Fast Algorithms for Graph Arboricity and Related Problems

Ruoxu Cen*, Henry Fleischmann†, George Z. Li†, Jason Li† and Debmalya Panigrahi*
*Department of Computer Science
Duke University
Durham, United States
Emails: {ruoxu.cen, debmalya}@duke.edu
†Computer Science Department
Carnegie Mellon University
Pittsburgh, United States
Emails: {hfleisch, gzli, jmli}@cs.cmu.edu

***Abstract*—We give an algorithm for finding the arboricity of a weighted, undirected graph, defined as the minimum number of spanning forests that cover all edges of the graph, in $\sqrt{n}m^{1+o(1)}$ time. This improves on the previous best bound of $\tilde{O}(nm)$ for weighted graphs and $\tilde{O}(m^{3/2})$ for unweighted graphs (Gabow 1995) for this problem. The running time of our algorithm is dominated by a logarithmic number of calls to a directed global minimum cut subroutine – if the running time of the latter problem improves to $m^{1+o(1)}$ (thereby matching the running time of maximum flow), the running time of our arboricity algorithm would improve further to $m^{1+o(1)}$.**

We also give a new algorithm for computing the entire *cut hierarchy* – laminar multiway cuts with minimum cut ratio in recursively defined induced subgraphs – in $mn^{1+o(1)}$ time. The cut hierarchy yields the ideal edge loads (Thorup 2001) in a fractional spanning tree packing of the graph which, we show, also corresponds to a max-entropy solution in the spanning tree polytope. For the cut hierarchy problem, the previous best bound was $\tilde{O}(n^2m)$ for weighted graphs and $\tilde{O}(nm^{3/2})$ for unweighted graphs.

***Index Terms*—arboricity, cut hierarchy, ideal edge loads, tree packing**

I. Introduction

Consider an undirected graph $G = (V, E)$ with n vertices and m edges with integer edge weights $c_e \in \{1, 2, \ldots, C\}$. The *arboricity* of the graph (see, e.g., Schrijver [1]) is defined as the minimum number of disjoint forests that cover all the edges of the graph. (The notion of coverage is that each edge belongs to exactly as many forests as its weight.) Picard and Queyranne [2] initiated the algorithmic study of arboricity and showed that the problem is polynomial-time solvable for $O(1)$ edge capacities. This was later extended to general edge capacities by Padberg and Wolsey [3]. The running time of these algorithms are large polynomials. A more efficient algorithm was obtained by Gabow and Westermann [4], whose running time was $\widetilde{O}(\min(m^{5/3}, mn))$ when $m = \Omega(n \log n)$, and slightly more for sparser graphs.[1] This was further improved in the work of Gabow [5], who leveraged parametric directed minimum cut algorithms to obtain the arboricity of a weighted graph in $\tilde{O}(mn)$ time and that of an unweighted graph in $\tilde{O}(m^{3/2})$ time, respectively. If approximation is allowed, faster algorithms are known. Eppstein [6] obtained a linear-time algorithm for a 2-approximation of arboricity. In their famous work on positive LPs, Plotkin, Shmoys, and Tardos [7] gave an FPTAS for finding the arboricity of a graph, whose running time was later improved in [8], [9]. Nevertheless, for computing the arboricity of a graph exactly, the 30-year-old algorithm of Gabow [5] remains the state of the art.

