OrderedCuts: A new approach for computing Gomory-Hu tree

Vladimir Kolmogorov
Institute of Science and Technology Austria (ISTA)
vnk@ist.ac.at

Abstract

The Gomory-Hu tree, or a cut tree, is a classic data structure that stores minimum \( s-t \) cuts of an undirected weighted graph for all pairs of nodes \( (s, t) \). We propose a new approach for computing the cut tree based on a reduction to the problem that we call OrderedCuts. Given a sequence of nodes \( s, v_1, \ldots, v_\ell \), its goal is to compute minimum \( \{s, v_1, \ldots, v_{i-1}\}-v_i \) cuts for all \( i \in [\ell] \). We show that the cut tree can be computed by \( \tilde{O}(1) \) calls to OrderedCuts. We also establish new results for OrderedCuts that may be of independent interest. First, we prove that all \( \ell \) cuts can be stored compactly with \( O(n) \) space in a data structure that we call an OC tree. We define a weaker version of this structure that we call depth-1 OC tree, and show that it is sufficient for constructing the cut tree. Second, we prove results that allow divide-and-conquer algorithms for computing OC tree. We argue that the existence of divide-and-conquer algorithms makes our new approach a good candidate for a practical implementation.

We study the problem of computing the Gomory-Hu (GH) tree \([GH61]\) (aka cut tree) in an undirected weighted graph \( G = (V, E, w) \) with non-negative weights. This is a weighted spanning tree \( \mathcal{T} \) on nodes \( V \) with the property that for any distinct \( s, t \in V \), a minimum \( s-t \) cut \( S \subseteq V \) in \( \mathcal{T} \) is also a minimum \( s-t \) cut in \( G \). It has numerous applications in various domains (see e.g. Section 1.4 in \([AKT21a]\)).

More than half a century ago Gomory and Hu \([GH61]\) showed that such tree exists and can be computed using \( n - 1 \) maximum flow computations and graph contractions. This bound has been improved upon only very recently: in a breakthrough result Abboud et al. showed in \([AKL+22]\) that the GH tree can be computed w.h.p. in time \( \tilde{O}(n^2) \). This very exciting development still leaves some open questions:

- Is further improvement possible, e.g. is there an \( O(m^{1+o(1)}) \) algorithm? (This is Open Question 1.4 in \([AKT21a]\)).
- Is there a Las-Vegas algorithm that computes a GH tree with probability one (rather than w.h.p.) faster than \( O(n) \) maxflows?
- What is the most efficient algorithm in practice?

We argue that a Las-Vegas algorithm may be preferable in practice to a Monte-Carlo algorithm even if one is satisfied with an answer that holds with high probability (and so the last two questions are connected). Indeed, in a Monte-Carlo algorithm one may need to set parameters such as the number of iterations to their worst-case bounds since one cannot observe directly whether an iteration succeeded or not. This is not the case for a Las Vegas algorithm, and so for practical instances the latter may perform much faster than the worst-case bound.

In an attempt to address these questions we investigate a new approach for constructing a GH tree based on the reduction to the problem that we call OrderedCuts. The latter is formulated
OrderedCuts

1 for each $i \in [\ell]$ compute $s$-$v_i$ cut $S_i$ with $v_i \in S_i$ s.t. $\text{cost}(S_i) \leq f(\{s, v_1, \ldots, v_{i-1}\}, v_i)$
2 for each $v \in V$ compute $\lambda(v) = \min_{i : \alpha \in S_i} \text{cost}(S_i)$, return $\lambda$

One possibility is to choose $S_i$ as a minimum $\{s, v_1, \ldots, v_{i-1}\}$-$v_i$ cut for each $i \in [\ell]$; we will refer to such version as strong OrderedCuts (and accordingly to the general version as weak OrderedCuts). Another option is to choose $S_i$ as a minimum $s$-$v_i$ cut; then the procedure would return vector $\lambda$ with $\lambda(v_i) = f(s, v_i)$. Thus, OrderedCuts$(s, v_1, \ldots, v_{\ell}; G)$ can be trivially implemented (in time $O(n + m)$) if we have an access to the Gomory-Hu tree for $G$. Our first result is the opposite direction given by the theorem below. To prove it, we used techniques from [AKT20] and [LP21].

**Theorem 1.** There exists a randomized (Las-Vegas) algorithm for computing GH tree for graph $G$ with expected complexity $O(1) \cdot (t_{OC}(n, m) + t_{MC}(n, m))$, where $t_{OC}(n, m)$ and $t_{MC}(n, m)$ are the complexities of OrderedCuts($\cdot$) and minimum $s$-$t$ cut computations respectively on a sequence of graphs $H_1, \ldots, H_k$ that have $O(n)$ nodes and $O(m)$ edges in total. $\Box$

We then focus on the strong version of OrderedCuts, and prove several new results for it (which we believe may be of independent interest).

First, we show that all cuts $S_1, \ldots, S_{\ell}$ can be stored compactly with $O(n)$ space using a data structure that we call an $\text{OC}$ tree. We also define the notion of a depth-1 $\text{OC}$ tree (or $\text{OC}_1$ tree), and prove that Theorem 1 still holds if the $\text{OC}$ tree oracle is replaced with an $\text{OC}_1$ tree oracle (at the expense of extra logarithmic factors). Note that the $\text{OC}_1$ tree can be trivially computed from the $\text{OC}$ tree by repeatedly removing leaves at depth two or larger, but could potentially be computed faster.

Second, we present lemmas that allow the use of divide-and-conquer strategies for computing $\text{OC}$ tree. One of the strategies can be formulated as follows (ignoring base cases): (1) split input sequence as $(s, v_1, \ldots, v_{\ell}) = \alpha \beta$ with $|\alpha| \approx |\beta|$; (2) solve recursively the problem for sequence $\alpha$ in $G$; (3) compute minimum $s$-$\alpha$ cut $(S, T)$ in $G$; (4) solve recursively the problem for sequence $s \beta$ in the graph obtained from $G$ by contracting $S$ to $s$; (5) combine the results. If, for example, the order of $(v_1, \ldots, v_{\ell})$ is a random permutation then we will have $|T| \leq \frac{1}{2} |V_G|$ with probability at least $\frac{1}{2}$ (since orderings $\alpha \beta$ and $\beta \alpha$ are equally likely), which leads to an improved complexity over a naive approach. $\Box$

**Theorem 2.** There exists an algorithm for solving strong OrderedCuts for sequence $(s, v_1, \ldots, v_{\ell})$ that has complexity of maximum computations on $O(n)$ graphs of size $(n, m)$ each. If the order of $v_1, \ldots, v_{\ell}$ is a uniformly random permutation then the expected total number of nodes and edges in these graphs is $(O(n^{1+\gamma}), O(n^\gamma m))$ where $\gamma = \log_2 1.5 = 0.584$.

The ordering used in the GH algorithm is not random, but satisfies $\lambda(v_1) \geq \ldots \geq \lambda(v_{\ell})$ where $\lambda(v)$ is the current (over-)estimation of $f(s, v)$. Thus, “heavier” nodes come first, which might give some hope to prove that the size of set $T$ is sufficiently small (if e.g. source node $s$ is chosen uniformly at random). Unfortunately, we do not know how to formalize this argument.

---

1In the case of minimum $s$-$t$ cut computations it suffices to take $k = 1$, since such computations in graphs $H_1, \ldots, H_k$ are equivalent to a minimum $s$-$t$ cut computation in the union of $H_1, \ldots, H_k$. The same holds for the two special cases of OrderedCuts discussed before Theorem 1 (the sources need to be merged). However, in the most general case of OrderedCuts such reduction may not work.

2Throughout the paper we assume that $t_{MC}(n, m) = \Omega(n + m)$ and $t_{OC}(n, m) = \Omega(n + m)$, and so we do not explicitly include the time spent outside these subroutines in complexity bounds.

3The actual algorithm formulated in Section 2 is slightly different: instead of computing minimum $s$-$\alpha$ cut, we split $G$ into components using the information obtained in step (2), and then compute an appropriate minimum cut in each component. The guarantees for random permutations are still preserved.
On the high level, divide-and-conquer strategies for computing OC tree have the same form as the computations performed by the classical Gomory-Hu algorithm: given a problem on the current graph $G$, run maxflow algorithm to obtain a certain cut $(S,T)$ and then recurse on the problems defined by $S$ and by $T$ (contracting the other set to a single vertex). There is, however, an important distinction between the two: cuts computed in the GH approach are minimum $s$-$t$ in the original graph for some terminals $s, t$, whereas cuts in the OC approach are minimum $S$-$T$ cuts in the original graph where the sizes of $S, T$ may grow at deeper levels of the recursion, and furthermore the size of $T$ can be controlled to a certain degree. Thus, one may expect the OC approach to give more balanced cuts in practice. Note, if all cuts $(S, T)$ were balanced (meaning $\min\{|S|, |T|\} = \Omega(|V_G|)$) then the maximum depth of the recursion would be logarithmic in $n$ and the approaches would take $\tilde{O}(1)$ calls to the maxflow algorithm on graphs of size $(O(n), O(m))$.

To test this intuition, we have implemented a version of the algorithm based on the OrderedCuts procedure, and compared it experimentally with the classical Gomory-Hu algorithm. On all of our test instances (except for one family of graphs, namely cycle graphs) the total size of graphs on which the maxflow algorithm is run was significantly smaller in the OrderedCuts approach compared to the GH approach. For the majority of test instances this translated to smaller runtimes (and outperformed previous implementations of [GT01] and [AIS+16]). Details of this study are reported in [Ko22].

We thus believe that besides theoretical contributions described above our work has a significant practical value, as it provides a basis for a practical implementation that appears to be the current state-of-the-art.

**Related work** GH tree construction algorithms have been extensively studied in the literature. A significant progress has been made for unweighted simple graphs [BHLP07, AKT15, AKT21a, LPS21, AK22] and for the approximate version of the problem [AKT20, LP21, LNPS22]. Several authors also explored fine-grained reductions from the GH construction problem to some other problems. Our work also follows this direction, so we describe existing results in more detail.

Abboud, Krauthgamer and Trabelsi considered in [AKT20] the “single source min cut problem” whose goal is to compute minimum $s$-$v$ cuts $C_{sv}$ for a fixed source node $s$ and all other nodes $v$. More precisely, Abboud et al. considered two variants; one of them asks for the set of edges of $C_{sv}$, and the other one asks only for the cost of $C_{sv}$ which we denote as $f(s, v) = \text{cost}(C_{sv})$. These problems can be formulated as the follows.

**Algorithm 2:** $\text{SingleSourceMinCut}(s; G)$.

1. for each $v \in V - \{s\}$ compute minimum $s$-$v$ cut $C_{sv}$
2. build data structure $D$ that supports operation $\text{Query}(v)$ for given $v \in V - \{s\}$, return $D$
3. /* Query$(v)$ should return $\delta C_{sv}$ in time $\tilde{O}(|\delta C_{sv}|)$, where $\delta C_{sv}$ is the set of edges of the cut $(C_{sv}, V - C_{sv})$ */

**Algorithm 3:** $\text{SingleSourceMaxFlow}(s; G)$.

1. for each $v \in V - \{s\}$ compute $\lambda(v) = f(s, v)$, return $\lambda$

It was shown in [AKT20] that the GH tree can be constructed by $\tilde{O}(1)$ calls to $\text{SingleSourceMinCut}(\cdot)$ on graphs with $O(n)$ nodes and $O(m)$ edges plus $\tilde{O}(m)$ additional work. For the second procedure [AKT20] was able to show a weaker result, namely how to construct a flow-equivalent tree for $G$ by $\tilde{O}(1)$ calls to $\text{SingleSourceMaxFlow}(\cdot)$ on graphs with $O(n)$ nodes and $O(m)$ edges plus $\tilde{O}(n)$ additional work. Li, Panigrahi and Saranurak observed in [LPS21] that instead of computing values $f(s, v)$ it suffices to verify whether $f(s, v)$ equals a given upper bound $\mu(v)$. More precisely,

4A flow-equivalent tree for $G$ is a spanning tree $T$ on $V$ such that for any $s, t \in V$, the costs of the minimum cut $s$-$t$ in $G$ and in $T$ coincide. Every GH tree is a also flow-equivalent tree, but the reverse is not true.
It is an S partition tree putting the cut tree. It works with a for brevity.

Consider an undirected weighted graph $G$. They constructed in $[AKL+22]$ a Monte-Carlo algorithm for $\text{SingleSourceMaxflowVerification}$ with $\gamma = 0.1$ whose complexity is $\tilde{O}(n^2)$, and thus obtained the improvement over $O(n)$ maxflow computations mentioned earlier.

We remark that $\text{OrderedCuts}$ and $\text{SingleSourceMaxflowVerification}$ can be viewed as non-standard relaxations of the $\text{SingleSourceMaxflow}$ problem. The former two problems appear incomparable.

**Connection to the Hao-Orlin algorithm** The strong version of $\text{OrderedCuts}$ is reminiscent of the Hao-Orlin algorithm $[HO93]$ for computing a minimum cut in a graph. The latter also computes minimum $\{s,v_1,\ldots,v_i\}$ cut for each $i \in [\ell]$, for some sequence of nodes $s,v_1,\ldots,v_i$. It has the same complexity $O((nm\log(n^2/m)))$ as a single call to a push-relabel maxflow algorithm. The crucial difference is that the Hao-Orlin algorithm selects the order of nodes $v_1,\ldots,v_i$ itself; it cannot be used for a given ordering of nodes. Note that the Hao-Orlin algorithm works for directed graphs, whereas our results for $\text{OrderedCuts}$ are restricted to undirected weighted graphs.

The rest of the paper is organized as follows. Section 1 gives background on GH tree construction algorithms. Section 2 analyzes strong $\text{OrderedCuts}$: it defines the notion of OC trees and studies their properties. Section 3 introduces depth-1 OC trees and shows how they can be used for constructing GH trees. Finally, Section 4 shows how to construct GH tree using weak $\text{OrderedCuts}$. All proofs are given in Appendices $A$, $B$, $C$.

## 1 Background and notation

Consider an undirected weighted graph $G = (V_G, E_G, w_G)$. A cut of $G$ is a set $U$ with $\emptyset \subsetneq U \subsetneq V_G$. It is an $S$-$T$ cut for disjoint subsets $S,T$ of $V_G$ if $T \subsetneq U \subsetneq V_G - S$. The cost of $U$ is defined as $\text{cost}_G(U) = \sum_{uv \in E_G : \{u,v\} \cap U = \emptyset} w_G(uv)$. The cost of a minimum $S$-$T$ cut in $G$ is denoted as $f_G(S,T)$. If one of the sets $S,T$ is singleton, e.g. $T = \{t\}$, then we say “$S$-$t$ cut” and write $f_G(S,t)$ for brevity.

When graph $G$ is clear from the context we omit subscript $G$ and write $V$, $E$, $\text{cost}(U)$, $f(S,T)$.

### Gomory-Hu algorithm

Below we review the classical Gomory-Hu algorithm $[GH61]$ for computing the cut tree. It works with a partition tree for $G$ which is a spanning tree $T = (V_T, E_T)$ such that $V_T$ is a partition of $V$. An element $X \in V_T$ is called a supernode of $T$. Each edge $XY \in E_T$ defines a cut in $G$ in a natural way; it will be denoted as $C_{XY}$ where we assume that $Y \subseteq C_{XY}$. We view $T$ as a weighted tree where the weight of $XY$ (equivalently, $f_T(X,Y)$) equals $\text{cost}_G(C_{XY})$. Tree $T$ is complete if all supernodes are singleton subsets of the form $\{v\}$; such $T$ can be identified with a spanning tree on $V$ in a natural way.

