G(S)}{\min(\mathbf{d}(S), \mathbf{d}(V \setminus S))} \geq \phi.$$

We use the following lemma for finding a decomposition of a graph into a collection of $(\phi, \mathbf{d})$-expanders, it is based on a reduction to the BalCutPrune procedure which has been used for constructing expander decomposition in the deterministic sequential setting [CGL+20, LS21]. The proof of the lemma is provided in Appendix C.

LEMMA 6.1. *Let $G = (V, E, w)$ be a weighted graph with $1 \leq w_e \leq U$ for all $e \in E$, $\mathbf{d} \in \mathbb{R}_+^{|V|}$ be a demand vector such that $\mathbf{d}(v) \in \{0\} \cup [1, U]$, and $\phi \in (0, 1)$ a parameter. There exists an algorithm that partitions V into sets $V_1, \ldots, V_k$ such that,*

1. *For every $i \in [k]$, $G[V_i]$ is a $(\phi, \mathbf{d}_{V_i})$-expander, where $\mathbf{d}_{V_i}$ is the vector $\mathbf{d}$ limited to the vertices in V_i.*
2. *The total weight of inter-cluster edges is at most $O(\phi \mathbf{d}(V) \log(mU))$.*

The algorithm uses $\tilde{O}(n)$ randomized cut queries and succeeds with probability $1 - 1/\operatorname{poly}(n)$.

6.2 Friendly Cut Sparsifier Construction

In this section we provide a construction of a friendly cut sparsifier. Recall that a cut $S \subseteq V$ is α-friendly if for every vertex $v \in V$ we have $\mathrm{cr}(v)/d(v) \leq 1 - \alpha$. Note that originally, friendly cut sparsifiers were defined with a fixed $\alpha = 0.4$ [AKT22]. We show how to implement the algorithm of [AKT22] using $O(\alpha^{-1}n\sqrt{w})$ cut queries. The proof requires the following lemma, whose proof follows similar lines to [AKT22] and is deferred to Appendix B.

Algorithm 6.1 Friendly Cut Sparsifier Construction

Input: An unweighted graph $G = (V, E)$
procedure FRIENDLY-CUT-SPARSIFIER(G, α)
  $V_1, \ldots, V_k \leftarrow (\phi, \mathbf{d})$-expander decomposition of G with $\phi = 0.01$ and $\mathbf{d}(v) = \phi^{-1}\sqrt{w}$ for all $v \in V$
  for $i \in [k]$ **do**
    $R_i \leftarrow \left\{ v \in V_i \mid d_G(v) < \max\{10\alpha^{-1}\sqrt{w}, 4\alpha \cdot |E_G(\{v\}, V \setminus H_i)|\} \right\}$
    $H_i' \leftarrow H_i \setminus R_i$
  $G' \leftarrow$ contract in G each H_i' into a single vertex v_i
  recover the edges of G' using $O(|E'|)$ cut queries iteratively using Claim 2.2.
  return G'.

LEMMA 6.2. *Given an unweighted graph G on n vertices, applying Algorithm 6.1 on G with parameter α and w, returns a friendly (α, w)-cut sparsifier G' of G with $\tilde{O}(\alpha^{-1}n\sqrt{w})$ edges.*

Proof of Lemma 1.7. Begin by constructing a $(\phi, \mathbf{d})$-expander decomposition of G by applying Theorem 6.1 with $\phi = 0.01$ and $\mathbf{d}(v) = \phi^{-1}\sqrt{w}$ for all $v \in V$. Notice that this requires $\tilde{O}(n)$ cut queries. Then, for each expander H_i in the decomposition, remove all vertices $v \in H_i$ such that $d_G(v) < \max\{10\alpha^{-1}\sqrt{w}, 4\alpha|E_G(\{v\}, V \setminus H_i)|\}$. This requires at most $O(n)$ cut queries since one can find $d_G(v), |E_G(\{v\}, T)|$ for every $v \in V$ and $T \subseteq V$ using $O(1)$ cut queries by Claim 2.1. Therefore, computing the shaved expanders H_i' requires at most $\tilde{O}(n)$ cut queries. To recover the graph $G' = (V', E')$, the algorithms recovers its entire edge set using $\tilde{O}(|E'|)$ cut queries by iteratively finding edges using Claim 2.2. Since the number of edges in G' is at most $\tilde{O}(\alpha^{-1}n\sqrt{w})$ by Theorem 6.2, we find that the total number of cut queries used by the algorithm is $\tilde{O}(\alpha^{-1}n\sqrt{w})$. Finally, the success probability of the algorithm is determined solely by the expander decomposition, which succeeds with probability $1 - n^{-10}$ by Theorem 6.1. This concludes the proof of Lemma 1.7. ∎

References

[ABK+02] Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. In *43rd Symposium on Foundations of Computer Science (FOCS 2002)*, page 197. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181943.