A closely related problem is that of computing the edge loads in an *ideal tree packing*, a paradigm introduced by Thorup [10], [11]. To define this, recall that a *multiway cut* of G (denoted $\mathcal{P}$) is a vertex partition $V = V_1 \sqcup V_2 \sqcup \ldots \sqcup V_k$ for any $k \geq 2$, where each V_i is said to be a *side* of $\mathcal{P}$. (If $k = 2$, then $\mathcal{P}$ has only two sides, and is simply a cut of G.) The *cut ratio* of $\mathcal{P}$ is defined as $\frac{d(\mathcal{P})}{|\mathcal{P}|-1}$, where $d(\mathcal{P}) := \sum_{e \in \partial\mathcal{P}} c_e$ is the sum of weights of all edges with endpoints in two different sides of $\mathcal{P}$ (call this set of edges $\partial\mathcal{P}$), and $|\mathcal{P}|$ is the number of sides of $\mathcal{P}$. A multiway cut that attains the minimum cut ratio across all the multiway cuts in a graph is called a *minimum ratio cut*. Now, consider the following recursive algorithm. Find a min ratio cut $\mathcal{P}$ and set $x_e = \frac{|\mathcal{P}|-1}{d(\mathcal{P})}$ for all edges $e \in \partial\mathcal{P}$, i.e., the reciprocal of the minimum cut ratio. Then, we recurse on the induced subgraphs for each non-singleton component after removing the edges $\partial\mathcal{P}$. The resulting hierarchy of laminar multiway cuts on induced subgraphs form the *cut hierarchy* of the graph and the values x_e on the edges are called the *ideal loads*. As observed by de Vos and Christiansen [12], the arboricity of a graph is the reciprocal of the minimum ideal load on an edge of the graph. Therefore, we can obtain the arboricity of a graph as a by-product of computing the ideal loads of all edges.

The cut hierarchy is implicit in the definition of ideal tree packing by Thorup [10], [11]. It is also equivalently defined by Trubin [13] (referred to as HS-decomposition; HS stands for homogeneously strong). Trubin claimed an efficient algorithm for computing the cut hierarchy, and therefore,

[1] Throughout the paper, we use $\widetilde{O}(\cdot)$ to hide $\text{polylog}(n)$ factors.

the ideal loads. However, we show in Appendix B that this algorithm is incorrect. In order to construct the cut hierarchy, we observe that arboricity is related to a minimal vertex set in the laminar family defined by cut hierarchy. Given this observation, the previous best algorithm for computing the cut hierarchy and ideal loads is via $O(n)$ calls to Gabow's algorithm for arboricity [5], which has an overall running time of $\widetilde{O}(mn^2)$ for weighted graphs and $\widetilde{O}(m^{3/2}n)$ for unweighted graphs. We also know of a greedy tree packing algorithm to approximate the ideal loads up to small additive error in near-linear time [10].

Before describing our results, we note some other implications of computing the cut hierarchy and the ideal loads. Just as arboricity is the reciprocal of the minimum ideal load, the reciprocal of the *maximum* ideal load is called the *strength* of the graph (see, e.g., Schrijver [1]). This is also equal to the minimum cut ratio among all the multiway cuts of the graphs, and is denoted $\sigma(G) := \min_{\text{multiway cut } \mathcal{P} \text{ of } G} \frac{d(\mathcal{P})}{|\mathcal{P}|-1}$. Our cut hierarchy algorithm also returns the strength of the graph, and its running time matches, up to lower order terms, the best previous bound for this problem of $\widetilde{O}(nm)$. This latter bound is achieved by running a binary search over the values of strength, using an algorithm of Barahona [14] that can decide whether the strength is > 1 in $\widetilde{O}(nm)$ time. A closely related parameter is the *packing number* of a graph, which is the maximum number of spanning trees that can be packed into the graph. By a classical result of Nash-Williams [15] and Tutte [16], the packing number of a graph is equal to the integral floor of the strength of the graph $\lfloor \sigma \rfloor$. In fact, the entire set of ideal loads yield a *fractional spanning tree* of the graph. We show that this fractional tree defined by the ideal loads is precisely the *entropy maximizer* in the spanning tree polytope (see Section VI), i.e., it is an optimal solution to the convex program $\max_{\mathbf{x} \in \mathbb{T}} \sum_{e \in E} x_e \ln 1/x_e$, where $\mathbb{T}$ is the spanning tree polytope of the graph.

A. Our Results

Our first result is a new algorithm to compute the cut hierarchy, and therefore ideal loads, of a weighted, undirected graph. The algorithm is randomized and its running time is dominated by $\widetilde{O}(n)$ calls[2] to any maximum flow algorithm on *directed* graphs with $O(m)$ edges and $O(m)$ vertices. Using the current state of the art max-flow algorithms [17], [18], this results in a running time of $nm^{1+o(1)}$. This improves on the previous best algorithm that calls Gabow's arboricity algorithm [5] $O(n)$ times, and therefore has a running time of $\widetilde{O}(mn^2)$ for weighted graphs and $\widetilde{O}(m^{3/2}n)$ for unweighted graphs.