We use the following notation for a graph $G$, partition tree $T$ on $V$ and supernode $X \in V_T$:

- $H = G[T, X]$ is the auxiliary graph obtained from $G$ as follows: (i) initialize $H := G$, let $F$ be the forest on $V_T - \{X\}$ obtained from tree $T$ by removing node $X$; (ii) for each edge $XY \in E_T$ find the connected component $C_Y$ of $F$ containing $Y$, and then modify $H$ by contracting nodes in $\bigcup_{C \in C_Y} C$ to a single node called $v_Y$. Note that $V_H = X \cup \{v_Y : XY \in E_T\}$.

---

**Algorithm 4: SingleSourceMaxflowVerification**

```plaintext
/* preconditions: $X \subseteq V - \{s\}$, $\max_{v \in X} f(s,v) \leq (1+\gamma) \min_{v \in X} f(s,v)$, $\mu(v) \geq f(s,v)$ for all $v \in X$ */

1 compute $Y = \{v \in X : \mu(v) = f(s,v)\}$, return $Y$
```

It was shown in $[LPS21]$ that the GH tree can be constructed via $\tilde{O}(1)$ calls to $\text{SingleSourceMaxflowVerification}$ and to the maxflow algorithm on graphs of size $(n,m)$; if $\gamma = +\infty$ then the answer is correct with probability 1, otherwise it is correct w.h.p.. Abboud et al. designed in $[AKL+22]$ a Monte-Carlo algorithm for $\text{SingleSourceMaxflowVerification}$ with $\gamma = 0.1$ whose complexity is $\tilde{O}(n^2)$, and thus obtained the improvement over $O(n)$ maxflow computations mentioned earlier.
Algorithm 5: Gomory-Hu algorithm for graph $G$.

1. set $\mathcal{T} = (\{V\}, \emptyset)$
2. while exists $X \in V_\mathcal{T}$ with $|X| \geq 2$ do
   3. pick supernode $X \in V_\mathcal{T}$ with $|X| \geq 2$ and distinct $s, t \in X$
   4. form auxiliary graph $H = G[\mathcal{T}, X]$
   5. compute minimum $s$-$t$ cut $S$ in $H$
   6. define $(A, B) = (X - S, X \cap S)$, update $V_\mathcal{T} := (V_\mathcal{T} - \{X\}) \cup \{A, B\}$ and $E_\mathcal{T} := E_\mathcal{T} \cup \{AB\}$
   7. for each $XY \in E_\mathcal{T}$ update $E_\mathcal{T} := (E_\mathcal{T} - \{XY\}) \cup \{CY\}$ where $C = \begin{cases} A & \text{if } v_Y \notin S \\ B & \text{if } v_Y \in S \end{cases}$

Note that at every step Algorithm 5 splits some supernode $X$ into two smaller supernodes, $A$ and $B$. We will use a generalized version proposed by Abboud et al. [AKT20] in which one iteration may split $X$ into more than two supernodes.

Algorithm 6: Generalized Gomory-Hu algorithm for graph $G$.

1. set $\mathcal{T} = (\{V\}, \emptyset)$
2. while exists $X \in V_\mathcal{T}$ with $|X| \geq 2$ do
   3. pick supernode $X \in V_\mathcal{T}$ with $|X| \geq 2$, form auxiliary graph $H = G[\mathcal{T}, X]$
   4. find node $s \in X$ and non-empty laminar family $\Pi$ of subsets of $V_H - \{s\}$ such that each $S \in \Pi$ is a minimum $s$-$t$ cut in $H$ for some $t \in X$
   5. while $\Pi \neq \emptyset$ do
      6. pick minimal set $S \in \Pi$
      7. update $\mathcal{T}$ as in lines 6-7 of Alg. 5 for given $S$ (with $(A, B) = (X - S, X \cap S)$)
      8. update $X := A$, update $H$ accordingly to restore $H = G[\mathcal{T}, X]$ /* now $v_B \in V_H$ */
      9. remove $S$ from $\Pi$, for each $S' \in \Pi$ with $S \subseteq S'$ replace $S'$ with $(S' - S) \cup v_B$ in $\Pi$

Note that graph $H$ at line 8 is updated by contracting set $S$ to a single vertex named $v_B$. Clearly, such transformation and the update at line 9 preserve the property of set $\Pi$: it is still a laminar family of subsets of $V_H - \{s\}$ such that each $S \in \Pi$ is a minimum $s$-$t$ cut in $H$ for some $t \in X$. This means Algorithm 6 updates tree $\mathcal{T}$ in the same way as Algorithm 5 (with certain choices of triplets $(X, s, t)$). This fact implies the correctness of Algorithm 6.

To conclude this section, we state two more results that we will need later. The first one deals with procedure IsolatingCuts, which appeared in [LP20] (and independently in [AKT21b]). It can be formulated as follows.

Algorithm 7: IsolatingCuts($s, Y; G$).

1. for each $v \in Y$ compute minimum $(Y \cup \{s\} - \{v\})$-$v$ cut $C_v$, return these cuts

Importantly, Algorithm 7 can be efficiently implemented.

Theorem 3 ([LP20]). There exists an implementation of IsolatingCuts($s, \{v_1, \ldots, v_\ell\}; G$) that runs in time $O((\log \ell) \cdot \log(n, m))$ and outputs a set of disjoint cuts $\{C_{v_i} : i \in [\ell]\}$.

The second result appeared in [GT01] in the context of the Hao-Orlin algorithm. Using our notation, it can be stated as follows.

Lemma 4 ([GT01]). Consider distinct nodes $v_0, v_1, \ldots, v_k$ in an undirected graph $G$ with $k \geq 1$. If $f(\{v_0, \ldots, v_k\}) = \min_{i \in [k]} f(\{v_0, \ldots, v_{i-1}\}, v_i)$ then $f(\{v_0, \ldots, v_{k-1}\}, v_k) = f(v_0, v_k)$.

5Ignoring base cases, the algorithm can be described as follows:
   (1) compute minimum $\{s, v_1, \ldots, v_k\} - \{v_k\}$ cut $(S, T)$ in $G$ where $k = \lfloor \ell/2 \rfloor$;
   (2) make recursive calls IsolatingCuts($s', \{v_1, \ldots, v_k\}; G'$) and IsolatingCuts($s', \{v_k, v_{k+1}, \ldots, v_\ell\}; G'$) where $G'$ and $G''$ are the graphs obtained from $G$ by contracting respectively sets $S$ and $T$ to a single nodes $s'$ and $s''$;
   (3) combine the results.
We say that

\[ \text{Definition 5.} \quad \text{Consider sequence} \quad \varphi \quad \text{of distinct nodes in} \quad G \quad \text{OrderedCuts} \quad s \quad \text{Properties of strong} \quad \text{OC} \quad \text{can be defined as the unique} \quad \text{tree for} \quad \varphi \quad \text{holds: if} \quad \varphi \quad \text{tree should always be clear from the context (or} \quad \text{context).} \quad \text{Note that operation} \quad \varphi \quad \text{tree encodes all cuts in the defini-} \quad \text{Historically, a valid} \quad \text{problem; for example, set} \quad \varphi \quad \text{node} \quad s \quad \text{is laminar (this can be easily deduced from the submodularity inequality for cuts).} \quad \text{We denote} \quad \text{for a set of subsets} \quad \Pi. \quad \text{Consider undirected weighted graph} \quad G = (V, E, w). \quad \text{We will use letters} \quad \varphi, \alpha, \beta, \ldots \quad \text{sequence} \quad \varphi = s \ldots. \quad \text{An} \quad \text{OC tree for} \quad \varphi \quad \text{is a pair} \quad (\Omega, \mathcal{E}) \quad \text{where} \quad \begin{itemize} \item (\varphi, \mathcal{E}) \quad \text{is a rooted tree with the root} \quad s \quad \text{and edges oriented towards the root such that for any} \quad uv \in \mathcal{E} \quad \text{we have} \quad v \sqsubseteq u. \quad \text{We write} \quad u \preceq v \quad \text{for nodes} \quad u, v \in \varphi \quad \text{if this tree has a directed path from} \quad u \quad \text{to} \quad v. \quad \item \Omega \quad \text{is a partition of some set} \quad V \quad \text{with} \quad |\Omega| = |\varphi| \quad \text{such that} \quad |A \cap \varphi| = 1 \quad \text{for all} \quad A \in \Omega. \quad \text{For node} \quad v \in V \quad \text{we denote} \quad [v] \quad \text{to be the unique component in} \quad \Omega \quad \text{containing} \quad v \quad \text{then} \quad [u] \neq [v] \quad \text{for distinct} \quad u, v \in \varphi. \quad \text{We also denote} \quad [v] = \bigcup_{u \in \varphi : u \preceq v} [u] \quad \text{for} \quad v \in \varphi. \quad \text{We say that} \quad (\Omega, \mathcal{E}) \quad \text{is an} \quad \text{OC tree for} \quad (\varphi, G) \quad \text{(or is valid for} \quad G, \quad \text{when} \quad \varphi \quad \text{is clear)} \quad \text{if the following holds: if} \quad \varphi = \alpha v \ldots \quad \text{with} \quad |\alpha| \geq 1 \quad \text{then} \quad [v] \quad \text{is a minimum} \quad \alpha \text{-}v \text{cut in} \quad G. \quad \end{itemize} \quad \text{Note that operation} \quad [\cdot] \quad \text{depends on} \quad \Omega. \quad \text{Below we will need to work with multiple} \quad \text{OC trees. Unless noted otherwise, we will use the following convention: different} \quad \text{OC trees will be denoted with an additional symbol, e.g.} \quad (\Omega, \mathcal{E}), \quad (\Omega', \mathcal{E}'), \ldots, \quad \text{and the corresponding operations will be denoted accordingly as} \quad [\cdot], \quad [\cdot]' , \ldots. \quad \text{Operations} \quad \preceq \quad \text{and} \quad (\cdot)^{\perp} \quad \text{also depend on the} \quad \text{OC tree, but we will always use the same notation for them; the corresponding} \quad \text{OC tree should always be clear from the context (or these operations would be the same for all considered} \quad \text{OC trees).} \quad \text{Consider} \quad \text{OC tree} \quad (\Omega, \mathcal{E}) \quad \text{for sequence} \quad \varphi \quad \text{and a leaf node} \quad u \in \varphi. \quad \text{Let} \quad v \quad \text{be the parent of} \quad u \quad \text{in} \quad (\varphi, \mathcal{E}) \quad (\text{i.e.} \quad uv \in \mathcal{E}). \quad \text{We define} \quad \varphi^{-u} \quad \text{to be the sequence obtained from} \quad \varphi \quad \text{by removing node} \quad u, \quad \text{and} \quad (\Omega^{-u}, \mathcal{E}^{-u}) \quad \text{to be the} \quad \text{OC tree for} \quad \varphi^{-u} \quad \text{obtained from} \quad (\Omega, \mathcal{E}) \quad \text{by removing edge} \quad uv \quad \text{and merging components} \quad [u] \quad \text{and} \quad [v] \quad \text{of} \quad \Omega \quad \text{into a single component} \quad [v]^{-u} = [v] \cup [u]. \quad \text{Equivalently,} \quad (\Omega^{-u}, \mathcal{E}^{-u}) \quad \text{can be defined as the unique} \quad \text{OC tree for} \quad \varphi^{-u} \quad \text{with the property that} \quad [w]^{-u} = [w]^{\perp} \quad \text{for all} \quad w \in \varphi^{-u}. \quad \text{Figure 1: Example} \quad \text{OC tree for sequence} \quad \varphi = sv_1v_2v_3v_4. \quad \text{Black and white circles are nodes in} \quad \varphi \quad \text{and in} \quad V - \varphi, \quad \text{re-}

\begin{align*}
\varphi = \{ v_0, \ldots, v_{i-1}, v_i \}, \quad &\text{Suppose the claim is false, then there exists} \quad s-v_k \quad \text{cut} \quad S \quad \text{with} \quad v_k \in S \quad \text{and} \quad \text{cost}(S) < \lambda_k. \quad \text{Let} \quad i \quad \text{be the largest index in} \quad [k] \quad \text{such that} \quad \{ s, v_1, \ldots, v_{i-1} \} \subseteq V - S, \quad \text{then} \quad v_i \in S \quad \text{and so} \quad \text{cost}(S) \geq \lambda_i, \quad \text{contradicting condition} \quad \lambda_k \leq \min_{i \in [k]} \lambda_i. \quad \square
\end{align*}

Throughout the paper we will use notation \( C_{uv} \) to denote the minimal minimum \( u-v \) cut in a given graph. (The graph will always be clear from the context). Note that \( v \in C_{uv} \). Set \( C_{uv} \) is also known as the latest \( u-v \) cut [Gab91]. It is well-known that the family \( \{ C_{sv} : v \neq s \} \) for a fixed node \( s \) is laminar (this can be easily deduced from the submodularity inequality for cuts).

We denote \( (\Pi) = \bigcup_{S \in \Pi} S \) for a set of subsets \( \Pi \).
Lemma 6. Suppose that \((\Omega, \mathcal{E})\) is an OC tree for \(\varphi u\) and \(v \in \varphi\) is the parent of \(u\) in \((\varphi u, \mathcal{E})\). Suppose further that \((\Omega^- u, \mathcal{E}^- u)\) is an OC tree for \((\varphi, G)\). Then \((\Omega, \mathcal{E})\) is valid for \(G\) if and only if set \([u]\) is a minimum \(v\)-\(u\) cut in \(G([v]^- u; v]\).

Lemma 6 gives a constructive proof of the existence of OC tree for a given \((\varphi, G)\).

Corollary 7. For every undirected graph \(G\) and non-empty sequence \(\varphi\) of nodes in \(G\) there exists OC tree \((\Omega, \mathcal{E})\) for \((\varphi, G)\).

**Proof.** It suffices to show that if there exists OC tree \((\Omega, \mathcal{E})\) for \((\varphi, G)\) and \(u\) is a node in \(V - \varphi\), then there also exists OC tree \((\hat{\Omega}, \hat{\mathcal{E}})\) for \((\varphi u, G)\); the claim will then follow by induction. Let \(v \in \varphi\) be the unique node in \(\varphi\) with \(u \in [v]\). Let \(T\) be a minimum \(v\)-\(u\) cut in \(G([v]^- u; v]\). Define \(\hat{\Omega} = (\Omega - [v]) \cup \{v - T, T\}\) and \(\hat{\mathcal{E}} = \mathcal{E} \cup \{uv\}\). Note that \((\hat{\Omega}, \hat{\mathcal{E}})\) is valid for \((\varphi u, G)\). \(\square\)

**Lemma 8.**If \(\text{cost}([u]\mathcal{E}) \geq \text{cost}([u]\mathcal{E})\) for all \(w \in \pi^*(u) - \{s\}\) then \([u]\mathcal{E}\) is a minimum \(s\)-\(u\) cut.

**Lemma 9.** Consider sequence \(\varphi = \alpha \ldots\) in graph \(G\) with \(|\alpha| \geq 1\). Suppose that \((\Omega^O, \mathcal{E}^O)\) is an OC tree for \((\varphi, G)\), and for each \(v \in \alpha\) pair \((\Omega^v, \mathcal{E}^v)\) is an OC tree for \((\varphi \cap [v], G([v]; v]\)). Then \((\Omega, \mathcal{E})\) is an OC tree for \((\varphi, G)\) where \(\Omega = \bigcup_{v \in \alpha} \Omega^v\) and \(\mathcal{E} = \mathcal{E}^O \cup \bigcup_{v \in \alpha} \mathcal{E}^v\).