[ACK19] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*, pages 265–276. ACM, 2019. doi:10.1145/3313276.3316361.

[ACK21] Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph connectivity and single element recovery via linear and OR queries. In *29th Annual European Symposium on Algorithms, ESA 2021*, volume 204 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ESA.2021.7.

[AEG+22] Simon Apers, Yuval Efron, Pawel Gawrychowski, Troy Lee, Sagnik Mukhopadhyay, and Danupon Nanongkai. Cut query algorithms with star contraction. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 507–518. IEEE, 2022. doi:10.1109/FOCS54457.2022.00055.

[AKL+22] Amir Abboud, Robert Krauthgamer, Jason Li, Debmalya Panigrahi, Thatchaphol Saranurak, and Ohad Trabelsi. Breaking the cubic barrier for all-pairs max-flow: Gomory-Hu tree in nearly quadratic time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 884–895. IEEE, 2022. doi:10.1109/FOCS54457.2022.00088.

[AKL+25] Amir Abboud, Rasmus Kyng, Jason Li, Debmalya Panigrahi, Maximilian Probst Gutenberg, Thatchaphol Saranurak, Weixuan Yuan, and Wuwei Yuan. Deterministic almost-linear-time gomory-hu trees. *CoRR*, abs/2507.20354, 2025. doi:10.48550/ARXIV.2507.20354.

[AKT20] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Cut-equivalent trees are optimal for min-cut queries. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 105–118. IEEE, 2020. doi:10.1109/FOCS46700.2020.00019.

[AKT21a] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. APMF<APSP? Gomory-Hu Tree for Unweighted Graphs in Almost-Quadratic Time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 1135–1146. IEEE, 2021. doi:10.1109/FOCS52979.2021.00112.

[AKT21b] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Subcubic algorithms for Gomory-Hu tree in unweighted graphs. In *STOC '21*, pages 1725–1737. ACM, 2021. doi:10.1145/3406325.3451073.

[AKT22] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Friendly cut sparsifiers and faster Gomory-Hu trees. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 3630–3649. SIAM, 2022. doi:10.1137/1.9781611977073.143.

[ALPS23] Amir Abboud, Jason Li, Debmalya Panigrahi, and Thatchaphol Saranurak. All-pairs max-flow is no harder than single-pair max-flow: Gomory-Hu trees in almost-linear time. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 2204–2212. IEEE, 2023. doi:10.1109/FOCS57990.2023.00137.

[ASW25] Aditya Anand, Thatchaphol Saranurak, and Yunfan Wang. Deterministic edge connectivity and max flow using subquadratic cut queries. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025*, pages 124–142. SIAM, 2025. doi:10.1137/1.9781611978322.4.

[BENW16] Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *32nd International Symposium on Computational Geometry, SoCG 2016*, volume 51 of *LIPIcs*, pages 22:1–22:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.SOCG.2016.22.

[BGK05] Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In *Graph-Theoretic Concepts in Computer Science, 31st International Workshop, WG 2005*, volume 3787 of *Lecture Notes in Computer Science*, pages 16–27. Springer, 2005. doi:10.1007/11604686_2.

[BM11] Nader H. Bshouty and Hanna Mazzawi. Reconstructing weighted graphs with minimal query complexity. *Theor. Comput. Sci.*, 412(19):1782–1790, 2011. doi:10.1016/J.TCS.2010.12.055.

[CGJS22] Deeparnab Chakrabarty, Andrei Graur, Haotian Jiang, and Aaron Sidford. Improved lower bounds for submodular function minimization. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 245–254. IEEE, 2022. doi:10.1109/FOCS54457.2022.00030.

[CGL+20] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 1158–1167. IEEE, 2020. doi:10.1109/FOCS46700.2020.00111.

[CK08] Sung-Soon Choi and Jeong Han Kim. Optimal query complexity bounds for finding graphs. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 749–758. ACM, 2008. doi:10.1145/1374376.1374484.