Theorem I.1. *There is a randomized algorithm that, given an undirected graph G on m edges and n vertices with integer edge weights in $\{1, 2, \ldots, C\}$, outputs the cut hierarchy (and hence ideal edge loads) of G in $\widetilde{O}(nm \log C)$ time plus $\widetilde{O}(n \log C)$ calls to a maximum flow subroutine on directed graphs with $O(m)$ edges and $O(m)$ vertices and integer edge weights at most $\text{poly}(n, C)$.*

[2]In our formal theorems, we make the dependence on C explicit, but in our informal descriptions, we assume that C is polynomially bounded in n for brevity.

Recall that arboricity is given by the reciprocal of the minimum ideal load. Therefore, the above theorem yields an algorithm for computing arboricity in the same running time of $mn^{1+o(1)}$, which is slightly slower than the previous best bounds for this problem. We improve this bound further by giving a randomized algorithm for arboricity whose running time is dominated by $\widetilde{O}(1)$ calls to two subroutines: any maximum flow algorithm on *directed* graphs with $O(m)$ edges and $O(m)$ vertices, and any global minimum cut algorithm on *directed* graphs with $O(m)$ edges and $O(n)$ vertices. Using the current state of the art maximum flow algorithms [17], [18] and global minimum cut algorithms in directed graphs [19], this results in a running time of $\sqrt{n}m^{1+o(1)}$. This improves on the previous best algorithm for arboricity [5] that has a running time of $\widetilde{O}(mn)$ for weighted graphs and $\widetilde{O}(m^{3/2})$ for unweighted graphs. Moreover, if the running time of directed minimum cut improves to almost linear time, it will immediately improve the running time of the arboricity algorithm to almost linear time as well.

Theorem I.2. *There is a randomized algorithm that, given an undirected graph G on m edges and n vertices with integer edge weights in $\{1, 2, \ldots, C\}$, outputs the arboricity of G in $O(m \log(nC))$ time plus $O(\log(nC))$ calls to a maximum flow subroutine on directed graphs with $O(m)$ edges and $O(m)$ vertices and $O(\log(nC))$ calls to a global minimum cut subroutine on directed graphs with $O(m)$ edges and $O(n)$ vertices, both with integer edge weights at most $\text{poly}(n, C)$.*

B. Our Techniques

Recall that in defining the cut hierarchy, we used a top down recursive definition where the algorithm repeatedly computes a minimum ratio cut on an induced subgraph. In designing our cut hierarchy algorithm, however, we take a bottom up approach. We start with a multiway cut on an induced subgraph at the bottom layer of the hierarchy, and end with the minimum ratio cut that appears at the top level of the hierarchy. Note that any multiway cut at the bottom level of the cut hierarchy must comprise singleton vertices as its sides, since the leaves of the cut hierarchy are the vertices of the graph. Such a vertex set whose induced subgraph appears at the bottom level of the cut hierarchy is called a *star set*. Suppose we had an algorithm to identify a star set S. Then, we can use this algorithm to reveal the entire cut hierarchy as follows: Contract the star set S returned by the algorithm and recursively find the cut hierarchy of the contracted graph. In this recursively constructed cut hierarchy, S appears as a leaf since it is a vertex of the contracted graph. Now, return the cut hierarchy for the original graph where we replace leaf S with a star centered at S, whose leaves are the vertices in S.

But, how do we identify a star set? Intuitively, a minimum ratio cut splits the graph into induced components that are denser than the overall graph. Repeating this argument along

each branch of the cut hierarchy, we should expect that the star sets are denser than the induced subgraphs represented by their ancestors, and, since they are at the bottom of the hierarchy, there are no denser subgraphs contained in them. To make this formal, we introduce the notion of a *dense core*, which is an induced subgraph of a graph that is denser than induced subgraphs containing them, as well as induced subgraphs contained in them. Our main structural lemma connects these two concepts: we show that *every dense core in a graph is necessarily a star set of the cut hierarchy*.