**Algorithm 8: OrderedCuts(\(\varphi; G)\):** divide-and-conquer algorithm.

1. if \(|\varphi| \leq 2\) then compute OC tree \((\Omega, \mathcal{E})\) for \((\varphi, G)\) non-recursively and return \((\Omega, \mathcal{E})\)
2. split \(\varphi = \alpha \beta\) where \(|\alpha| = \left\lfloor \frac{1}{2} (|\varphi| + 1) \right\rfloor\)
3. call \((\Omega^\alpha, \mathcal{E}^\alpha) \leftarrow \text{OrderedCuts}(\alpha; G)\), set \((\Omega, \mathcal{E}) = (\varnothing, \mathcal{E}^\alpha)\)
4. for \(v \in \alpha\) do
5. compute minimum \(v\)-\(\beta\) cut \((S, T)\) in \(G([v]^-; v]\) such that \(T\) is minimal
6. let \((\Omega^v, \mathcal{E}^v) \leftarrow \text{OrderedCuts}(v(\beta \cap T), G[T \cup \{v]\]; v]\))
7. update \(\Omega := \Omega \cup \{S \cup [v]^\alpha\} \cup (\Omega^v - [v]^\alpha)\) and \(\mathcal{E} := \mathcal{E} \cup \mathcal{E}^v\)
8. return \((\Omega, \mathcal{E})\)
The correctness follows by an induction argument and by the previous lemmas: at line 7 \((\{S \cup \{v\} \cup (\Omega' - \{v\}),\mathcal{E}')\) is an \(\mathcal{O}\) tree for \(G[\{v\};v]\) by Lemma 10 and the output at line 8 is an \(\mathcal{O}\) tree for \((\varphi,G)\) by Lemma 9.

**Theorem 11.** Algorithm 5 for sequence \( \varphi \) has complexity of maximum flow computations on \(O(n)\) graphs of size \((n,m)\) each. If \( \varphi = (s,v_1,\ldots,v_t) \) and the order of \(v_1,\ldots,v_t\) is a uniformly random permutation then the expected total number of nodes and edges in these graphs is \((O(n^{1+\gamma}),O(n^{1+\gamma}m))\) where \( \gamma = \log_2 1.5 = 0.584\ldots \).

### 3 Computing Gomory-Hu tree via depth-1 \(\mathcal{O}\) trees

In this section we introduce the notion of a depth-1 \(\mathcal{O}\) tree, or \(\mathcal{O}_1\) tree. We then show how we can use \(\mathcal{O}_1\) tree to compute the GH tree.

**Definition 12.** Consider sequence \( \varphi = s \ldots \) in graph \( G \). A \(\mathcal{O}_1\) tree for \((\varphi,G)\) is a pair \((\Omega,\mathcal{E})\) such that there exists subsequence \( \psi = s \ldots \) of \( \varphi \) with the following properties.

- \((\psi,\mathcal{E})\) is a star graph with the root \( s \), i.e. \( \mathcal{E} = \{vs : v \in \psi - \{s\}\} \).
- \((\psi,\mathcal{E})\) is an \(\mathcal{O}\) tree for \((\varphi,G)\).
- For every \( v \in \varphi - \{s\} \) there exists \( u \in \psi - \{s\} \) such that \( u \subseteq v \) and \( v \in [u] \).

The following result shows the existence of an \(\mathcal{O}_1\) tree.

**Lemma 13.** Let \((\Omega,\mathcal{E})\) be an \(\mathcal{O}\) tree for \((\varphi,G)\). Define \( \psi = \varphi \), and let us modify tuple \((\psi,\Omega,\mathcal{E})\) using the following algorithm: while \((\psi,\Omega,\mathcal{E})\) has a leaf node of depth two or larger, pick the rightmost such node \( u \in \psi \) and replace \((\psi,\Omega,\mathcal{E})\) with \((\psi^{u},\Omega^{u},\mathcal{E}^{u})\). Then the resulting pair \((\Omega,\mathcal{E})\) is an \(\mathcal{O}_1\) tree for \((\varphi,G)\).

The lemma means that an \(\mathcal{O}_1\) tree for \((\varphi,G)\) can be trivially constructed if we have \(\mathcal{O}\) for \((\varphi,G)\). However, an \(\mathcal{O}_1\) tree can potentially be constructed faster than an \(\mathcal{O}\) tree, since we do not need to recurse on subproblems corresponding to depths larger than 1.

Let \(\text{OrderedCuts}_1(\varphi;G)\) be a procedure that outputs an \(\mathcal{O}_1\) tree for \((\varphi,G)\). The main result of this section is the following theorem.

**Theorem 14.** There exists a randomized (Las-Vegas) algorithm for computing GH tree for graph \( G \) with expected complexity \(O(1) \cdot (t_{\mathcal{O}_1}(n,m))\), where \( t_{\mathcal{O}_1}(n,m) \) is the complexity of \(\text{OrderedCuts}_1(\cdot)\) on a graph with \(O(n)\) nodes and \(O(m)\) edges.

It will be convenient to define a named partition of set \( V \) as a pair \((\mathcal{S},\Pi)\) where \( \mathcal{S} \subseteq V \) and \( \Pi \) is a set of disjoint subsets of \( V \) such that (i) \( \mathcal{S} \subseteq \Pi \) and (ii) for each \( v \in \mathcal{S} \) there exists unique \( U \in \Pi \) with \( v \in U \). With some abuse of terminology we will denote a named partition with a single letter \( \mathcal{S} \) (treating it as a subset of \( V \) when needed), and will denote set \( U \) with \( \mathcal{S} \) as \( \mathcal{S}_v \). We denote \( \langle \mathcal{S} \rangle = \bigcup_{v \in \mathcal{S}} \mathcal{S}_v \).

Note that an \(\mathcal{O}_1\) tree \((\Omega,\mathcal{E})\) can be uniquely represented by a named partition \( \mathcal{S} \) with \( \mathcal{S} = \{v : v \in \mathcal{E}\} \) and \( \mathcal{S}_v \in \Omega \) for \( v \in \mathcal{S} \). We can thus assume that procedure \(\text{OrderedCuts}_1(\varphi;G)\) for sequence \( \varphi = s \ldots \) returns named partition \( \mathcal{S} \) of \( V_G - \{s\} \), with \( \mathcal{S} \subseteq \varphi - \{s\} \). In the new notation Definition 12 can be reformulated as follows.

**Definition 15.** Named partition \( \mathcal{S} \) of \( V_G - \{s\} \) is an \(\mathcal{O}_1\) tree for sequence \( \varphi = sv_1 \ldots v_t \) and graph \( G \) if

(i) For every \( v_k \in \mathcal{S} \) set \( \mathcal{S}_{v_k} \) is a minimum \( s((v_1,\ldots,v_{k-1}) \cap \mathcal{S})v_k \) cut in \( G \).

(ii) For every \( k \in [t] \) there exists \( v_i \in \{v_1,\ldots,v_k\} \cap \mathcal{S} \) with \( v_k \in \mathcal{S}_{v_i} \).

**Remark 1.** In the algorithm below instead of condition (ii) we will only need a slightly weaker property: condition (ii) needs to hold only for \( k \in [t] \) such that \( \{v_1,\ldots,v_{k-1}\} \cap C_{sv_k} = \emptyset \) where \( C_{sv_k} \) is a minimum \( s-v_k \) cut.
3.1 Computing cuts for a fixed source

Let us consider graph \( G = (V, E, w) \), node \( s \in V \) and subset \( X \subseteq V - \{s\} \). We first show how we can use procedure \texttt{OrderedCuts}_1 to compute minimum \( s \)-\( v \) cuts for many nodes \( v \in X \) under the following condition.

\textbf{Assumption 1.} For every \( v \in X \) the minimum \( s \)-\( v \) cut in \( G \) is unique.

In particular, we will present an algorithm with the following properties.

\textbf{Theorem 16.} There exists randomized algorithm \texttt{FixedSourcePartition}_1\((s, X; G)\) that outputs a named partition \( S \) of \( V_G - \{s\} \) with \( S \subseteq X \) satisfying the following:

(a) \( S \) is a minimum \( s \)-\( v \) cut in \( G \) for every \( v \in S \).

(b) If Assumption 1 holds then \( X \subseteq \langle S \rangle \) with probability at least \( 1 - 1/poly(|X|) \) (for an arbitrary fixed polynomial).

It makes \( O(\log^3 |X|) \) calls to procedure \texttt{OrderedCuts}_1\((\cdot)\) for graph \( G \) and performs \( O(|X| \log^4 |X|) \) additional work.

The algorithm is given below. \texttt{RandomSubset}(\( X; \alpha \)) in line 3 is a random subset of \( X \) in which each element \( v \in X \) is included independently with prob. \( \alpha \).

\begin{algorithm}
  \caption{FixedSourcePartition\(_1\)((s, X; G))\).}
  \begin{algorithmic}[1]
    \State set \( \lambda(v) = \text{cost}\{(v)\} \) for all \( v \in X \)
    \For{\( i = 1, \ldots, N + 1 \)}
      \State sample \( Y \leftarrow \text{RandomSubset}(X; \alpha_i) \) /* use \( \alpha_{N+1} = 1 \) */
      \State sort nodes in \( Y \) as \( v_1, \ldots, v_\ell \) so that \( \lambda(v_1) \geq \ldots \geq \lambda(v_\ell) \), \( \ell = |Y| \)
      \State call \( S \leftarrow \text{OrderedCuts}_1(sv_1, \ldots, v_\ell; G) \) /* \( S \subseteq Y \subseteq X \) */
      \If{\( i \leq N \)}
        \For{each \( v \in S \)}
          \State update \( X := X - (S_v - \{v\}) \)
          \For{each \( v \in S \) and \( u \in S_v \)}
            \State update \( \lambda(u) := \min\{\lambda(u), \text{cost}(S_v)\} \)
        \EndFor
      \EndIf
      \Else
        \State let \( S^* = \{v_k \in S : \text{cost}(S_{v_k}) \leq \text{cost}(S_v) \ \forall v \in \{v_1, \ldots, v_k\} \cap S\} \)
        \State return named partition \( S^* \) where \( S^*_v = S_v \) for \( v \in S^* \)
  \EndFor
  \end{algorithmic}
\end{algorithm}

Note that the algorithm depends on the sequence \( \alpha_1, \ldots, \alpha_N \). We set it by taking sequence \( (2^0, 2^{-1}, \ldots, 2^{-d}) \) for \( d = \lfloor \log_2 |X| \rfloor \) and repeating it \( \Theta(\log^2 |X|) \) times with an appropriate constant (so that \( N = \Theta(\log^3 |X|) \)). Note that the idea of this construction comes from [LP21]. The analysis of the algorithm with this choice and the proof of Theorem 16 are given in Section B.2.

3.2 Overall algorithm

We now come back to the problem of computing the GH tree. Let us recall the main computational problem at line 4 of Algorithm 6: given graph \( H = (V_H, E_H) \) and subset \( X \subseteq V_H \) with \(|X| \geq 2\), we need compute node \( s \in X \) and a set of disjoint subsets \( \Pi \) such that each \( S \in \Pi \) is an \( s \)-\( t \) cut in \( H \) for some \( t \in X \). From now on we fix graph \( H \) and set \( X \).

Let us introduce a total order \( \preceq \) on cuts \( S \subseteq V_H \) as follows: (i) if \( \text{cost}(S) < \text{cost}(S') \) then \( S \prec S' \); (ii) if \( \text{cost}(S) = \text{cost}(S') \) and \(|S \cap X| < |S' \cap X|\) then \( S < S' \); (iii) otherwise use an arbitrary rule. We write \( u \sqsubseteq v \) for distinct nodes \( u, v \in X \) if \( C_{uv} < C_{uv} \). Equivalently, \( u \sqsubseteq v \) if \( u \in S \) where \( S \) is the minimum subset w.r.t. \( \preceq \) that separates \( u \) and \( v \). We write \( u \sqsubseteq v \) if either \( u \sqsubseteq v \) or \( u = v \).

\textbf{Lemma 17.} Relation \( \sqsubseteq \) is a total order on \( X \).
Proof. It suffices to show that for any distinct $a, b, c \in X$ relation $\subseteq$ is not cyclic on $a, b, c$. We can assume w.l.o.g. that $C_{ab}$ is the smallest subset w.r.t. $\preceq$ among $C_{xy}$ for distinct $x, y \in \{a, b, c\}$. Note that $b \sqsubseteq a$. If $c \in C_{ab}$ then $C_{ab}$ is an $a$-$c$ cut with $C_{ab} \preceq C_{ac}$, implying $C_{ac} = C_{ab}$ and $c \sqsubseteq a$. If $c \notin C_{ab}$ then $C_{ab}$ is a $c$-$b$ cut with $C_{ab} \preceq C_{cb}$, implying $C_{cb} = C_{ab}$ and $b \sqsubseteq c$. In both cases $\subseteq$ is not cyclic on $a, b, c$. 

Total order $\sqsubseteq$ will play an important role in the algorithm. We will also need to make sure that Assumption $\square$ holds. For that we can use a standard technique of adding a small random perturbation to the edge weights. The correctness of this technique follows from the Isolation Lemma of [MVV87] (see e.g. [AKT20] Proposition 3.14).

**Proposition 18.** One can add random polynomially-bounded values to the edge weights in $H$ such that any GH tree in the new graph $H'$ is also a valid GH tree for $H$, and furthermore with high probability $H'$ has a unique GH tree.

We are now ready to describe the algorithm (see below); the sequence of probabilities $\beta_1, \ldots, \beta_M$ will be specified later.

**Algorithm 10:** Line 4 of Algorithm 6

1: while true do
2:   pick arbitrary $s \in X$
3:   apply random perturbation to $H$ as in Proposition 18 to get graph $H'$
4:   for $i = 1, \ldots, M$ do
5:     sample $Y \leftarrow \text{RandomSubset}(X - \{s\}; \beta_i)$
6:     call $S \leftarrow \text{FixedSourcePartition}(s, Y; H')$
7:     if exists $v \in S$ with $V_H - S_v \prec S_v$ then pick arbitrary such $v$ and update $s := v$
8:     else if $X - \{s\} \subseteq \langle S \rangle$ then terminate and return $(s, \{S_v : v \in S\})$

In Sections B.3 and B.4 we will show the following.

**Lemma 19.** Suppose that $(\beta, \ldots, \beta_M) = (2^{-d}, \ldots, 2^{-d}, \ldots, 2^{-2}, \ldots, 2^{-2}, \ldots, 2^{-1}, \ldots, 2^{-1}, 1)$ where $d = \lfloor \log_2 |X| \rfloor$ and $K = \Theta(\log \log n)$ (with an appropriate constant). Then each run of Algorithm 10 (i.e. lines 2-8) terminates with probability $\Omega(1)$.

**Theorem 20.** The expected complexity of Algorithm 6 with line 4 implemented as in Algorithm 10 is $O(t_{OC}(n,m)) + n \log n \cdot \log^5 n \cdot \log \log n)$ where $t_{OC}(n,m)$ is the complexity of procedure OrderedCuts$_1$($\varphi$; $G$) on graph with $O(n)$ nodes and $O(m)$ edges.

## 4 Computing Gomory-Hu tree via OrderedCuts

In this section we show how to compute GH tree via $\tilde{O}(1)$ calls to (weak) OrderedCuts. Note that $OC_1$ tree and weak OrderedCuts for given input $(\varphi, G)$ are incomparable. In particular, cuts contained in an $OC_1$ tree may not give a valid output of weak OrderedCuts (as defined by Algorithm 1), while the output of weak OrderedCuts comes without a tree structure. Note, if we have a procedure that computes an $OC$ tree for $\varphi$ then we can easily compute both $OC_1$ tree and valid output of weak OrderedCuts. In that case the algorithm in this section will be more efficient than the one in the previous section by a factor of $(\log n) \cdot (\log \log n)$.