[CKL+25] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *J. ACM*, 72(3):19:1–19:103, 2025. doi:10.1145/3728631.

[CL23] Deeparnab Chakrabarty and Hang Liao. A query algorithm for learning a spanning forest in weighted undirected graphs. In *International Conference on Algorithmic Learning Theory, ALT 2023*, volume 201 of *Proceedings of Machine Learning Research*, pages 259–274. PMLR, 2023. URL: https://proceedings.mlr.press/v201/chakrabarty23a.html.

[GH61] Ralph E. Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[GK98] Vladimir Grebinski and Gregory Kucherov. Reconstructing a hamiltonian cycle by querying the graph: Application to DNA physical mapping. *Discret. Appl. Math.*, 88(1-3):147–165, 1998. doi:10.1016/S0166-218X(98)00070-5.

[GNT20] Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster algorithms for edge connectivity via random 2-out contractions. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1260–1279. SIAM, 2020.

[GPRW20] Andrei Graur, Tristan Pollner, Vidhya Ramaswamy, and S. Matthew Weinberg. New query lower bounds for submodular function minimization. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, volume 151 of *LIPIcs*, pages 64:1–64:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ITCS.2020.64.

[Jia23] Haotian Jiang. Minimizing convex functions with rational minimizers. *J. ACM*, 70(1):5:1–5:27, 2023. doi:10.1145/3566050.

[KK25] Yotam Kenneth-Mordoch and Robert Krauthgamer. Cut-query algorithms with few rounds. In *33rd Annual European Symposium on Algorithms, ESA 2025*, volume 351 of *LIPIcs*, pages 100:1–100:14, 2025. doi:10.4230/LIPICS.ESA.2025.100.

[LC24] Hang Liao and Deeparnab Chakrabarty. Learning spanning forests optimally in weighted undirected graphs with CUT queries. In *International Conference on Algorithmic Learning Theory, ALT 2024*, volume 237 of *Proceedings of Machine Learning Research*, pages 785–807. PMLR, 2024. URL: https://proceedings.mlr.press/v237/liao24b.html.

[LLSZ21] Troy Lee, Tongyang Li, Miklos Santha, and Shengyu Zhang. On the cut dimension of a graph. In *36th Computational Complexity Conference, CCC 2021*, volume 200 of *LIPIcs*, pages 15:1–15:35. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.CCC.2021.15.

[LP20] Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 85–92. IEEE, 2020. doi:10.1109/FOCS46700.2020.00017.

[LPS21] Jason Li, Debmalya Panigrahi, and Thatchaphol Saranurak. A nearly optimal all-pairs min-cuts algorithm in simple graphs. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 1124–1134. IEEE, 2021. doi:10.1109/FOCS52979.2021.00111.

[LS21] Jason Li and Thatchaphol Saranurak. Deterministic weighted expander decomposition in almost-linear time. *CoRR*, abs/2106.01567, 2021. URL: https://arxiv.org/abs/2106.01567, arXiv:2106.01567.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 1049–1065. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.68.

[MN20] Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 496–509. ACM, 2020. doi:10.1145/3357713.3384334.

[MN21] Sagnik Mukhopadhyay and Danupon Nanongkai. A note on isolating cut lemma for submodular function minimization. *CoRR*, abs/2103.15724, 2021. URL: https://arxiv.org/abs/2103.15724, arXiv:2103.15724.

[NI92] Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(5&6):583–596, 1992. doi:10.1007/BF01758778.

[PRW24] Orestis Plevrakis, Seyoon Ragavan, and S. Matthew Weinberg. On the cut-query complexity of approximating Max-Cut. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024*, volume 297 of *LIPIcs*, pages 115:1–115:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.115.

[RSW18] Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018*, 2018. doi:10.4230/LIPICS.ITCS.2018.39.

A Concentration Inequalities

THEOREM A.1. *Let $X_1, \ldots, X_m \in [0, a]$ be independent random variables. For every $\delta \in [0, 1]$ and $\mu \geq \mathbb{E}\left[\sum_{i=1}^m X_i\right]$, we have*

$$\mathbb{P}\left[\left|\sum_{i=1}^m X_i - \mathbb{E}\left[\sum_{i=1}^m X_i\right]\right| \geq \delta\mu\right] \leq 2\exp\left(-\frac{\delta^2\mu}{3a}\right).$$

B Friendly Cut Sparsifier Proof In this section, we prove Theorem 6.2. We begin by restating the theorem for convenience.