We are left with the problem of identifying a dense core. A natural idea is to find the induced subgraph of maximum (rather than maximal) edge density. (Our notion of edge density is slightly different from the usual one, but this is a finer point that we ignore in this summary.) For this purpose, we use variants of previously known constructions due to Goldberg [20] and Gabow [5] that define analogous minimum cut problems in directed graphs to identify the densest subgraph. We could use an off-the-shelf directed min-cut algorithm to complete our algorithm, and this would suffice if there were close-to-linear time bounds for the directed min-cut problem. Unfortunately, the current fastest directed min-cut algorithm [19] is slower—it runs in $m^{1+o(1)}\sqrt{n}$ time. This still suffices if the resulting star set has $\Omega(\sqrt{n})$ vertices, since the running time of directed min-cut can be amortized over the size of the vertex set that we contract. But, what if the star set is only of size $o(\sqrt{n})$? To handle this possibility, we redesign the directed min-cut algorithm to run in $m^{1+o(1)}k$ time, where k is the size of the star set that we identify and contract. This redesign requires a careful binary search for the correct value of k, where for any guessed k, we give an $m^{1+o(1)}k$-time verification algorithm that either confirms that a vertex set is indeed a dense core if the guess is at least the true value of k, or identifies that the guess of k is too small.

Note that the cut hierarchy also yields the ideal loads for all the edges of the graph. This is because for any edge, its ideal load is simply the reciprocal of the cut ratio of the unique multiway cut containing the edge in the cut hierarchy. Once we have the ideal loads, we can return the reciprocal of the smallest ideal load, equivalently the maximum cut ratio among all multiway cuts in the cut hierarchy, as the arboricity of the graph. But, this takes $nm^{1+o(1)}$ time. Can we do better? From the intuition that density is higher at lower levels of the cut hierarchy, we have that arboricity is exactly the maximum over cut ratios of star sets in the cut hierarchy. Since we are building the cut hierarchy in a bottom-up manner, it is unnecessary to build the whole hierarchy for the purpose of computing arboricity. As alluded to above, we can identify the densest subgraph by calling a global minimum cut subroutine in an appropriately defined directed graph. We make this connection precise by giving a direct reduction from arboricity to the directed minimum cut problem–we show that it suffices to make $\mathrm{polylog}(n)$ directed minimum cut calls, plus $\mathrm{polylog}(n)$ maximum flow calls and $\widetilde{O}(m)$ time outside these calls, in order to compute arboricity. This yields an algorithm for computing arboricity in $\sqrt{n}m^{1+o(1)}$ time, the dominant cost being the running time of the directed minimum cut subroutine [19].

Roadmap. The key structural property relating star sets and dense cores appears in Section III. This is then used in our algorithm for the cut hierarchy and ideal loads, which is given in Section IV. The algorithm for arboricity is given in Section V. Finally, in Section VI, we relate the ideal loads to the entropy-maximizing fractional spanning tree. We use an adaptation of the global directed min cut algorithm [19] as a subroutine in our algorithm; the proof appears in Appendix A for completeness. Appendix B describes the algorithm of Trubin [13] for the cut hierarchy and shows that it is incorrect.

II. Preliminaries

A. Basic Notations for Graphs and Cuts

A *multiway cut* of an undirected graph $G = (V, E)$ is a partition of V, i.e., a disjoint collection of subsets of V such that their union is V. Each subset in this partition is called a *side* of the multiway cut. For a multiway cut $\mathcal{P}$, we use $\partial\mathcal{P}$ to denote the set of cut edges, i.e., edges whose two endpoints are in different sides of $\mathcal{P}$. Sometimes, we abuse the term multiway cut to refer to the edge set of a multiway cut. The *all-singleton cut* refers to the multiway cut that partitions V into singleton vertices, i.e., $\{\{v\} : v \in V\}$. A multiway cut with two sides $(S, V \setminus S)$ is a *cut*, and its edge set is denoted ∂S.

We denote edge weights $c : E \to \mathbb{R}^{\geq 0}$. We extend c to a set function $2^E \to \mathbb{R}^{\geq 0}$ by taking the sum. So, $c(\partial\mathcal{P}) = \sum_{e\in\partial\mathcal{P}} c(e)$ is the cut value of $\mathcal{P}$, and $c(E)$ is the total edge weight. We abbreviate $c(\partial\mathcal{P})$ by $d(\mathcal{P})$; for a cut S, we abbreviate $c(\partial S)$ by $d(S)$. We use a subscript to disambiguate functions defined on different graphs.