### 4.1 Procedure CertifiedOrderedCuts

Consider undirected weighted graph $G = (V, E, w)$ and a sequence of distinct nodes $s, v_1, \ldots, v_\ell$ in $V$. First, we describe a procedure that computes some $s$-$v_i$ cut for each $i \in \ell$, and certifies some
of these cuts as “true” minimum \( s \)-\( v \) cuts.

**Algorithm 11: CertifiedOrderedCuts**

1. call \( \lambda \leftarrow \text{OrderedCuts}(s, v_1, \ldots, v_\ell; G) \)
2. compute \( Y^* = \{v_k : \lambda(v_k) \leq \min_{i \in [k]} \lambda(v_i), k \in [\ell] \} \)
3. call \( \text{IsolatingCuts}(s, Y^*; G) \) to get cut \( S_s \subseteq V - \{s\} \) for each \( v \in Y^* \)
4. define \( C^* = \{S_v : v \in Y^*, \ \text{cost}(S_v) = \lambda(v)\} \)
5. return \( (\lambda, C^*) \)

**Lemma 21.** Let \( (\lambda, C^*) \) be the output of \( \text{CertifiedOrderedCuts}(s, v_1, \ldots, v_\ell; G) \), and denote \( Y = \{v_1, \ldots, v_\ell\} \). Pair \((\lambda, C^*)\) has the following properties.

(a) Subsets in \( C^* \) are disjoint, and each \( S \in C^* \) is a minimum \( s \)-\( v \) cut for some \( v \in Y \).
(b) If \( Y \cap C_{sv_k} = \{v_k\} \) and \( f(s, v_k) \leq \min_{i \in [k]} f(s, v_i) \) then \( C^* \) contains a minimum \( s \)-\( v_k \) cut.
(c) There exists a family of cuts \( C \subseteq \{S : \emptyset \neq S \subseteq V - \{s\}\} \) such that
   (i) \( \lambda(v) = \min_{S \in C} \text{cost}(S) \) for each \( v \in V - \{s\} \) (and hence \( \lambda(v) \geq f(s, v) \));
   (ii) if \( \{v_1, \ldots, v_\ell\} \cap C_{sv_k} = \{v_k\} \) then \( C \) contains a minimum \( s \)-\( v_k \) cut (and hence \( \lambda(v_k) = f(s, v_k) \) and \( \lambda(v) \leq f(s, v_k) \) for all \( v \in C_{sv_k} \)).

**Remark 2.** We will use only properties of \( \text{CertifiedOrderedCuts}(\cdot) \) given in Lemma 21. Any other implementation of \( \text{CertifiedOrderedCuts}(\cdot) \) would work, as long as Lemma 21 holds. If, for example, a strong version of \( \text{OrderedCuts} \) is used that explicitly constructs \( OC \) tree then set \( C^* \) can be constructed from this tree without calling \( \text{IsolatingCuts} \), see Lemma 8. Note that this lemma gives a weaker criterion for adding cuts to \( C^* \) that in general can certify a larger set of cuts stored in an \( OC \) tree.

4.2 Computing cuts for a fixed source

Let us consider graph \( G = (V, E, w) \), node \( s \in V \), subset \( X \subseteq V - \{s\} \) and number \( L \geq 1 \). Denote

\[
\hat{\Pi}^L_s = \{S \subseteq V - \{s\} : S \text{ is a minimum } s\text{-}t \text{ cut for some } t \in X, \ |S \cap X| \leq L\}
\]

\[
\Pi^L_s = \{S = C_{st} : t \in X, \ |S \cap X| \leq L\} \subseteq \hat{\Pi}^L_s
\]

We say that the output of \( \text{CertifiedOrderedCuts}(\cdot) \) is minimal if under the condition (b) of Lemma 21 family \( C^* \) contains the minimal \( s \)-\( v_k \) cut \( C_{sv_k} \), rather than an arbitrary minimum \( s \)-\( v_k \) cut. The main result of this section is given by the following theorem.

**Theorem 22.** There exists an algorithm that outputs a laminar family of subsets \( \Pi \subseteq \hat{\Pi}^L_s \). It makes \( O(\log^3 |X|) \) calls to procedure \( \text{CertifiedOrderedCuts} \) for graph \( H \) and performs \( O(|X| \log^4 |X|) \) additional work. Furthermore, it satisfies \( \langle \Pi \rangle = \langle \Pi^L_s \rangle \) with probability at least \( 1 - 1/poly(|X|) \) (for an arbitrary fixed polynomial) assuming that all outputs of \( \text{CertifiedOrderedCuts} \) are minimal.

Our algorithm is given below.

**Algorithm 12: FixedSourcePartition**

1. set \( \Pi = \emptyset \) and \( \mu(v) = \text{cost}(\{v\}) \) for all \( v \in V - \{s\} \)
2. for \( i = 1, \ldots, 2N \) do
   3. sample \( Y \leftarrow \text{RandomSubset}(X - \langle \Pi \rangle; \alpha_i) \)
   4. sort nodes in \( Y \) as \( v_1, \ldots, v_\ell \) so that \( \mu(v_1) \geq \ldots \geq \mu(v_\ell), \ \ell = |Y| \)
   5. call \( (\lambda, C^*) \leftarrow \text{CertifiedOrderedCuts}(s, v_1, \ldots, v_\ell; H) \)
   6. update \( \mu(v) := \min\{\mu(v), \lambda(v)\} \) for \( v \in Y \)
   7. update \( \Pi := \Pi \cup \{S \in C^* : |S \cap X| \leq L\} \)
   8. if \( \Pi \) becomes non-laminar then terminate and return \( \emptyset \)
9. return \( \Pi \)
We set sequence $\alpha_1, \ldots, \alpha_{2N}$ in the same way as in Section 3.1 (with $N = \Theta(\log^3 |X|)$). The analysis of the algorithm with this choice and the proof of Theorem 22 are given in Section C.2.

### 4.3 Overall algorithm

We now need to show how to implement line 4 of Algorithm 6 for given graph $H$ and subset $X \subseteq V_H$ with $|X| \geq 2$. One possibility is to use Algorithm 10 where the call $\text{FixedSourcePartition}(s, Y; H')$ is replaced with $\text{FixedSourcePartition}(s, Y, L; H')$ for a sufficiently large $L$ (namely $L \geq |X|$); clearly, in this case the outputs of these two calls have the same guarantee. Below we describe an alternative procedure whose complexity is smaller by a factor of $(\log n) \cdot (\log \log n)$. It uses $L = |X|/2$, which means that large cuts are discarded inside the call $\text{FixedSourcePartition}(s, Y; H')$. This allows to simplify the selection of the source $s \in X$: it can now be selected uniformly at random (as in AKT12).

**Algorithm 13:** Line 4 of Algorithm 6

1. while true do
2. sample $s \in X$ uniformly at random
3. apply random permutation to $H$ as in Proposition 18 to get graph $H'$
4. call $\Pi \leftarrow \text{FixedSourcePartition}(s, X - \{s\}, L; H')$ where $L \overset{\text{def}}{=} |X|/2$
5. remove non-maximal subsets from $\Pi$ /* this is to simplify the analysis */
6. if $|(V_H - (\Pi)) \cap X| \leq L$ then terminate and return $(s, \Pi)$

**Lemma 23.** Each iteration of Algorithm 13 succeeds with probability $\Omega(1)$ (and thus the expected number of iterations is $O(1)$).

**Proof.** We use for the analysis total order $\subseteq$ on $X$ defined in Section 3.2. Let $a$ be the $L$-th smallest node in $X$ w.r.t. $\subseteq$. With probability $\Omega(1)$ the following events will hold jointly: (i) $a \subseteq s$; (ii) $H'$ has the unique GH tree (and thus the outputs of $\text{CertifiedOrderedCuts}$ will be minimal); (iii) $\text{FixedSourcePartition}(s, X, L; H')$ returns $\Pi$ with $\langle \Pi \rangle = \langle \Pi^{(L)}_s \rangle$. In that case for any $v \in U \overset{\text{def}}{=} \{v \in X : v \subseteq a\}$ we have $v \subseteq s$, or equivalently $C_{sv} > C_{vs}$. This implies that $|C_{sv} \cap X| \leq |C_{vs} \cap X| \leq |(V - C_{sv}) \cap X|$, and so $|C_{sv} \cap X| \leq L$, $C_{sv} \in \Pi^{(L)}_s$ and hence $v \in C_{sv} \subseteq \langle \Pi^{(L)}_s \rangle = \Pi$. We have $U \subseteq \langle \Pi \rangle$ and $|U| = L$, and so $|(V - (\Pi)) \cap X| \leq |(V - U) \cap X| \leq L$.

By using the same reasoning as in Section B.4 we obtain

**Theorem 24.** The expected complexity of Algorithm 6 with line 4 implemented as in Algorithm 13 is $O((t_{bc}(n, m) + b_{bc}(n, m) + n \log n) \cdot \log^4 n)$.

When combined with Theorem 3, this yields Theorem 1.

### A Proofs for Section 2

#### A.1 Proof of Lemma 6

We will need the following well-known fact about minimum cuts. It follows, for example, from the properties of the parametric maxflow problem [CGTS9].

**Lemma 25.** Consider graph $G$ and pairs of disjoint subsets $(S, T)$ and $(S', T')$ with $S \subseteq S'$ and $T \supseteq T'$. If $U$ is a minimum $S$-$T$ cut then there exists a minimum $S'$-$T'$ cut $U'$ with $U' \subseteq U$.

We proceed with the proof of Lemma 6.
Lemma 6 (restated). Suppose that \((\Omega, E)\) is an OC tree for \(\varphi_u \text{ and } v \in \varphi\) is the parent of \(u\) in \((\varphi_u, E)\). Suppose further that \((\Omega^{-u}, E^{-u})\) is an OC tree for \((\varphi, G)\). Then \((\Omega, E)\) is valid for \(G\) if and only if set \([u]\) is a minimum \(v\)-\(u\) cut in \(G[[v]^{-u}; v]\).

Proof. Clearly, we have \([w]^\downarrow = [w]^{-u}\) for all \(w \in \varphi\) and \([w]^\downarrow = [u]\) for \(u = w\). Therefore, \((\Omega, E)\) is valid for \(G\) if and only if \([u]\) is a minimum \(v\)-\(u\) cut in \(G\); since \(\varphi \cap [v]^{-u} = \{v\}\), it suffices to show that \(T \subseteq [v]^{-u}\).

Let us write \(\varphi = \alpha \varphi \ldots \). We claim that \(T \subseteq [v]^\downarrow = [v]^{-u}\). Indeed, assume that \(|\alpha| \geq 1\) (otherwise \([v]^\downarrow = V\) and the claim is trivial). Since \([v]^\downarrow\) is a minimum \(\alpha\)-\(v\) cut and \(u \in [v]^\downarrow\), set \([v]^\downarrow\) is also a minimum \(\alpha\)-\(\{v, u\}\) cut. Applying Lemma 25 gives that \(T \subseteq [v]^\downarrow\), as desired.

Now consider node \(w \in \varphi\) with \(w < v\). Let us write \(\varphi = \alpha \varphi \beta \ldots\). Let \(U\) be a minimum \(\alpha \varphi \beta \ldots\) cut. We know that \([w]^\downarrow = [w]^{-u}\) is a minimum \(\alpha \varphi \beta \ldots \) cut. Define \(A = [w]^\downarrow - U\) and \(B = U - [w]^\downarrow\). By symmetry of cuts and submodularity, we have

\[
\text{cost}(\{w\}^\downarrow) + \text{cost}(U) = \text{cost}(\{w\}^\downarrow) + \text{cost}(V - U) \\
\geq \text{cost}(\{w\}^\downarrow \cap (V - U)) + \text{cost}(\{w\}^\downarrow \cup (V - U)) \\
= \text{cost}(A) + \text{cost}(V - B) = \text{cost}(A) + \text{cost}(B)
\]

Clearly, we have \(\alpha \varphi \beta \cap A = \alpha \varphi \beta \cap B = \emptyset\), \(w \not\in A\) and \(u \in B\). Set \(B\) is a minimum \(\alpha \varphi \beta\) cut, and thus \(\text{cost}(A) \geq \text{cost}(\{w\}^\downarrow)\). This implies that \(\text{cost}(B) \leq \text{cost}(U)\), and thus \(B\) is a minimum \(\alpha \varphi \beta\) cut. Lemma Lemma 25 gives that \(T \subseteq B\). We have \(B \cap [w]^\downarrow = \emptyset\) and thus \(T \cap [w]^\downarrow = \emptyset\). Since this holds for all \(w \in \varphi\) with \(w < v\) (and since \(T \subseteq [v]^\downarrow\)), we conclude that \(T \subseteq [v]^{-u}\), as desired.

\(\square\)

A.2 Proof of Lemma 8

Part (b) follows directly from part (a) and Lemma 4, so we focus on proving part (a). Consider OC tree \((\Omega, E)\) for sequence \(\varphi\) with \(|\varphi| \geq 2\), and let \(t \in \varphi\) be a leaf node in \((\varphi, E)\). We say that \(t\) is a free leaf in \((\varphi, E)\) if the following holds: if \(\varphi = \ldots t \ldots u \ldots\) then \(t\) and \(u\) have different parents. In particular, the last node of \(\varphi\) is a free leaf if \(|\varphi| \geq 2\). We will show the following result.

Lemma 26. Suppose that \((\Omega, E)\) is an OC tree for \((\varphi, G)\) and \(t \in \varphi\) is a free leaf in \((\varphi, E)\). Then \((\Omega^{-t}, E^{-t})\) is an OC tree for \((\varphi^{-t}, G)\).

Proof. We use induction on \(|\varphi|\). If \(|\varphi| = 2\) then the claim is trivial; suppose that \(|\varphi| > 2\). Clearly, for any \(w \in \varphi^{-t}\) we have \([w]^{-t} = [w]^\downarrow\).

Let \(u\) be the last node of \(\varphi\). It can be seen that \((\Omega^{-u}, E^{-u})\) is an OC tree for \(\varphi^{-u}\) (if \(\varphi^{-u} = \alpha \varphi \ldots\) then set \([w]^{-u} = [w]^\downarrow\) is indeed a minimum \(\alpha\)-\(w\) cut). Thus, if \(t = u\) then the claim holds. Suppose that \(\varphi = \ldots u \ldots\). Note that \(u\) are leaf nodes with distinct parents. Denote \(\varphi^{-tu} = (\varphi^{-u})^{-t}\) and \((\Omega^{-tu}, E^{-tu}) = ((\Omega^{-u})^{-t}, (E^{-u})^{-t}) = ((\Omega^{-t})^{-u}, (E^{-t})^{-u})\). Clearly, \(t\) is a free leaf in \((\varphi^{-u}, E^{-u})\), and thus \((\Omega^{-tu}, E^{-tu})\) is an OC tree for \((\varphi^{-tu}, G)\) by the induction hypothesis. Let \(v\) be the parent of \(u\) in \((\varphi, E)\). We have \([u] = [u]^{-t}\) and \([v]^{-u} = [v]^{-tu}\) since \(t\) is a leaf node which is not a child of \(v\).

\((\Omega^{-u}, E^{-u})\) is an OC tree for \((\varphi^{-u}, G)\) and \((\Omega, E)\) is an OC tree for \((\varphi, G)\). By Lemma 6, \([u]\) is a minimum \(v\)-\(u\) cut in \(G[[v]^{-u}; v]\).