LEMMA B.1. *Given an unweighted graph G on n vertices, applying Algorithm 6.1 on G with parameter α and w, returns a friendly (α, w)-cut sparsifier G' of G with $\tilde{O}(\alpha^{-1}n\sqrt{w})$ edges.*

The proof is based on the following two claims. Notice that combining these two claims immediately yields Theorem 6.2.

CLAIM B.2 (Equivalent of Claim 2.3 in [AKT22]). *Let $S \subseteq V$ be an α-friendly cut of G with at most w edges. Then, the output of Algorithm 6.1 preserves S (i.e. no edge of $E(S, V \setminus S)$ is contracted).*

CLAIM B.3 (Equivalent of Claim 2.4 in [AKT22]). *The number of edges in the output of Algorithm 6.1 is at most $\tilde{O}(\alpha^{-1}n\sqrt{w_j})$.*

Proof of Claim B.2. The proof follows the same lines as the proof of Claim 2.1 in [AKT22]. Notice that an edge $(x, y) \in E(S, V \setminus S)$ is contracted if and only if both endpoints x, y are contained in the same shaved expander H_i'. Let $L = H_i \cap S$ and $R = H_i \cap (V \setminus S)$ be the induced the cut on the original expander, before shaving. Assume without loss of generality that $x \in L, y \in R$. Since H_i is a $(\phi, \mathbf{d})$-expander, we have,

$$\frac{|E(L, R)|}{\min\{\mathbf{d}(L), \mathbf{d}(R)\}} \geq \phi.$$

Hence, $\min\{\mathbf{d}(L), \mathbf{d}(R)\} \leq \phi^{-1}w$ since $|E(L, R)| \leq \mathrm{cut}_G(S) \leq w$ by the theorem statement. We now lower bound $\mathbf{d}(L)$ ($\mathbf{d}(R)$ is bounded by a symmetric argument). Notice that since the cut S is friendly in G, we have $|E(\{x\}, S)| = \deg(x) - \mathrm{cr}(v) \geq \alpha \cdot \deg(x)$ since $\mathrm{cr}(x) \leq (1 - \alpha) \cdot \deg(x)$. Furthermore, since x was not shaved then $|E(\{x\}, V \setminus H_i)| \leq \alpha \cdot \deg(x)/4$ and,

$$|E(\{x\}, L)| = |E(\{x\}, S \cap H_i)| \geq |E(\{x\}, S)| - |E(\{x\}, V \setminus H_i)| \geq (\alpha - \alpha/4)\deg(x) \geq 5/2\deg(x)\sqrt{w}, \tag{B.1}$$

where the last inequality follows as $\deg(x) \geq 10\alpha^{-1}\sqrt{w}$ since x was not shaved. Notice that since G is a simple graph, we have that $|E(\{x\}, L)| \leq |L|$. By the definition of $\mathbf{d}$, we have $\mathbf{d}(L) = |L|\phi^{-1}\sqrt{w} \geq (5/2)\phi^{-1}w$, where the inequality is from Equation (B.1). The argument is symmetric for R, hence we have $\min\{\mathbf{d}(L), \mathbf{d}(R)\} \geq (5/2)\phi^{-1}w$ which leads to a contradiction. Therefore, no edge of $E(S, V \setminus S)$ is contracted, and the algorithm preserves the cut S. ∎

Proof of Claim B.3. The proof follows the same lines as the proof of Claim 2.2 in [AKT22]. Notice that G' has three types of edges,

1. The outer edges of the expander decomposition, of which there are at most $O(\phi\mathbf{d}(V)\log m) = \tilde{O}(n\sqrt{w})$ by Theorem 6.1 and the value of the demand vector.
2. The edges adjacent to vertices shaved because of their degree, i.e. those with $\deg(v) < 10\alpha^{-1}\sqrt{w}$, of which there are at most $O(\alpha^{-1}n\sqrt{w})$.
3. Edges that are incident to vertices that were shaved because at least $\alpha^{-1}\deg(v)/4$ of their edges went outside their expander. For every $v \in V$ let $d_{out}(v)$ be the cardinality of the edge set $E(\{v\}, V \setminus H)$, where H is the cluster to which v belongs. Furthermore, let $X \subseteq V$ be the set of the aforementioned shaved vertices, observe that the total number of inter-cluster edges is at most $\tilde{O}(n\sqrt{w}) = \sum_{v\in V} d_{out}(v)$ as explained above. Since for every $x \in X$ we have $d_{out}(x) \geq \alpha\deg(x)/4$, we find that the number of edges incident to X is at most $\tilde{O}(\alpha^{-1}n\sqrt{w})$.