We use $G[S]$ to denote the induced subgraph on a vertex set S, and use $E[S]$ to denote the edge set of $G[S]$. Contracting a vertex set S means to replace S by a new node s, replace all edges $(u, v), u \in S, v \notin S$ by (s, v), and delete all edges in $E[S]$. We use G/S to denote the graph formed by contracting S from G. For $U \supseteq S$ or $U \cap S = \varnothing$, we use U/S to denote the result of U after contracting S, i.e., $U/S = (U \setminus S) \cup \{s\}$ if $U \supseteq S$ and $U/S = U$ if $U \cap S = \varnothing$. For a partition $\mathcal{P}$ of V and a set S contained in a side of $\mathcal{P}$, we use $\mathcal{P}/S$ to denote the corresponding partition in G/S, i.e., $\mathcal{P}/S = \{P/S : P \in \mathcal{P}\}$.

Rank. We use $r(\cdot)$ to denote the rank function of the graphic matroid on graph $G = (V, E)$. For any edge set $F \subseteq E$, we have $r(F) = |V| - \mathcal{C}(F)$, where $\mathcal{C}(F)$ is the number of connected components in the graph (V, F). The following properties are standard.

Fact II.1. *$r(\cdot)$ is submodular, i.e., for edge sets $A \subseteq B$ and X disjoint from B,*

$$r(B \cup X) - r(B) \leq r(A \cup X) - r(A)$$

Fact II.2. *Consider a graph $G = (V, E)$ and vertex subset $U \subseteq V$, and suppose a multiway cut $\mathcal{P}$ of G partitions U into*

$\{U_i\}_{i=1}^k$, and all induced subgraphs $G[U_i]$ as well as $G[U]$ are connected. Then,

$$r(E[U]) = |U| - 1, \quad r(E[U] \setminus \partial\mathcal{P}) = |U| - k.$$

It follows that

$$k - 1 = r(E[U]) - r(E[U] \setminus \partial\mathcal{P}).$$

B. Cut Ratio and Skew-Density

For any multiway cut $\mathcal{P}$, we call $\frac{d(\mathcal{P})}{|\mathcal{P}|-1}$ its *cut ratio*. Here, $|\mathcal{P}|$ represents the number of subsets in the partition, i.e., the number of sides of $\mathcal{P}$. A *minimum ratio cut* is a multiway cut with minimum cut ratio; its value is called the *strength* of the graph. An important property of a min-ratio cut is that all its sides are connected.

Fact II.3. *Suppose G is connected and $\mathcal{P}$ is the min-ratio cut of G. Then, the induced subgraph of each side $S \in \mathcal{P}$ is also connected.*

Proof. Assume for contradiction that some $S \in \mathcal{P}$ is disconnected. Let $\{S_i\}_{i=1}^k, k \geq 2$ be the connected components of S. By replacing S with $S_1, \ldots, S_k$ in $\mathcal{P}$, we get another multiway cut with the same cut edge set and strictly more sides. This implies the new cut will have strictly less ratio than $\mathcal{P}$, contradicting the assumption that $\mathcal{P}$ is a min ratio cut. □

For $S \subseteq V$ with $|S| \geq 2$, we define the *skew-density* of S as

$$\rho(S) = \frac{c(E[S])}{|S| - 1}.$$

In particular, skew-density of singleton sets or the empty set is defined to be 0. Because singleton subgraphs have no edges, for any nonempty S we have

$$c(E[S]) = \rho(S) \cdot (|S| - 1). \tag{1}$$

C. Cut Hierarchy

Consider the following recursive process. Given a connected graph G, find a min-ratio cut $\mathcal{P}$. By Fact II.3, all sides of $\mathcal{P}$ are connected, so we can recursively run the same process on the induced subgraphs of all sides of $\mathcal{P}$. The recursion terminates at singleton subgraphs. All sides of multiway cuts visited during this process form a laminar family. We call this family the *cut hierarchy* of G.

The cut hierarchy can be represented by a tree, where the root is V, and for each internal node U, all children of U represent a side of a min-ratio cut of $G[U]$. All leaves represent singleton sets.