\((\Omega^{-tu}, E^{-tu})\) is an OC tree for \((\varphi^{-tu}, G)\) and \([u]^{-t}\) is a minimum \(v\)-\(u\) cut \(G[[v]^{-tu}; v]\). By Lemma 6, \((\Omega^{-t}, E^{-t})\) is an OC tree for \((\varphi^{-t}, G)\).
construction, $u$ is always in $\varphi$. We claim that the sequence $\pi^*(u)$ never changes. Indeed, it suffices to show that each step preserves $\pi(u)$; applying this fact to $\pi(\pi(u)), \pi(\pi(\pi(u))), \ldots$ will then yield the claim. It suffices to consider the case when $u$ comes after $t$, i.e. $\varphi = s \ldots t \ldots u$. Let $v$ be the parent of $u$. We say that node $w \in \varphi$ is $u$-admissible in $(\varphi, E)$ if it satisfies the following conditions: (i) $w \subseteq u$; (ii) $w \in \{v\} \cup \{w : w \in E\}$. Recall that $\pi(u)$ is the maximal $u$-admissible node $w$ in $(\varphi, E)$ w.r.t. $\subseteq$. We cannot have $w = t$, since $t$ is a free leaf. It can be checked that a node $w \neq t$ is $u$-admissible in $(\varphi, E)$ if and only if it is $u$-admissible in $(\varphi^{-t}, E^{-t})$; this implies the claim.

Denote $\psi = \pi^*(u)u$. We will show next that upon termination we have $\varphi = \psi$; clearly, this will imply Lemma 8(a). Suppose the claim is false, then $\varphi = s \ldots t \alpha$ where $t \not\in \psi$ and $\alpha \subseteq \psi$. If node $t$ has a child $w$ then $w \in \psi$ and thus $t \in \pi^*(w) \subseteq \psi$ - a contradiction. Thus, $t$ is a leaf in $(\varphi, E)$. It is not a free leaf since the algorithm has terminated, thus there exists $w \in \alpha$ such that $t$ and $w$ have the same parent. But then $t \in \pi^*(w) \subseteq \psi$ - a contradiction.

A.3 Proof of Lemma 9

**Lemma 9** (restated). Consider sequence $\varphi = \alpha \ldots$ in graph $G$ with $|\alpha| \geq 1$. Suppose that $(\Omega^\varphi, E^\varphi)$ is an $\mathcal{OC}$ tree for $(\alpha, G)$, and for each $v \in \alpha$ pair $(\Omega^v, E^v)$ is an $\mathcal{OC}$ tree for $(\varphi \cap [v]^*; G[[[v]^*]; v])$. Then $(\Omega, E)$ is an $\mathcal{OC}$ tree for $(\varphi, G)$ where $\Omega = \bigcup_{v \in \alpha} \Omega^v$ and $E = E^{\varphi} \cup \bigcup_{v \in \alpha} E^v$.

We use induction on $|\varphi|$. If $\varphi = \alpha$ then the claim is trivial. Suppose that $\varphi = \alpha \beta u$. Let $v \in \alpha$ be the unique node with $u \in [v]^*$, and denote $\gamma = \beta \cap [v]^*$. We will work with pairs $(\Omega^v, E^v)$ and $((\Omega^v)-u, (E^v)-u)$, which are $\mathcal{OC}$ trees for sequences $\alpha \beta$ and $v \gamma$ respectively. Clearly, we have $[w] = [w]^v$ for $w \in \gamma u$ and $[w]-u = ([w]^v)-u$ for $w \in \gamma$ (where $([w]^v)-u$ is the component of $(\Omega^v)-u$ to which $w$ belongs). Let $p \in v \gamma$ be the parent of $u$ in $(\varphi, E)$ (equivalently, in $(v \gamma u, E^{v\gamma})$). Denote $H = G[[[v]^*]; v]$. By Lemma 26, $((\Omega^v)-u, (E^v)-u)$ is an $\mathcal{OC}$ tree for $(v \gamma, H)$. Thus, the induction hypothesis gives that $(\Omega^\gamma, E^\gamma)$ is an $\mathcal{OC}$ tree for $(\alpha \beta, G)$. By Lemma 8 $[u]^\gamma = [u]$ is a minimum $p-u$ cut in $H(((p)^v)-u; p) = G[[p]^u; v]$. By Lemma 6 $(\Omega, E)$ is an $\mathcal{OC}$ tree for $(\varphi, G)$.

A.4 Proof of Lemma 10

**Lemma 10** (restated). Consider sequence $\varphi = s \alpha \ldots$ in graph $G$ with $|\alpha| \geq 1$. Let $(S, T)$ be a minimum $s-a cut$ in $G$, and let $T_s = T \cup \{s\}$. Suppose that $(\Omega', E')$ is an $\mathcal{OC}$ tree for $(\varphi \cap S, G[S; s])$ and $(\Omega'', E'')$ is an $\mathcal{OC}$ tree for $(\varphi \cap T_s, G[T_s; s])$. Then $(\Omega, E)$ is an $\mathcal{OC}$ tree for $(\varphi, G)$ where $\Omega = \{[s]^\varphi \cup [s]^\alpha\} \cup (\Omega' - \{[s]^\alpha\}) \cup (\Omega'' - \{[s]^\alpha\})$ and $E = E' \cup E''$.

First, let us consider node $u \in \varphi \cap S$ with $u \neq s$. We can write $\varphi = s \alpha \beta u \ldots$, since $\alpha \cap S = \varnothing$. We need to show that $[u]^* = \min s \alpha u$-cut. If $|\alpha| = 1$ then the claim follows from Lemma 9 (since $(\Omega^\varphi, E^\varphi) = ([S, T], \{\alpha s\})$ is an $\mathcal{OC}$ tree for $(s \alpha, G)$). The general case can be reduced to the case above by contracting nodes in $\alpha$ to a single node. Such transformation preserves set $S$ in a minimum $s-a cut$ $(S, T)$; thus, $\mathcal{OC}$ tree $(\Omega', E')$ for $(\varphi \cap S, G[S; s])$ is not affected, and therefore set $[u]^*$ is also not affected.

Now consider node $u \in \varphi \cap T$. Let us write $\varphi = \beta u \ldots$, and denote $\beta \cap S = A$ and $\beta \cap T = B$. For brevity, we denote $X_1 \ldots X_k = X_1 \cup \ldots \cup X_k$ for disjoint subsets $X_1, \ldots, X_k$; if $X_i \equiv \{x_i\}$ then we write $x_i$ instead of $X_i$. We make two claims:

- There exists a minimum $AB-u$ cut $(S', T')$ with $S \subseteq S'$. Indeed, $(S, T)$ is a minimum $s-(\alpha \cup \{u\})$ cut (since $u \in T$). The claim now holds by Lemma 25.

- $[u]^*$ is a minimum $SB-u$ cut in $G$. Indeed, $(\Omega'', E'')$ is an $\mathcal{OC}$ tree for $(\varphi \cap T_s, G[T_s; s])$, thus set $[u]^* = [u]^*_{SB}$ is a minimum $sB-u$ cut in $G[T_s; s]$. This is equivalent to the claim above.

These claims imply that $[u]^*$ is a minimum $AB$-cut, or equivalently a minimum $\beta-u$ cut.
A.5 Proof of Theorem 11: analysis of Algorithm 8

Computations performed by the algorithm can be represented by a rooted tree whose nodes have the form $\sigma = (\varphi; G)$. For such node we denote $\varphi_\sigma = \varphi$, and define $H_\sigma = (V_\sigma, E_\sigma)$ to be the graph obtained from $G$ by removing node $s$ and incident edges, where $s$ is the first node of sequence $\varphi$. The leaves $\sigma$ of this tree satisfy $|\varphi_\sigma| \leq 2$. Each non-leaf node $\sigma$ has a child $\tau$ corresponding to the recursive call at line 3 (we call the left child), and children $\{\tau^v : v \in \alpha\}$ corresponding to recursive calls at line 6 (we call them right children). Let us assign label $\lambda_\sigma = (\lambda_1(\sigma),\ldots,\lambda_d(\sigma)) \in \{0,1\}^d$ according to the following rules: (i) the label of the root node is the empty string; (ii) if $\sigma$ is a non-leaf node then its left child is assigned label $(\lambda(\sigma),0)$ and its right children are assigned label $(\lambda(\sigma),1)$. Let $d_\sigma$ be the depth of node $\sigma$, then $d_\sigma = |\lambda(\sigma)|$. The set of labels that appears during the algorithm will be denoted as $\Lambda \subseteq \{0,1\}^*$, and for a label $\mu \in \Lambda$ let $\Sigma_\mu$ be the set of nodes $\sigma$ with $\lambda(\sigma) = \mu$.

**Lemma 27.** The execution of $\text{OrderedCuts}(\varphi; G)$ satisfies the following properties.

(a) The maximum depth satisfies $d_{\text{max}} \leq \log_2 |\varphi| + O(1)$, and $|\Lambda| \leq O(|\varphi|)$.

(b) Consider non-leaf node $\sigma$, and let $\tau$ be its left child and $\tau_1,\ldots,\tau_k$ be its right children. Then $(V_\tau, E_\tau) = (V_\sigma, E_\sigma)$, $V_{\tau_1},\ldots,V_{\tau_k}$ are disjoint subsets of $V_\sigma$, and $E_{\tau_1},\ldots,E_{\tau_k}$ are disjoint subsets of $E_\sigma$.

(c) For each $\mu \in \Lambda$ sets in $\{V_\sigma : \sigma \in \Sigma_\mu\}$ and in $\{E_\sigma : \sigma \in \Sigma_\mu\}$ are disjoint.

Proof. By construction, if $\tau$ is a child of $\sigma$ then $|\varphi_\tau| \leq |\varphi_\sigma|/2 + 1$ if $\varphi_\sigma$ is even, and $|\varphi_\tau| \leq (|\varphi_\sigma| + 1)/2$ if $\varphi_\sigma$ is odd. Let $d_{\text{max}}$ be the maximum depth. For each non-leaf node $\sigma$ we have $|\varphi_\sigma| \geq 2^{d_{\text{max}} - d - 1} + 2$, therefore $d_{\text{max}} \leq \log_2 |\varphi| + O(1)$ where $\varphi$ is the input string. This implies that $|\Lambda| \leq 1 + 2^1 + \ldots + 2^{d_{\text{max}}} \leq O(|\varphi|)$, and proves part (a).

Part (b) follows from the algorithm’s construction. Part (c) follows from part (b) combined with a straightforward induction argument.

Lemma 27 gives the first part of Theorem 11 i.e. that the complexity of Algorithm 8 is $O(|\varphi| \cdot \text{hC}(n,m))$. We now focus on the second part. Let us define a generalized sequence $\varphi$ as a set of nodes together with a partial order $\sqsubseteq$ on this set. With some abuse of terminology the set of nodes in $\varphi$ will be denoted simply as $\varphi$. We say that $\varphi$ is a 0-sequence if $\sqsubseteq$ does not impose any relations on $\varphi$, and $\varphi$ is 1-sequence if there exists $s \in \varphi$ such that $s \sqsubseteq v$ for all $v \in \varphi - \{s\}$ and $\sqsubseteq$ does not impose any relations on $\varphi - \{s\}$. If $\varphi$ is a 1-sequence then we will write $\varphi = s\varphi'$ where $\varphi'$ is a 0-sequence.

Let us define a randomized algorithm $\text{OrderedCuts}'(\varphi; G)$ that takes as an input 1-sequence $\varphi = s\varphi'$, graph $G$ and returns triplet $(\Omega, \mathcal{E}, \sqsubseteq)$ where $\sqsubseteq$ is a partial order on $\varphi'$. This algorithm has the same structure as $\text{OrderedCuts}(\varphi; G)$ but with the following modifications:

- At line 2 the split $\varphi = \alpha\beta$ is replaced by the following operation: sample a random subset $A \subseteq \varphi'$ of size $|A| = \left\lceil \frac{1}{2}(|\varphi| + 1) \right\rceil - 1$, let $\alpha$ be the 1-sequence with $\alpha = sA$, and let $\beta$ be a 0-sequence with $\beta = \varphi - \alpha$.
- Replace lines 3 and 6 with “$(\Omega^o, \mathcal{E}^o, \sqsubseteq^o) \leftarrow \ldots$” and “$(\Omega^v, \mathcal{E}^v, \sqsubseteq^v) \leftarrow \ldots$” respectively.
- At line 1 replace output $(\Omega, \mathcal{E})$ with $(\Omega, \mathcal{E}, \sqsubseteq)$ where $\sqsubseteq$ is the given total order on $\varphi$.
- At line 8 return $(\Omega, \mathcal{E}, \sqsubseteq)$ where $\sqsubseteq$ copies all relations from $\sqsubseteq^o$ (that impose a partial order on $\alpha - \{s\}$), copies all relations from $\sqsubseteq^v$ for $v \in \alpha$ (that impose a partial order on $v(\beta \cap T)$), and imposes additionally relations $a \sqsubseteq b$ for all $a \in \alpha - \{s\}, b \in \beta$.

Suppose that $\text{OrderedCuts}'(s\varphi'; G)$ returns $(\Omega, \mathcal{E}, \sqsubseteq)$. The following claims can be easily verified by a straightforward induction:
(i) OrderedCuts\((s, v_1, \ldots, v_\ell; G)\) returns \((\Omega, \mathcal{E})\) for any permutation \(v_1, \ldots, v_\ell\) of \(\varphi'\) consistent with \(\subseteq\).

(ii) If \(v_1, \ldots, v_\ell\) is sampled as a uniformly random permutation of \(\varphi'\) consistent with \(\subseteq\) then \(v_1, \ldots, v_\ell\) is a uniformly random permutation of \(\varphi'\).

To prove the second part of Theorem 11 it thus suffices to show that the expected complexity of OrderedCuts\('\(\varphi; G)\) for a given 1-sequence \(\varphi\) is \(O(h_{qc}(n, m) \cdot \log |\varphi|)\). We use the same notation and terminology for analyzing OrderedCuts\('\(\varphi; G)\) that we defined in the beginning of this section for OrderedCuts\((\varphi; G)\). We make the following claim about the execution of OrderedCuts\('\(\varphi; G)\).

Lemma 28. (a) Consider non-leaf node \(\sigma\), and let \(\tau_1, \ldots, \tau_k\) be its right children. Then
\[\mathbb{E}[\sum_{i \in [k]} |V_{\tau_i}|] \leq \frac{1}{2}|V_{\sigma}|\] and
\[\mathbb{E}[\sum_{i \in [k]} |E_{\tau_i}|] \leq \frac{1}{4}|E_{\sigma}|.\]

(b) For each \(\mu \in \Lambda\) we have \(\mathbb{E}[|V_\mu|] \leq n \cdot \frac{1}{2} |\mu|\) and \(\mathbb{E}[|E_\mu|] \leq m \cdot \frac{1}{2} |\mu|\) where \(V_\mu = \bigcup_{\lambda(\sigma) = \mu} V_\sigma\), \(E_\mu = \bigcup_{\lambda(\sigma) = \mu} E_\sigma\), \(n = |V_G|\), \(m = |E_G|\), and \(|\mu|\) is the number of 1’s in \(\mu\).

Proof. Part (b) is a straightforward consequence of part (a) (and Lemma 27(b)), so we focus on proving part (a). Consider the call OrderedCuts\('\(s\varphi'; G)\) with graph \(G = (V, E, w)\) corresponding to node \(\sigma\). Define \(\ell = |\varphi'|\), \(p = \left\lceil \frac{1}{2} |\varphi| + 1 \right\rceil - 1\) and \(q = \ell - p\), then \(p \geq q\). Define sets \(A = \alpha - \{s\}\) and \(B = \beta\). Recall that \(A\) is a random subset of \(\varphi'\) of size \(p\), and \(B = \varphi' - A\).