Summing up the three types of edges, we find that the total number of edges in G' is at most $\tilde{O}(\alpha^{-1}n\sqrt{w})$. ∎

C Expander Decomposition

In this section we prove Theorem 6.1, which we restate here for convenience.

LEMMA C.1. *Let $G = (V, E, w)$ be a weighted graph with $1 \leq w_e \leq U$ for all $e \in E$, $\mathbf{d} \in \mathbb{R}_+^{|V|}$ be a demand vector such that $\mathbf{d}(v) \in \{0\} \cup [1, U]$, and $\phi \in (0, 1)$ a parameter. There exists an algorithm that partitions V into sets $V_1, \ldots, V_k$ such that,*

1. *For every $i \in [k]$, $G[V_i]$ is a $(\phi, \mathbf{d}_{V_i})$-expander, where $\mathbf{d}_{V_i}$ is the vector $\mathbf{d}$ limited to the vertices in V_i.*
2. *The total weight of inter-cluster edges is at most $O(\phi\mathbf{d}(V)\log(mU))$.*

The algorithm uses $\tilde{O}(n)$ randomized cut queries and succeeds with probability $1 - 1/\operatorname{poly}(n)$.

We use the BalCutPrune approach to find an expander decomposition of G.

DEFINITION C.2. *Given an undirected graph $G = (V, E)$, demands vector $\mathbf{d} \in \mathbb{R}_+^{|V|}$, sparsity parameter $\phi \in (0, 1]$, and approximation factor α, the goal of the α-approximate BalCutPrune problem is to find a partition (A, B) of V such that $w(E(A, B)) \leq \alpha\phi\min\{\mathbf{d}(A), \mathbf{d}(B)\}$ and, either:*

1. ***Cut:*** *$\mathbf{d}(A), \mathbf{d}(B) \geq \mathbf{d}(V)/3$; or*
2. ***Prune:*** *$\mathbf{d}(A) \geq \mathbf{d}(V)/2$ and $\Phi_{G[A]} \geq \phi$,*

where $\Phi_{G[A]}$ is the expansion of the induced subgraph $G[A]$ defined as,

$$\Phi_{G[A]} = \min_{S\subseteq A, 0<|S|<|A|} \frac{w(E(S, A \setminus S))}{\min \mathbf{d}(S), \mathbf{d}(A \setminus S)}$$

The main technical lemma needed for the proof of Theorem 6.1 is the following lemma, which shows how to solve the BalCutPrune problem using few cut queries.

LEMMA C.3. *Given an undirected graph $G = (V, E)$ on n vertices, a subset of vertices $W \subseteq V$, demand vector $\mathbf{d} \in \mathbb{R}_+^{|W|}$, and sparsity parameter $\phi \in (0, 1]$, one can solve the 2-approximate BalCutPrune on $G[W]$ using $\tilde{O}(|W|)$ randomized cut queries. The algorithm succeeds with probability $1 - n^{-10}$.*

Proof. Notice that one can simulate any cut query $S \subseteq W$ in $G[W]$ by $O(1)$ cut queries in G, since $\operatorname{cut}_{G[W]}(S) = |E(S, W\setminus S)|$ which can be recovered using $O(1)$ cut queries in G by Claim 2.1. Begin by constructing a quality $(1 \pm 1/2)$-cut sparsifier H of G, this requires $\tilde{O}(n)$ cut queries by Theorem 2.9. Notice that this allows

us to approximate the value $w(E(A,B))$ for any partition (A,B) of V up to factor 2. Hence, the algorithm can find all partitions (A,B) of V such that $w(E(A,B)) \leq 2 \cdot \phi \min\{\mathbf{d}(A), \mathbf{d}(B)\}$ and then return the first partition that satisfies either the **Cut** or **Prune** condition. Finally, notice that the only probabilistic part of the algorithm is the construction of the cut sparsifier, which succeeds with probability $1 - n^{-10}$ by Theorem 2.9. This concludes the proof of Lemma C.3. ∎

Proof of Theorem 6.1. The proof follows the line of the proof of Corollary 2.5 in [LS21]. Throughout the proof we maintain a collection $\mathcal{H}$ of disjoint subgraphs of G which are called clusters, the collection is partitioned into *active* clusters $\mathcal{H}^A$ and *inactive* clusters $\mathcal{H}^I$. We also maintain a set of edges E' which are outside all the clusters.