Canonical Cut Hierarchy. Note that the cut hierarchy is not necessarily unique because the min-ratio cut may not be unique. To avoid ambiguity in our algorithm, we always choose the min-ratio cut with the most sides. We show that this min-ratio cut is unique, because of the following property:

Lemma II.4. *If $\mathcal{P}$ and $\mathcal{Q}$ are both min-ratio cuts, then the partition $\mathcal{W}$ formed by all connected components of $E \setminus (\partial\mathcal{P} \cup \partial\mathcal{Q})$ is also a min-ratio cut, and $\partial\mathcal{W} = \partial\mathcal{P} \cup \partial\mathcal{Q}$.*

Proof. We first prove $\partial\mathcal{W} = \partial\mathcal{P} \cup \partial\mathcal{Q}$.

1) For any edge (u, v) in $\partial\mathcal{W}$, u and v are disconnected in $E \setminus (\partial\mathcal{P} \cup \partial\mathcal{Q})$. The edge (u, v) must be deleted to disconnect u, v, so $(u, v) \in \partial\mathcal{P} \cup \partial\mathcal{Q}$. We conclude that $\partial\mathcal{W} \subseteq \partial\mathcal{P} \cup \partial\mathcal{Q}$.
2) For any edge $(u, v) \in \partial\mathcal{P}$, u and v are in different sides of $\mathcal{P}$. Then, any u-v path must contain an edge in $\partial\mathcal{P}$, and there cannot be a u-v path after deleting $\partial\mathcal{P}$. In other words, u and v are disconnected in $E \setminus (\partial\mathcal{P} \cup \partial\mathcal{Q})$, and $(u, v) \in \partial\mathcal{W}$. We conclude that $\partial\mathcal{P} \subseteq \partial\mathcal{W}$. $\partial\mathcal{Q} \subseteq \partial\mathcal{W}$ follows the same argument, so $\partial\mathcal{P} \cup \partial\mathcal{Q} \subseteq \partial\mathcal{W}$.

Next, we prove that $\mathcal{W}$ is a min ratio cut. Let π be the ratio of min ratio cuts $\mathcal{P}$ and $\mathcal{Q}$. It suffices to prove that the ratio of $\mathcal{W}$ is at most π.

We start by applying the inclusion-exclusion principle to get

$$c(\partial\mathcal{P} \cup \partial\mathcal{Q}) = c(\partial\mathcal{P}) + c(\partial\mathcal{Q}) - c(\partial\mathcal{P} \cap \partial\mathcal{Q}).$$

Notice that the first three terms are cut values. We have

$$d(\mathcal{W}) = d(\mathcal{P}) + d(\mathcal{Q}) - c(\partial\mathcal{P} \cap \partial\mathcal{Q}).$$

The last term may not be a cut value. We relate it to another partition as follows. Let $\mathcal{R}$ be the partition formed by all connected components in $E \setminus (\partial\mathcal{P} \cap \partial\mathcal{Q})$. By the same argument as point 1 above, $\partial\mathcal{R} \subseteq \partial\mathcal{P} \cap \partial\mathcal{Q}$. We have

$$c(\partial\mathcal{P} \cap \partial\mathcal{Q}) \geq c(\partial\mathcal{R}) \geq \pi(|\mathcal{R}| - 1).$$

When $\mathcal{R}$ is a multiway cut, the second inequality above is by minimality of π; when $\mathcal{R}$ only has one component, the second inequality degenerates to $0 \geq 0$.

Recall that $\pi = \frac{d(\mathcal{P})}{|\mathcal{P}|-1} = \frac{d(\mathcal{Q})}{|\mathcal{Q}|-1}$. So,

$$\begin{aligned} d(\mathcal{W}) &= d(\mathcal{P}) + d(\mathcal{Q}) - c(\partial\mathcal{P} \cap \partial\mathcal{Q}) \\ &\leq \pi(|\mathcal{P}| - 1) + \pi(|\mathcal{Q}| - 1) - \pi(|\mathcal{R}| - 1). \end{aligned}$$

We claim that $|\mathcal{P}| + |\mathcal{Q}| - |\mathcal{R}| \leq |\mathcal{W}|$. Then, we have $d(\mathcal{W}) \leq \pi(|\mathcal{W}| - 1)$, which means the ratio of $\mathcal{W}$ is at most π.