Let \(A'\) be a random subset of \(A\) of size \(q\), and let \(T^*\) be the minimal minimum \(A'-B\) cut in \(G\). We claim that \(\mathbb{E}[|V_{G[T^*]}|] \leq \frac{1}{2}|V|\) and \(\mathbb{E}[|E_{G[T^*]}|] \leq \frac{1}{4}|E|\). Indeed, for disjoint subsets \(X, Y \subseteq \varphi'\) of size \(q\) let \(T^*_{X,Y}\) be the minimal minimum \(X-Y\) cut in \(G\). Clearly, sets \(T^*_{X,Y}\) and \(T^*_{Y,X}\) are disjoint. Conditioned on the event \(\{A', B\} = \{X, Y\}\), we have \(\{A', B\} = (X, Y)\) with prob. \(\frac{1}{2}\) and \(\{A', B\} = (Y, X)\) with prob. \(\frac{1}{2}\). Therefore,
\[\mathbb{E}[|V_{G[T^*]}|] \mid \{A', B\} = \{X, Y\}] = \frac{1}{2}|V_{G[T^*_{X,Y}]}| + \frac{1}{2}|V_{G[T^*_{Y,X}]}| \leq \frac{1}{2}|V|\]
Summing this over pairs \((X, Y)\) with appropriate probabilities gives the desired bound on \(\mathbb{E}[|V_{G[T^*]}|]\).

The bound on \(\mathbb{E}[|E_{G[T^*]}|]\) can be obtained in a similar way.

Now consider node \(v \in \alpha\), and let \((S, T)\) be the cut computed at line 5 for node \(v\). \((S, T)\) can be equivalently defined as the minimum \(A_v-B_v\) cut in \(G\) with minimal \(T\) where \(A_v = (V - \{v\}) \cup \{v\}\) and \(B_v = B \cap \{v\}\). We have \(A' \subseteq A_v\) and \(B_v \subseteq B\), so by Lemma 25 we have \(T \subseteq T^*\). This implies \(H_{\tau_v}\) is a subgraph of \(G[T^*]\) where \(\tau_v\) is the right child of \(\sigma\) corresponding to node \(v\), and thus proves the claim.

\[\square\]

Using Lemma 28(b), we can bound the expected total number of nodes in edges as follows:
\[\sum_{\mu \in \Lambda} \mathbb{E}[|V_\mu|] \leq n \cdot \sum_{\mu \in \Lambda} \left(\frac{1}{2}\right)^{|\mu|}\]
\[\sum_{\mu \in \Lambda} \mathbb{E}[|E_\mu|] \leq m \cdot \sum_{\mu \in \Lambda} \left(\frac{1}{2}\right)^{|\mu|}\]

Theorem 11 now follows from the following calculation:
\[\sum_{\mu \in \Lambda} \left(\frac{1}{2}\right)^{|\mu|} \leq \sum_{d=0}^{\max} \sum_{\mu \in \{0, 1\}^d} \left(\frac{1}{2}\right)^{|\mu|} = \sum_{d=0}^{\max} \sum_{k=0}^{d} \frac{d!}{k!} \left(\frac{1}{2}\right)^k = \sum_{d=0}^{\max} (1 + \frac{1}{2})^d\]
\[= 2((\frac{1}{2})^{\max+1} - 1) \leq \left(\frac{3}{2}\right)^{\log_2 n + O(1)} = O(n^{\log_2(3/2)})\]
B Proofs for Section 3

B.1 Proof of Lemma 13

Lemma 13 (restated). Let $(\Omega, \mathcal{E})$ be an $\mathcal{OC}$ tree for $(\varphi, G)$. Define $\psi = \varphi$, and let us modify tuple $(\psi, \Omega, \mathcal{E})$ using the following algorithm: while $(\psi, \mathcal{E})$ has a leaf node of depth two or larger, pick the rightmost such node $u \in \psi$ and replace $(\psi, \Omega, \mathcal{E})$ with $(\psi-u, \Omega-u, \mathcal{E}-u)$. Then the resulting pair $(\Omega, \mathcal{E})$ is an $\mathcal{OC}_1$ tree for $(\varphi, G)$.

Proof. We will use Lemma 26. In the light of this lemma, it suffices to show the following: if $(\Omega, \mathcal{E})$ is an $\mathcal{OC}$ tree for sequence $\varphi$ and $t$ is the rightmost leaf of $(\varphi, \mathcal{E})$ that has depth two or larger then $t$ is a free leaf in $(\varphi, \mathcal{E})$. Suppose not, then $\varphi = \ldots t \ldots u \ldots$ where $t, u$ have the same parent. Thus, $u$ has depth two or larger. Let $v$ be a leaf of the subtree of $(\varphi, \mathcal{E})$ rooted at $u$, then $v$ also has depth two or larger. Furthermore, either $v = u$ or $\varphi = \ldots t \ldots u \ldots v \ldots$ This contradicts the choice of $t$. \qed

B.2 Analysis of Algorithm 9

Note that each update at line 7 preserves the property $S \subseteq X$, since the sets in $\{S_v : v \in S\}$ are disjoint. This means, in particular, that the update $X := X - (S_v - \{v\})$ is equivalent to the update $X := (X - S_v) \cup \{v\}$.

It can be seen that the output of Algorithm 9 satisfies property (a) of Theorem 16; this follows from the definition of $\text{OrderedCuts}_1$ and from Lemma 4. We will show next that there exists sequence $\alpha_1, \ldots, \alpha_N$ with $N = \Theta(\log^3 |X|)$ so that property (b) of Theorem 16 is satisfied. In the remainder of the analysis we assume that Assumption 1 holds.

We let $\lambda_i, X_i, Y_i$ for $i \in [N+1]$ to be the variables at the end of the $i$-th iteration, and $\lambda_0, X_0$ be the variables after initialization at line 1. Note that $Y_i = \text{RandomSubset}(X_{i-1}; \alpha_i)$ for $i \in [N+1]$. We always use letter $X$ to denote the original set $X = X_0$.

We will need some additional notation. Recall that $C_{sv}$ for $v \in X$ is the minimum $s$-$v$ cut in $G$ (which is now unique), and the family $\{C_{sv} : v \in X\}$ is laminar. We write $u \sim v$ for nodes $u, v \in X$ if $C_{su} = C_{sv}$. Clearly, $\sim$ is an equivalence relation on $X$. Let $\hat{\Phi}$ be the set of equivalence classes of this relation with $\langle \Phi \rangle = X$, and for $v \in X$ let $[v] \in \Phi$ be the class to which $v$ belongs. We write $[u] \leq [v]$ if $C_{su} \subseteq C_{sv}$. Clearly, $\leq$ is a partial order on $\Phi$. Furthermore, laminarity of cuts $C_{sv}$ implies that $\leq$ can be represented by a rooted forest $F$ on nodes $\Phi$ so that $A \leq B$ for $A, B \in \Phi$ iff $A$ belongs to the subtree of $F$ rooted at $B$. We denote $\text{Roots}(F)$ to be the set of roots of this forest. For node $A \in \Phi$ we denote $A^\downarrow = \{B \in \Phi : B \leq A\}$; equivalently, $A^\downarrow$ is the set of nodes of the subtree of $F$ rooted at $A$. It can be seen that $C_{sv} \cap X = ([v]^\downarrow)$ for $v \in X$. We define function $f: \Phi \rightarrow \mathbb{R}$ via $f([v]) = \text{cost}(C_{sv}) = f(s, v)$ for $v \in X$. Assumption 1 implies that $f(B) < f(A)$ for $B < A$.

Lemma 29. For each $R \in \text{Roots}(F)$ and $i \in [N]$ there holds $R \cap X_i \neq \emptyset$.

Proof. We use induction on $i$ (for a fixed $R \in \text{Roots}(F)$). Consider the $i$-th iteration, and let $S$ be the $\mathcal{OC}_1$ tree computed at line 5. We make the following claim:

- If $v \in S$ and $v \notin R$ then $S_v \cap R = \emptyset$. Indeed, conditions $v \notin R$ and $R \in \text{Roots}(F)$ imply that $C_{sv} \cap R = \emptyset$.

By the definition of $\mathcal{OC}_1$ tree, set $S_v$ is a minimum $sv$ cut for some sequence $\alpha$. Lemma 25 gives that $S_v \subseteq C_{sv}$, implying the claim.

The claim of Lemma 29 now easily follows (recalling that the update $X := X - (S_v - \{v\})$ at line 7 is equivalent to the update $X := (X - S_v) \cup \{v\}$). \qed
Consider node \( R \in \Phi \). We say that subset \( \Psi \subseteq \Phi \) is \( R \)-rooted if it can be represented as \( \Psi = R^1 - \bigcup_{A \in \Phi} A^i \) for some subset \( \Psi \subseteq R^i \). The minimal set \( \tilde{\Psi} \) in this representation will be denoted as \( \Psi^- \); clearly, such minimal \( \tilde{\Psi} \) exists, and equals the set of maximal elements of \( R^i - \Psi \) w.r.t. \( \preceq \). One exception is the set \( \Psi = \varnothing \): for such \( \Psi \) we define \( \Psi^- = \{ R \} \) (node \( R \) will always be clear from the context). It can be seen that if \( \Psi \) is \( R \)-rooted and \( A, B \) are distinct nodes in \( \Psi^- \) then sets \( A^i, B^i \) are disjoint.

Now consider subsets \( Y \subseteq X \), \( \Psi \subseteq \Phi \) and node \( v \in X \). Borrowing terminology from \cite{LP21}, we say that \( Y \) \emph{hits} \( v \) in \( \Psi \) if \( C_{sv} \cap \langle \Psi \rangle \cap Y = \{ v \} \). We denote

\[
\Psi[Y] = \Psi - \bigcup_{v: Y \text{ hits } v \text{ in } \Psi} [v]^i
\]

For subsets \( Y_1, \ldots, Y_k \) we also denote \( \Psi[Y_1, \ldots, Y_k] = \Psi[Y_1] \ldots [Y_k] \). Clearly, if \( \Psi \) is \( R \)-rooted then so is \( \Psi[Y_1, \ldots, Y_k] \). With this notation in place, we proceed with the analysis of Algorithm \( \text{Alg} \).

**Lemma 30.** Consider root \( R \in \text{Roots}(F) \), and for \( i \in \{ 1, \ldots, N \} \) denote \( \Psi_i = R^1[Y_i, \ldots, Y_i] \), so that \( R^1 = \Psi_0 \supseteq \Psi_1 \supseteq \ldots \supseteq \Psi_N \). Then the following holds for every \( A \in \Psi_i^-, i \in \{ 1, \ldots, N \} \):

- \(|\langle A^i \rangle \cap X_i| \leq 1\). If \( \langle A^i \rangle \cap X_i = \{ v \} \) then \( v \in A \) and \( \lambda_i(v) = f(A) \).

**Proof.** We use induction on \( i \). If \( i = 0 \) then \( \Psi_0^- = \varnothing \) and the claim is vacuous. Suppose it holds for \( i - 1 \); let us prove for \( i \in \{ 1, \ldots, N \} \).

Note that \( \Psi_i = \Psi_{i-1}[Y_i] \). Consider \( A \in \Psi_i^- \). If \( A \in \Psi_{i-1}^- \) then the claim holds by the induction hypothesis and the facts that \( X_i \subseteq X_{i-1} \) and \( \mu_i(v) \leq \mu_{i-1}(v) \) for all \( v \). Suppose that \( A \notin \Psi_{i-1}^- \). Condition \( A \in \Psi_i^- = (\Psi_{i-1}[Y_i])^- \) then implies that there exists \( v \in A \) such that \( Y_i \) hits \( v \) in \( \Psi_{i-1}^- \), i.e. \( C_{sv} \cap \langle \Psi_{i-1} \rangle \cap Y_i = \{ v \} \). We then have \( A = [v] \in \Psi_{i-1}^- \).

Let \( \varphi = sov \ldots \) be the ordering of nodes in \( \{ s \} \cup Y_i \) used in the algorithm. We claim that \( \alpha \cap C_{sv} = \varnothing \). Indeed, suppose that there exists \( u \in \alpha \cap C_{sv} \subseteq Y_i \cap C_{sv} \), then we must have \( u \notin \langle \Psi_{i-1} \rangle \) and hence \( [u] \notin \Psi_{i-1}^- \). Condition \( u \in C_{sv} \) implies that \( [u] \preceq [v] = A \). Conditions \( A \in \Psi_{i-1}^- \), \( [u] \preceq A \), \( [u] \notin \Psi_{i-1}^- \) imply that there exists \( B \in \Psi_{i-1}^- \) with \( [u] \preceq B \prec A \). We have \( u \in \langle B^i \rangle \) and \( u \in Y_i \subseteq X_i \). The induction hypothesis gives that \( \langle B^i \rangle \cap X_i = \{ u \} \), \( u \in B \) and \( \lambda_{i-1}(u) = f(B) \). We obtain that \( \lambda_{i-1}(u) = f(B) < f(A) \leq \lambda_{i-1}(v) \) and so \( u \) should come after \( v \) in \( \varphi \) - a contradiction.

Let \( S = \text{OrderedCuts}(\varphi; G) \) be the \( \text{OC}_i \) tree computed at iteration \( i \). By the definition \( \text{OC}_i \) tree, there exists \( u \in (av) \cap S \) such that \( S_u \) is a minimum \( sa' \)-u cut for some \( \alpha' \subseteq \alpha - \{ u \} \), and \( v \in S_u \). As shown above, we have \( \alpha' \cap C_{sv} = \varnothing \). By construction, we have \( X_i \subseteq X_{i-1} - \{ S_u \} \cap X_i \). Recall that \( C_{sv} \cap X = \langle A^i \rangle \) and so \( \langle A^i \rangle \cap X_i \subseteq C_{sv} \cap X_i \). We now consider two possible cases.

- \( u = v \). The conditions above imply that \( S_v = C_{sv} \), and so \( \langle A^i \rangle \cap X_i \subseteq C_{sv} \cap X_i \subseteq \{ v \} \).

Furthermore, if \( v \in X_i \) then we will have \( \lambda_i(v) \leq \text{cost}(S_v) = f(A) \).

- \( u \neq v \). Since \( \alpha' \cap C_{sv} = \varnothing \), set \( C_{sv} \) is also (the unique) minimum \( sa' \)-v cut. Recall that \( S_u \) is a minimum \( sa' \)-u cut and \( v \in S_u \), and so \( S_u \) is a minimum \( sa' \)-uv cut. Lemma \( 25 \) gives that \( C_{sv} \subseteq S_u \). Condition \( \alpha \cap C_{sv} = \varnothing \) implies that \( u \notin C_{sv} \) and so \( C_{sv} \subseteq S_u - \{ u \} \). Thus, we will have \( \langle A^i \rangle \cap X_i \subseteq C_{sv} \cap X_i = \varnothing \).

In both cases the claim of Lemma \( 30 \) holds.