The algorithm for the decomposition is as follows. While $\mathcal{H}^A \neq \emptyset$, run the 2-approximate BalCutPrune on $G[H]$ with demands $\mathbf{d}_H$ and sparsity parameter ϕ for all $H \in \mathcal{H}^A$ in parallel. Fix some $H \in \mathcal{H}$ and let (A,B) be the partition returned by the BalCutPrune procedure. If $\mathbf{d}(A), \mathbf{d}(B) \geq \mathbf{d}(H)/3$ then replace H with A and B in $\mathcal{H}^A$. Otherwise, $\mathbf{d}(A) \geq \mathbf{d}(H)/2$ and $\Phi_{G[A]} \geq \phi$, hence the algorithm removes H from $\mathcal{H}^A$ and adds A to $\mathcal{H}^I$ and B to $\mathcal{H}^A$. Finally, it adds the edges $E(A,B)$ to E'.

Notice that in every iteration, $\mathbf{d}_H(H)$ is reduced by at least a constant factor, therefore the algorithm terminates after at most $O(\log mU)$ iterations. Notice that when the algorithm terminates, we are guaranteed that $\Phi_{G[H]} \geq \phi$ for every $H \in \mathcal{H}^I$. This satisfies the first condition of the theorem.

We now bound the total weight of inter-cluster edges. Notice that since the cuts found are ϕ-sparse, in every cluster H we have that $w(E_{G[H]}(A,B)) \leq 2\phi \min\{\mathbf{d}_H(A), \mathbf{d}_H(B)\} \leq 2\phi \mathbf{d}_H(H)$, and summing over all partitions we find that the total weight of inter-cluster edges is at most, $2\phi\mathbf{d}(V)$. Therefore, in $O(\log(mU))$ iterations the algorithm adds edges of total weight at most $O(\phi \mathbf{d}(V) \log(mU))$ to E'.

To conclude the proof, we analyze the query complexity of the algorithm. Notice that the query complexity of the 2-approximate BalCutPrune on a cluster H is $\tilde{O}(|H|)$ by Lemma C.3. Since the total number of vertices in all clusters is at most n, the total query complexity of the algorithm in a single iteration is $\tilde{O}(n)$ and the overall query complexity is $\tilde{O}(n \log(mU))$. Finally, the success probability of the algorithm is determined solely by the BalCutPrune procedure, and using a union bound over all iterations we find that the algorithm succeeds with probability $1 - n^{-10} \cdot \tilde{O}(n \log(mU)) = 1 - 1/\operatorname{poly}(n)$. ∎

D Gomory-Hu Algorithm In this section we provide a reduction from a single-source minimum cut procedure algorithm to computing the Gomory-Hu tree of a graph G, proving Theorem 1.2. We begin by restating the lemma for convenience.

LEMMA D.1. *Suppose that given cut-query access to an unweighted graph G on n vertices, a set of perturbation edges that guarantee unique minimum s,t-cuts, and a partition of the vertex set $V_1, \ldots, V_k$ with pivots $\{p_i \in V_i\}_{i \in [k]}$, one can return for every $i \in [k]$ and $v \in V_i$ the minimum p_i, v-cut S_v using $q(n)$ cut queries and with success probability $1 - \rho_F(n)$.*

Then, one can compute a Gomory-Hu tree of an input graph on n vertices using $\tilde{O}(q(n) + n^{1.5})$ cut queries. The algorithm is randomized and succeeds with probability $1 - O(\rho_F(n) \cdot \log^2 n) - 1/\operatorname{poly}(n)$.

The algorithm is based on the general framework of [AKT22, AKT21a]. The proof requires the following definitions.