It remains to establish the claim. Consider any side R_i of $\mathcal{R}$. Let $\mathcal{P}_i \subseteq \mathcal{P}$ (resp. $\mathcal{Q}_i \subseteq \mathcal{Q}, \mathcal{W}_i \subseteq \mathcal{W}$) be the sides of $\mathcal{P}$ (resp. $\mathcal{Q}, \mathcal{W}$) that are contained in R_i. Because $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{W}$ are subdivisions of $\mathcal{R}$ (since $\partial\mathcal{R} \subseteq \partial\mathcal{P}, \partial\mathcal{R} \subseteq \partial\mathcal{Q}, \partial\mathcal{R} \subseteq \partial\mathcal{W}$), the subsets $\mathcal{P}_i$ (resp. $\mathcal{Q}_i, \mathcal{W}_i$) form a partition of $\mathcal{P}$ (resp. $\mathcal{Q}, \mathcal{W}$). In particular, $\sum_i |\mathcal{P}_i| = |\mathcal{P}|$, $\sum_i |\mathcal{Q}_i| = |\mathcal{Q}|$, and $\sum_i |\mathcal{W}_i| = |\mathcal{W}|$.

As a side of $\mathcal{R}$, R_i is connected after deleting $\partial\mathcal{P} \cap \partial\mathcal{Q}$. By definition of $\mathcal{W}$, each side of $\mathcal{W}$ is connected. Let T_i be a tree spanning R_i in $E \setminus (\partial\mathcal{P} \cap \partial\mathcal{Q})$ satisfying $|T_i \cap (\partial\mathcal{W})| = |\mathcal{W}_i| - 1$, which can be constructed by starting with a spanning tree in each $\mathcal{W}_i$ and then connecting the spanning trees to each other. Because T_i connects $|\mathcal{P}_i|$ components of $\mathcal{P}$, $|T_i \cap \partial\mathcal{P}| \geq |\mathcal{P}_i| - 1$. By the same argument, $|T_i \cap \partial\mathcal{Q}| \geq |\mathcal{Q}_i| - 1$. Because no edge of T_i is in both $\partial\mathcal{P}$ and $\partial\mathcal{Q}$, the sets $T_i \cap \partial\mathcal{P}$ and

$T_i \cap \partial\mathcal{Q}$ are disjoint. In conclusion,

$$\begin{aligned}|\mathcal{W}_i| - 1 = |T_i \cap \partial\mathcal{W}| &= |T_i \cap (\partial\mathcal{P} \cup \partial\mathcal{Q})| \\ &= |T_i \cap \partial\mathcal{P}| + |T_i \cap \partial\mathcal{Q}| \\ &\geq |\mathcal{P}_i| - 1 + |\mathcal{Q}_i| - 1.\end{aligned}$$

Suppose $\mathcal{R} = \{R_1, \ldots, R_k\}$. Summing over i,

$$\begin{aligned}|\mathcal{W}| - k &= \sum_{i=1}^{k} (|\mathcal{W}_i| - 1) \\ &\geq \sum_{i=1}^{k} (|\mathcal{P}_i| - 1 + |\mathcal{Q}_i| - 1) \\ &= |\mathcal{P}| - k + |\mathcal{Q}| - k,\end{aligned}$$

which implies the claim. □

Let $\mathcal{W}$ be the connected components after deleting the edge sets of all min-ratio cuts. By Lemma II.4, $\mathcal{W}$ is also a min-ratio cut. $\mathcal{W}$ maximizes the number of sides among all min-ratio cuts. We call $\mathcal{W}$ the *maximal min-ratio cut*.

We use *canonical cut hierarchy* to denote the hierarchy constructed by the recursive definition given above, where in each step we choose the maximal min-ratio cut. From now on, we mean the canonical cut hierarchy when we refer to the cut hierarchy of a graph.

III. Structure of Cut Hierarchy

Of key importance to our cut hierarchy algorithm are the notions of star sets and dense cores. We define these in this section, and establish structural connections between them. We then algorithmically exploit these properties in our cut hierarchy algorithm, which appears in the next section.