**Lemma 31.** Suppose that \( R^1[Y_1, \ldots, Y_N] = \varnothing \) for all \( R \in \text{Roots}(F) \). Then the output of Algorithm \( \text{Alg} \) satisfies \( X \subseteq \langle S^* \rangle \).

\[ \square \]
To summarize, we showed that \( X_N \subseteq \langle \text{Roots}(\mathcal{F}) \rangle \), and for each \( R \in \text{Roots}(\mathcal{F}) \) there exists unique \( v \in R \cap X_N \); this \( v \) satisfies \( \lambda_N(v) = f(R) \). Let \( \varphi = \{v_1, \ldots, v_t\} \) be the ordering of nodes in \( Y_N = X_N \) used in the algorithm, and let \( \mathcal{S} = \text{OrderedCuts}_1(\varphi; G) \) be the computed \( \mathcal{O}_1 \) tree for \( (\varphi, G) \). By the definition of \( \mathcal{O}_1 \) tree, for each \( v \in \mathcal{S} \) set \( \mathcal{S}_v \) is a minimum \( sa-v \) cut for some \( \alpha \subseteq X_N - \{v\} \). We have \( \alpha \cap C_{su} = \emptyset \), and thus \( \mathcal{S}_v = C_{sv} \). We must also have \( \mathcal{S} = X_N \) (if there exists \( v \in X_N - \mathcal{S} \) then \( v \in \mathcal{S}_u = C_{su} \) for some \( u \in \mathcal{S}, u \neq v \) - a contradiction). Since \( \lambda(v_1) \geq \ldots \geq \lambda(v_t) \), we will have \( \text{cost}(\mathcal{S}_{v_1}) \geq \ldots \text{cost}(\mathcal{S}_{v_t}) \) and so line 9 outputs set \( \mathcal{S}^* = \mathcal{S} \). The claim follows.

Our next goal is to construct sequence \( \alpha_1, \ldots, \alpha_N \) such that the precondition of Lemma 31 holds w.h.p. For that we will use the argument from [LP21]. First, we show the following.

**Lemma 32.** Let \( \Psi \) be an \( R \)-rooted set for some \( R \in \Phi \), and let \( Z_0, \ldots, Z_d \) be a sequence of independent subsets generated via \( Z_i \leftarrow \text{RandomSubset}(X, 2^{-i}) \) where \( d = \lfloor \log_2 |X| \rfloor \). Then \( \mathbb{E}[|\Psi[Z_0, \ldots, Z_d]|] \leq \left( 1 - \frac{\Theta(1)}{1 + \Theta} \right) |\langle \Psi \rangle| = \left( 1 - \frac{\Theta(1)}{\log X} \right) |\langle \Psi \rangle| \).

**Proof.** Define directed multigraph \( (\langle \Psi \rangle, P) \) as follows: for every \( i \in [0, d] \) and every \( v \in \langle \Psi \rangle \) such that \( Z_i \) hits \( v \) in \( \Psi \) add edges \( \{(v, u) : u \in C_{sv} \cap \langle \Psi \rangle\} \) to \( P \). (We say that these edges are at level \( i \)). We will show the following:

(a) The expected out-degree of each \( v \in \langle \Psi \rangle \) is \( \Omega(1) \).
(b) The in-degree of each \( u \in \langle \Psi \rangle \) is at most \( d + 1 \).
(c) If \( (v, u) \in P \) then \( u \notin \Psi[Z_0, \ldots, Z_d] \).

Part (a) will imply that \( P \) has \( \Omega(|\langle \Psi \rangle|) \) edges in expectation; combined with (b), this will mean that the expected number of nodes \( u \in \langle \Psi \rangle \) with at least one incoming edge is \( \Omega(|\langle \Psi \rangle|/(d + 1)) \).

Part (c) will then give the claim of the lemma. We now focus on proving (a)-(c).

**Part (a)** Denote \( n_v = |C_{sv} \cap \langle \Psi \rangle| \). There must exist \( i \in [0, d] \) such that \( n_v \in [2^i, 2^{i+1}] \). The probability that \( Z_i \) hits \( v \) in \( \Psi \) equals \( 2^{-i} \cdot (1 - 2^{-i})^{n_v-1} = \Theta(2^{-i}) \). If this event happens then \( n_v \) edges of the form \( (v, u), u \in C_{sv} \cap \langle \Psi \rangle \) are added to \( P \). Thus, the expected number of outgoing edges from \( v \) at level \( i \) is at least \( \Theta(2^{-i}) \cdot n_v = \Theta(1) \).

**Part (b)** Suppose that \( P \) contains edges \( (v, u), (v', u) \) with \( v \neq v' \) at the same level \( i \). \( Z_i \) hits both \( v \) and \( v' \) in \( \Psi \), so \( u \in C_{sv} \cap C_{sv'} \) and thus by laminarity either \( C_{sv} \subseteq C_{sv'} \) or \( C_{sv'} \subseteq C_{sv} \). Assume w.l.o.g. that \( C_{sv'} \subseteq C_{sv} \). Then \( C_{sv} \cap \langle \Psi \rangle \cap Z_i \) contains at least two nodes \( (v, v') \), which is a contradiction.

**Part (c)** Denote \( \Psi_i = \Psi[Z_1, \ldots, Z_i] \) for \( i \in [0, d] \), and consider edge \( (v, u) \in P \) at level \( i \). Let \( A = [v] \in \mathcal{R}^i \). We claim that \( A \notin \Psi_i \) (implying that \( A \notin \Psi_d \) and thus \( u \notin \langle \Psi \rangle_d \) since \( u \in \langle \mathcal{R}^i \rangle = C_{sv} \cap \langle X \rangle \)). Indeed, assume that \( A \in \Psi_i \subseteq \Psi_{i-1} \). Since \( Z_i \) hits \( v \) in \( \Psi \) we have \( C_{sv} \cap \langle \Psi \rangle \cap Z_i = \{v\} \). This means that \( C_{sv} \cap \langle \Psi_{i-1} \rangle \cap Z_i = \{v\} \), i.e. \( Z_i \) hits \( v \) in \( \Psi_{i-1} \). This in turn implies that \( A \notin \Psi_{i-1}[Z_i] = \Psi_i \), which is a contradiction.

**Corollary 33.** Denote \( n = |X| \). For every fixed polynomial \( p(n) \) there exists a sequence \( \alpha_1, \ldots, \alpha_N \) with \( N = \Theta(\log^3 n) \) such that the precondition of Lemma 31 holds with probability at least \( 1 - \frac{1}{p(n)} \).

**Proof.** Let \( \alpha_1, \ldots, \alpha_N \) be the sequence obtained by repeating the sequence used in Lemma 32 (i.e. \( 2^{-0}, 2^{-1}, \ldots, 2^{-d} \) \( K \) times, so that \( N = K(d + 1) \)). If \( K = \Theta(\log n) \) (with an appropriate constant) then for any fixed \( R \in \Phi \) we will have \( \mathbb{E}[|\langle R^i[Y_1, \ldots, Y_N] \rangle|] \leq \frac{1}{n^{p(n)}} \), and thus \( R^i[Y_1, \ldots, Y_N] = \emptyset \) with probability at least \( 1 - \frac{1}{n^{p(n)}} \). The precondition of Lemma 31 has at most \( n \) events of the form above, so by the union bound they will all hold with probability at least \( 1 - \frac{1}{p(n)} \).
This concludes the proof of Theorem 16.

B.3 Proof of Lemma 19

We can assume that the modification of $H$ at line 3 produces graph $H'$ with a unique GH tree (this happens with probability $\Omega(1)$), and so Assumption 3 always holds. We have, in particular, $C_{uv} = V_H - C_{vu}$ for any $u, v \in X$.

Denote $n = |X|$, and for $s \in X$ let $\text{rank}(s) \in [n]$ be the rank of $s$ according to the reverse of total order $\sqsubseteq$ (so that the largest node $s$ w.r.t. $\sqsubseteq$ satisfies $\text{rank}(s) = 1$). Clearly, if node $s$ is updated at line 7 then we must have $s \sqsubseteq v$, i.e. every such update strictly decreases the rank of $s$.

For $p \in [d]$ let “$p$-stage” be the sequence of iterations $i$ that use $\beta_i = 2^{-p}$. We say the $p$-stage is successful if at the end of this stage node $s$ satisfies $\text{rank}(s) \leq 2^{p-1}$ (or if the algorithm has terminated earlier). We claim that the $p$-stage is successful with probability at least $1 - \frac{1}{2^3}$ assuming that either $p = d$ or $p \in [d-1]$ and $(p + 1)$-stage is successful. Indeed, consider a single iteration (lines 5-8) with $\beta_i = 2^{-p}$. If $\text{rank}(s) \leq 2^{p-1}$ then the claim holds; we can thus assume that $\text{rank}(s) \in (2^{p-1}, 2^p]$. Let $Y \subseteq X - \{s\}$ be the set sampled at this iteration. The following events will hold jointly with probability at least $1 - \gamma$ for some constant $\gamma > 0$: $Y \cap \{v \in X : \text{rank}(v) \leq 2^{p-1}\} \neq \emptyset$ and $Y \cap \{v \in X : 2^{p-1} < \text{rank}(v) \leq 2^p\} = \emptyset$. The first event implies that this iteration will update $s$, and the second event implies that the new $s$ will satisfy $\text{rank}(s) \leq 2^{p-1}$. The $p$-stage has $K$ iterations, and thus it will be successful with probability at least $1 - \gamma^K = 1 - \gamma^{\Theta(\log \log n)} \geq 1 - \frac{1}{2^3}$ with an appropriate constant in the $\Theta(\cdot)$ notation.

There are $d$ such stages in total, so all of them will be successful with probability at least $\frac{1}{2^3}$. If this happens, $s$ becomes the largest node in $X$ w.r.t. $\sqsubseteq$, and so the last iteration (with $\beta_M = 1$) will produce named partition $S$ with $X - \{s\} \sqsubseteq \langle S \rangle$.

B.4 Proof of Theorem 20

In this section we analyze the overall complexity of Algorithm 6. Each supernode $X$ that appears during the execution of can be assigned a depth using natural rules: (i) the initial supernode $V$ is at depth 0; (ii) if supernode $X$ at depth $d$ is split into supernodes $X_0, \ldots, X_k$ then the latter supernodes are assigned depth $d + 1$. By construction, in the latter case we have $|X_i| \leq |X|/2$ for $i \in [0, k]$. Therefore, supernode $X$ at depth $d$ satisfies $|X| \leq n \cdot 2^{-d}$, and so the maximum depth is $O(\log n)$. We will need the following result from [AKT20].

Lemma 34 ([AKT20] Lemma 3.12). Let $X_1, \ldots, X_{k(d)}$ be the supernodes at depth $d$, and let $H_1, \ldots, H_{k(d)}$ be the corresponding auxiliary graphs. Then the total number of edges in $H_1, \ldots, H_{k(d)}$ is $O(m)$.

An analogous result also holds for the number of nodes.

Lemma 35. Let $X_1, \ldots, X_{k(d)}$ be the supernodes at depth $d$, and let $H_1, \ldots, H_{k(d)}$ be the corresponding auxiliary graphs. Then the total number of nodes in $H_1, \ldots, H_{k(d)}$ is $O(n)$.

Proof. Consider graph $H = H_i$ corresponding to supernode $X = X_i$. Recall that $V_H = X \cup \{v_Y : XY \in T\}$ where $T$ is the partition tree at the moment when $X$ is processed. Clearly, sets $X_1, \ldots, X_{k(d)}$ are disjoint, so we need to bound the total number of nodes of $H$ due to edges $XY \in T$ (let us call them “contracted nodes”). After $X$ is processed, each $XY$ is transformed to an edge $X'Y$ for some supernode $X' \subset X$, other edges of $T$ are kept intact (or “transformed to themselves”), and some new edges are added to $T$. Let us track how edge $XY$ transforms during the algorithm; eventually, it becomes an edge $xy$ of the final GH tree, with $x \in X$ and $y \in Y$. Let us charge node $v_Y$ in graph $H$ to node $x$. Clearly, no other node in graphs $H_1, \ldots, H_{k(d)}$ is charged to $x$. The number of edges in the final GH tree is $n - 1$, and thus the total of contracted nodes in $H_1, \ldots, H_{k(d)}$ is at most $2(n - 1)$. □
By putting all results together, we obtain Theorem 20.

C Proofs for Section 4

C.1 Proof of Lemma 21

Lemma 21 (restated). Let $(\lambda, C^*)$ be the output of CertifiedOrderedCuts$(s, v_1, \ldots, v_k; G)$, and denote $Y = \{v_1, \ldots, v_k\}$. Pair $(\lambda, C^*)$ has the following properties.

(a) Subsets in $C^*$ are disjoint, and each $S \subseteq C^*$ is a minimum $s$-$v$ cut for some $v \in Y$.

(b) If $Y \cap C_{sv_k} = \emptyset$ and $f(s, v_k) = \min_{i \in [k]} f(s, v_i)$ then $C^*$ contains a minimum $s$-$v_k$ cut.

(c) There exists a family of cuts $C \subseteq \{S \subseteq V - \{s\} \mid \lambda(S) \geq f(s, v)\}$ such that

(i) $\lambda(v) = \min_{C \subseteq \mathcal{V} : v \in C} \lambda(S)$ for each $v \in V - \{s\}$, and hence $\lambda(v) \geq f(s, v)$;

(ii) if $\{v_1, \ldots, v_k\} \cap C_{sv_k} = \emptyset$ then $C$ contains a minimum $s$-$v_k$ cut (and hence $\lambda(v_k) = f(s, v_k)$).

Proof. We take $C = \{S_k : k \in [\ell]\}$ to be the set of cuts considered inside the call to OrderedCuts$(\cdot)$, then the condition in (c.i) holds by the definition of OrderedCuts. We claim that for each $v_k \in Y^*$ we have $\lambda(v_k) = f(s, v_k)$. Indeed, let $S$ be a minimum $s$-$v_k$ cut and let $i$ be the largest index in $[k]$ such that $\{s, v_1, \ldots, v_{i-1}\} \subseteq V - S$, then $v_i \in S$ and so $f(s, v_k) = \lambda(S) \geq f(\{s, v_1, \ldots, v_{i-1}\}, s) \geq \lambda(v_i) \geq \lambda(v_k)$, implying the claim. (The middle inequality holds by the definition of OrderedCuts and the last inequality holds since $v_k \in Y^*$). We are now ready to prove the lemma.

Part (a) Sets $S_v$ at line 3 (and thus sets in $C^*$) are disjoint by the definition of IsolatingCuts. Set $S_v$ is added to $C^*$ only if $v \in Y^*$ and $\lambda(S_v) = \lambda(v) = f(s, v)$, and so the claim holds.

Parts (b) and (c.ii) Consider $k \in [\ell]$ with $\{v_1, \ldots, v_k\} \cap C_{sv_k} = \emptyset$. The condition implies that $C_{sv_k}$ is a minimum $\{s, v_1, \ldots, v_{k-1}\}$-$v_k$ cut and hence $f(\{s, v_1, \ldots, v_{k-1}\}, v_k) = f(s, v_k)$. Therefore, set $S_k \in C$ must satisfy $\lambda(S_k) \leq f(s, v_k)$, i.e. $S_k$ is a minimum $s$-$v_k$ cut. This proves part (c.ii). Now suppose that $Y \cap C_{sv_k} = \emptyset$ and $f(s, v_i) \geq f(s, v_k)$ for each $i \in [k]$. We have $\lambda(v_k) = f(s, v_k)$ and $\lambda(v_i) \geq f(s, v_i)$ for $i \in [k]$, therefore node $v_k$ will be added to $Y^*$. We also have $Y^* \cap C_{sv_k} = \emptyset$, and so set $C_{sv_k}$ returned by IsolatingCuts$(s, Y^*; G)$ is a minimum $s$-$v_k$ cut. This set satisfies $\lambda(C_{sv_k}) = \lambda(v_k) = f(s, v_k)$, and so $S_{v_k}$ will be added to $C^*$.