DEFINITION D.2 (Partial k-Tree). *A k-partial tree of a graph $G = (V,E)$ is a partition tree of G, such that for every two sets $X, Y \subseteq V$ in the partition,*

- *For every $x_1, x_2 \in X$, the minimum x_1, x_2-cut is larger than k, and*
- *For every $x \in X, y \in Y$ the minimum x,y-cut in G is equal to the minimum X,Y-cut in T.*

DEFINITION D.3 (Auxiliary Graph). *Given a partition tree of T of a graph $G = (V,E)$ and a super-vertex $W \in T$, the auxiliary graph G_W is the graph obtained by contracting every connected component of $T \setminus W$ into a single vertex.*

We begin by presenting the algorithm of [AKT21a] and then explain how to implement it in the cut-query model.

Gomory-Hu Algorithm

1. Compute a partial k-tree T of G with $k = \sqrt{n}$. This is the initial Gomory-Hu tree that is refined throughout the algorithm.
2. For each super vertex V_i, do the following:
   (a) If $|V_i| = 1$, then stop and continue to the next super vertex.
   (b) Get an auxiliary graph G_i using the current tree T, then compute a perturbed version $\tilde{G}_i$ of G_i such that the minimum cuts are unique.
   (c) Pick a pivot $p \in V_i$ uniformly at random, and compute single-source minimum cuts S_v from p to all other vertices in V_i on the perturbed graph $\tilde{G}_i$.
   (d) Denote each vertex such that $|S_v \cap V_i| \leq |V_i|/2$ as good, and the rest as bad.
   (e) If there are less than $|V_i|/4$ good vertices return to step 2c and pick a new pivot (If this fails $20 \log n$ times then abort the algorithm). Otherwise, continue to the next step.
   (f) Refine T based on the cuts as follows. For each good vertex $u \in V_i$, assign it to the largest good cut S_v such that $u \in S_v$. Then, let $S_1, \ldots, S_r$ be the cuts to which at least one good vertex was assigned. Refine the tree by taking a Gomory-Hu step vertex V_i using each of the cuts above. This results in replacing V_i with a set of super vertices $\{S_1, \ldots, S_r, V_i \setminus (\cup_i S_i)\}$.
   (g) Recurse on all the new super vertices by going to step 2a.
3. Return the final tree T.

The following result states that the algorithm above computes a Gomory-Hu tree of G.

LEMMA D.4 ([AKT21a]). *The algorithm above returns a Gomory-Hu tree of G. Furthermore, the probability of failure in step 2e is at most n^{-10}.*

We now prove Theorem 1.2.

Proof of Theorem 1.2. We explain how to implement the algorithm above in the cut-query model. The first step is to compute a partial k-tree of G with $k = \sqrt{n}$. We do this by recovering a $2\sqrt{n}$-NI sparsifier of G using Claim 2.7 and then constructing a partial k-tree of the sparsifier using some algorithm. Notice that since the sparsifier captures every cut of size at most $\sqrt{n}$, it suffices for constructing a partial k-tree. Note that this takes $\tilde{O}(n^{1.5})$ cut queries by Claim 2.7.

To proceed, instead of recursively breaking each super vertex V_i until it becomes a singleton, run the algorithm inside all super vertices of a given level in parallel. To perform step 2c, sample uniformly a pivot $p_i \in V_i$ for all i in parallel. Then, call the single-source minimum cut algorithm in the theorem statement. Note that the single-source minimum cut algorithm solves the problem of computing the minimum p_i, v-cut for every $v \in V_i$ on G directly, and not on the auxiliary graphs $\{G_i\}_i$. However, by the structure of the Gomory-Hu tree we are guaranteed that those cuts are the same as the minimum p_i, v-cuts in the auxiliary graphs G_i.

For every partition V_i which passes the condition in step 2e, the algorithm stops and refines the tree. Notice that we do not need any additional cut queries for checking the condition or the refinement step, since the cuts S_v and their values were already computed in the previous step.

Notice that the algorithm terminates with depth $O(\log n)$ and runs for at most $O(\log n)$ times in each level, therefore we call the single-source minimum cut algorithm at most $O(\log^2 n)$ times. Hence, the total number of cut queries used by the algorithm is at most $\tilde{O}(n^{1.5} + q(n) \log^2 n)$, where $q(n)$ is the query complexity of the single-source minimum cut algorithm in the theorem statement. The success probability follows from Lemma D.4 and the fact that the single-source minimum cut algorithm succeeds with probability ρ_F. This concludes the proof of Theorem 1.2. ∎