C.2 Analysis of Algorithm 12

By Lemma 21(a), set $\Pi$ in Algorithm 12 always satisfies $\Pi \subseteq \tilde{\Pi}^L$. For the rest of analysis we assume that all outputs of CertifiedOrderedCuts are minimal, and thus $\Pi \subseteq \Pi^L$. We will show next that there exists sequence $\alpha_1, \ldots, \alpha_{2N}$ with $N = \Theta(\log^2 |X|)$ for which $\langle \Pi \rangle = \langle \Pi^L \rangle$ w.h.p..

We use the same notation as in Section 3.1 let us recap it briefly. We write $u \sim v$ for nodes $u, v \in X$ if $C_{su} = C_{sv}$. Let $\Phi$ be the set of equivalence classes of relation $\sim$, and for $v \in X$ let $[v] \in \Phi$ be the class to which $v$ belongs. We write $[u] \preceq [v]$ if $C_{su} \subseteq C_{sv}$, then $\preceq$ is a partial order on $\Phi$ that can be represented by a rooted forest $F$ on nodes $\Phi$ so that $A \preceq B$ for $A, B \in \Phi$ if $A$ belongs to the subtree of $F$ rooted at $B$. For node $A \in \Phi$ we denote $A^\downarrow = \{B \in \Phi : B \preceq A\}$.

Consider node $R \in \Phi$. We say that subset $\Psi \subseteq \Phi$ is $R$-rooted if it can be represented as $\Psi = R^\downarrow - \bigcup_{A \in \Psi} A^\downarrow$ for some subset $\Psi \subseteq R^\downarrow$. The minimal set $\Psi$ in this representation will be denoted as $\Psi^-$. One exception is the set $\Psi = \emptyset$: for such $\Psi$ we define $\Psi^- = \{R\}$ (node $R$ will always be clear from the context).

Consider subsets $Y \subseteq X$, $\Psi \subseteq \Phi$ and node $v \in \langle \Psi \rangle$. We say that $Y$ hits $v$ in $\Psi$ if $C_{sv} \cap (\Psi) \cap Y = \{v\}$. We denote

$$
\Psi[Y] = \Psi - \bigcup_{v : \text{Y hits } v \text{ in } \Psi} [v]^\downarrow
$$

For subsets $Y_1, \ldots, Y_k$ we also denote $\Psi[Y_1, \ldots, Y_k] = \Psi[Y_1] \ldots [Y_k]$. 

We can assume that subset $Y$ in line 3 is generated by first sampling $\hat{Y} \leftarrow \text{RandomSubset}(X; \alpha_i)$ and then taking $Y = \hat{Y} - \langle \Pi \rangle$. Let $\Pi_i, \lambda_i, C_i^\ell, \mu_i, \hat{Y}_i, Y_i$ for $i \in [2N]$ be the variables at the end of the $i$-th iteration, and $\Pi_0, \mu_0$ be as initialized at line 1. We refer to iterations $i = 1, \ldots, N$ as the first stage, and to iterations $i = N + 1, \ldots, 2N$ as the second stage. The lemma below analyzes the first stage.

**Lemma 36.** Suppose that $R^{\ell}[\hat{Y}_1, \ldots, \hat{Y}_N] = \emptyset$ for $R \in \Phi$. Then $\mu_N(v) = f(s, v)$ for all $v \in R - \langle \Pi_N \rangle$.

**Proof.** Denote $\Psi_i = R^{\ell}[\hat{Y}_1, \ldots, \hat{Y}_i]$, then $R^{\ell} = \Psi_0 \supseteq \Psi_1 \supseteq \ldots \supseteq \Psi_N = \emptyset$. Let us define function $f : \Phi \rightarrow \mathbb{R}$ via $f([v]) = \text{cost}(C_{sv}) = f(s, v)$ for $v \in X$. We make the following claim:

(*) For each $i \in [0, N]$ and $A \in \Psi_i^-$ there holds $\mu_i(v) \leq f(A)$ for $v \in \langle A^\ell \rangle - \langle \Pi_i \rangle$.

To prove this, we use induction on $i$. If $i = 0$ then $\Psi_i^- = \emptyset$ and the claim is vacuous. Suppose it holds for $i - 1$; let us prove for $i \in [N]$. Note that $\Psi_i = \Psi_{i-1}[\hat{Y}_i]$. Consider $A \in \Psi_i^-$. If $A \in \Psi_{i-1}$ then for each $v \in \langle A^\ell \rangle - \langle \Pi_i \rangle \subseteq \langle A^\ell \rangle - \langle \Pi_{i-1} \rangle$ we have $\mu_i(v) \leq \mu_{i-1}(v) \leq f(A)$ where the last inequality is by the induction hypothesis. We thus suppose that $A \notin \Psi_{i-1}^-$. Condition $A \in \Psi_i^- = (\langle Y_i \rangle - \langle \Pi_i \rangle)^-$ then implies that there exists $a \in A$ such that $\hat{Y}_i$ hits $a$ in $\Psi_{i-1}$, i.e. $C_{sa} \cap (\langle \Psi_{i-1} \rangle \cap \hat{Y}_i) = \{a\}$. We then have $A = [a] \in \Psi_{i-1}$.

First, suppose that $a \notin Y_1$. Then $a \in \langle \Pi_i \rangle$ (since $a \in Y_1$ and $Y_1 = \hat{Y}_i - \langle \Pi_i \rangle$) and hence $A \subseteq \langle \Pi_i \rangle$ and $\langle A^\ell \rangle \subseteq \langle \Pi_i \rangle$, so (*) vacuously holds. Next, suppose that $\mu_{i-1}(a) = f(s, a) = f(A)$. If $C_{sa} = \langle A^\ell \rangle = \{a\}$ then (*) clearly holds. Otherwise $\mu_0(a) < f(s, a)$ and hence $\lambda_j(a) = f(s, a)$ for some $j \in [i - 1]$. This means that set $C_j$ computed in the $j$-th call to $\text{CertifiedOrderedCuts}$ contained set $S$ which is a minimum $s-a$ cut. We have $C_{sa} \subseteq S$, and hence $\mu_{i-1}(v) \leq \lambda_j(v) \leq \text{cost}(S) = f(s, a)$ for all $v \in C_{sa} \cap X = \langle A^\ell \rangle$, i.e. (*) holds.

It remains to consider the case when $a \in Y_i$ and $\mu_{i-1}(a) > f(s, a) = f(A)$. We claim that all $v \in (C_{sa} \cap Y_i) - \{a\}$ satisfy $\mu_{i-1}(v) \leq f(A)$. Indeed, we have $C_{sa} \cap (\langle \Psi_{i-1} \rangle \cap Y_i) = \{a\}$, and so $v \notin \langle \Psi_{i-1} \rangle$. Furthermore, $v \in C_{sa} \cap X = \langle A^\ell \rangle$ and $A \in \Psi_{i-1}$. These conditions imply that there exists $B \in \Psi_{i-1}^-$ with $B \subseteq A$ (implying $f(B) \leq f(A)$) and $v \in \langle B^\ell \rangle$. By the choice of $Y_i$ at line 3 we have $Y_i \cap \langle \Pi_{i-1} \rangle = \emptyset$, and so $v \notin \langle \Pi_{i-1} \rangle$ and $v \in \langle B^\ell \rangle - \langle \Pi_{i-1} \rangle$. By the induction hypothesis, $\mu_{i-1}(v) \leq f(B)$, which gives the claim.

Now consider the $i$-th step of Algorithm 12. As we just showed, we have $\mu_{i-1}(v) \leq f(A) < \mu_{i-1}(a)$ for all $v \in (C_{sa} \cap Y_i) - \{a\}$. Thus, nodes in $Y_i$ will be sorted as $v_1, \ldots, v_{k-1}, a, \ldots$ where $\{v_1, \ldots, v_{k-1}\} \cap C_{sa} = \emptyset$. Lemma 21(c) now gives that $\mu_i(v) \leq \lambda_i(v) \leq \text{cost}(C_{sa}) = f(A)$ for all $v \in C_{sa} \cap X = \langle A^\ell \rangle$.

We have proved claim (*). Let us use it for $i = N$. We have $\Psi_N = \emptyset$, so $\Psi_N^- = \{R\}$ and thus $\mu_N(v) \leq f(R)$ for all $v \in R - \langle \Pi_N \rangle$. We also have $\mu_N(v) \geq f(R)$ for such $v$, which gives the lemma.

We say that the first stage is successful if the precondition of Lemma 36 holds for every $R \in \Phi$. By the lemma, if the first stage is successful then $\mu_N(v) = f(s, v)$ for all $v \in X - \langle \Pi_N \rangle$.

**Lemma 37.** Suppose that the first stage is successful and $R^{\ell}[\hat{Y}_{N+1}, \ldots, \hat{Y}_{2N}] = \emptyset$ for $R \in \Phi$ with $|\langle R^\ell \rangle| \leq L$. Then $\langle R^\ell \rangle \subseteq \langle \Pi_{2N} \rangle$.

**Proof.** For $i \in [N, 2N]$ denote $\Psi_i = R^{\ell}[\hat{Y}_{N+1}, \ldots, \hat{Y}_i]$, then $R^{\ell} = \Psi_N \supseteq \Psi_{N+1} \supseteq \ldots \supseteq \Psi_{2N} = \emptyset$. We make the following claim:

(*) For each $i \in [N, 2N]$ and $A \in \Psi_i^-$ there holds $\langle A^\ell \rangle \subseteq \langle \Pi_i \rangle$. 

\[\square\]
To prove this, we use induction on $i$. If $i = N$ then $\Psi_i^{-} = \emptyset$ and the claim is vacuous. Suppose it holds for $i - 1$; let us prove for $i \in [N, 2N]$. Note that $\Psi_i = \Psi_{i-1}[\hat{Y}_i]$. Consider $A \in \Psi_i^{-}$. If $A \in \Psi_{i-1}^{−}$ then $\langle A^i \rangle \subseteq \langle \Pi_{i-1} \rangle \subseteq \langle \Pi_{i} \rangle$ where the first inclusion is by the induction hypothesis.

We thus suppose that $A \notin \Psi_{i-1}^{-}$. Condition $A \in \Psi_{i-1}^{−}$ implies that there exists $a \in A$ such that $\hat{Y}_i$ hits $a$ in $\Psi_{i-1}$, i.e. $C_{sa} \cap (\Pi_{i-1}) \cap \hat{Y}_i = \{a\}$. We then have $A = [a] \in \Psi_{i-1}$.

First, suppose that $a \notin \hat{Y}_i$. Then $a \in (\Pi_{i})$ (since $a \notin \hat{Y}_i$ and $\hat{Y}_i = \hat{Y}_i - (\Pi_{i})$) and hence $A \subseteq (\Pi_{i})$ and $\langle A \rangle \subseteq (\Pi_{i})$, i.e. $(*)$ holds. We thus assume from now on that $a \in \hat{Y}_i$. We claim that $C_{sa} \cap Y_i = \{a\}$. Indeed, suppose there exists $v \in (C_{sa} \cap Y_i) - \{a\}$. We have $C_{sa} \cap (\Pi_{i-1}) \cap \hat{Y}_i = \{a\}$ and thus $v \notin (\Pi_{i-1})$. Furthermore, $v \in C_{sa} \cap X = \langle A \rangle$ and $A \in \Psi_{i-1}$. These conditions imply that there exists $B \in \Psi_{i-1}$ with $B \leq A$ and $v \in \langle B \rangle$. By the induction hypothesis, $\langle B \rangle \subseteq (\Pi_{i-1})$ and so $v \in (\Pi_{i-1})$. By the choice of $Y_i$ at line 3 we thus have $v \notin Y_i$, which is a contradiction.

Now consider the $i$-th step of Algorithm \[12\] Let $v_1, \ldots, v_{k-1}, a, \ldots$ be the sorting of $Y_i$. Since the first stage is successful and nodes in $Y_i$ are sorted according to $\mu_i(\cdot)$, we have $f(s, v_1) \geq \ldots f(s, v_{k-1}) \geq f(s, a)$. Thus, the precondition of Lemma \[21\] holds for node $v_k = a$, and hence set $C_{sa}$ contains $C_{sa}$. We have $C_{sa} \cap X = \langle A \rangle \subseteq \langle R \rangle$ and hence $|C_{sa} \cap X| \leq L$, therefore $C_{sa}$ will be added to $\Pi_i$ at line 7 and hence $(*)$ holds.

We have proved claim $(*)$. Let us use it for $i = 2N$. We have $\Psi_{2N} = \emptyset$, so $\Psi_{2N}^{-} = \{R\}$ and thus $\langle R \rangle \subseteq (\Pi_{2N})$, as claimed.  

\[ \]

It remains to construct sequence $\alpha_1, \ldots, \alpha_{2N}$ such that the preconditions of Lemmas \[36\] and \[37\] hold w.h.p. For that we use the same technique as in Section 3.1, which is based on Lemma \[32\]. Applying the same argument as in Corollary \[33\] gives the following result, and concludes the proof of Theorem \[22\].

**Corollary 38.** Denote $n = |X|$. For every fixed polynomial $p(n)$ there exists a sequence $\alpha_1, \ldots, \alpha_{2N}$ with $N = \Theta(\log^3 n)$ such that the preconditions of Lemmas \[36\] and \[37\] hold with probability at least $1 - \frac{1}{p(n)}$.

**References**

[AIS+16] Takuya Akiba, Yoichi Iwata, Yosuke Sameshima, Naoto Mizuno, and Yosuke Yano. Cut tree construction from massive graphs. In *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM)*, pages 775–780, 2016.

[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 *FOCS*, 2022.

[AKT20] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Cut-equivalent trees are optimal for min-cut queries. In *FOCS*, 2020. Full version: arXiv:2009.06090.

[AKT21a] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. APMF < APSP? Gomory-Hu tree for unweighted graphs in almost-quadratic time. In *FOCS*, pages 1135–1146, 2021.

[AKT21b] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Subcubic algorithms for Gomory-Hu tree in unweighted graphs. In *STOC*, pages 1725–1737, 2021. Full version at arXiv:2012.10281.

[AKT22] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Friendly cut sparsifiers and faster Gomory-Hu trees. In *SODA*, pages 3630–3649, 2022.
A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. An o(mn) Gomory-Hu tree construction algorithm for unweighted graphs. In STOC, pages 605–614, 2007.

H. N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. In FOCS, pages 812–821, 1991.

G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. SIAM J. Computing, 18:30–55, 1989.

R. E. Gomory and T. C. Hu. Multi-terminal network flows. Journal of the Society for Industrial and Applied Mathematics, 9:551–570, 1961.

A. V. Goldberg and K. Tsioutsiouliklis. Cut tree algorithms: an experimental study. Journal of Algorithms, 38(1):51–83, 2001. Code: https://web.archive.org/web/20140115171857/http://www.cs.princeton.edu/~kt/cut-tree/experiments/.

J. Hao and J. B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. Algorithmica, 17:424–446, 1994.

Vladimir Kolmogorov. A computational study of Gomory-Hu tree construction algorithms. arXiv:2204.10169v3, August 2022.

Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. Fair cuts, approximate isolating cuts, and approximate gomory-hu trees in near-linear time. arXiv:2203.00751, March 2022.

Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic maxflows. In FOCS, 2020.

Jason Li and Debmalya Panigrahi. Approximate Gomory-Hu tree is faster than \( n - 1 \) max-flows. In STOC, pages 1738–1748, 2021.

Jason Li, Debmalya Panigrahi, and Thatchaphol Saranurak. A nearly optimal all-pairs min-cuts algorithm in simple graphs. In FOCS, pages 1135–1146, 2021.

K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. Combinatorica, 7(1):105–113, 1987.