A. Star Sets and Dense Cores

We define a *star set* to be a vertex set that corresponds to an internal node of the canonical cut hierarchy whose children are all leaves. I.e., the maximal min-ratio cut of $G[S]$ is the all-singleton cut.

A single step of our cut hierarchy algorithm is to find a star set, which it can then contract and recurse. To find a star set, our algorithm finds a dense core instead, which is defined below. Recall that $\rho(S)$ is the skew-density of S, defined as $\rho(S) = \frac{c(E[S])}{|S|-1}$ when $|S| \geq 2$, and $\rho(S) = 0$ when $|S| \leq 1$. A dense core is a vertex set that is skew-denser than all its subsets and supersets. Formally,

Definition III.1. A vertex set $S \subseteq V$ is called a *dense-core* if $\forall W \subseteq S, \rho(W) \leq \rho(S)$, and $\forall U \supsetneq S, \rho(U) < \rho(S)$.

We say a vertex set is a skew-densest set if it has maximum skew-density among all vertex sets of the graph. We say a vertex set is a maximum skew-densest set if it maximizes size among all skew-densest sets. The following fact is straightforward to check from the definition.

Fact III.2. *A maximum skew-densest set is a dense core. Hence, every graph contains a dense core.*

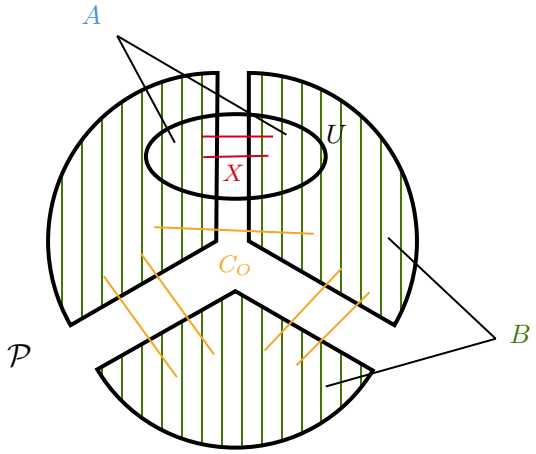

Fig. 1. Notation when U crosses $\mathcal{P}$ in the proof of Lemma III.4

The following fact shows that the induced subgraph of a dense core is connected.

Fact III.3. *If S is a dense core in a nonempty graph G, then $G[S]$ is connected.*

Proof. Let $\{S_i\}_{i=1}^{k}, k \geq 1$ be the connected components of S. Because S is a dense-core and $S_i \subseteq S$, we have $\rho(S_i) \leq \rho(S)$ for each S_i. Combined with (1), we have for all i,

$$c(E[S_i]) = \rho(S_i) \cdot (|S_i| - 1) \leq \rho(S) \cdot (|S_i| - 1)$$

Taking the sum gives

$$\begin{aligned}c(E[S]) &= \sum_{i=1}^{k} c(E[S_i]) \\ &\leq \rho(S) \cdot \left(\sum_{i=1}^{k} (|S_i| - 1) \right) \quad = \rho(S) \cdot (|S| - k).\end{aligned}$$

Combined with $c(E[S]) = \rho(S) \cdot (|S| - 1)$ from (1), we have

$$\rho(S)(|S| - 1) \leq \rho(S)(|S| - k) \implies \rho(S)(k - 1) \leq 0$$

We assume the graph G is nonempty, so $\rho(V) > 0$. Either $V = S$ or $V \supsetneq S$, in both cases we have $\rho(S) \geq \rho(V)$. Then $\rho(S) > 0$, and we conclude $k - 1 \leq 0$. Because k is a positive integer, we have $k = 1$, which means S is connected. □

B. Dense cores are Star Sets

Next, we show that dense cores and star sets are closely related. We first show the relationship between a dense core and the min-ratio cut.

Lemma III.4. *Let $\mathcal{P}$ be a min-ratio cut in a connected graph G. Suppose U is a dense core in G and $U \neq V$. Then, U is contained in one side of $\mathcal{P}$.*

Proof. Assume for contradiction that $G[U]$ is separated into more than one side of $\mathcal{P}$. Because $G[U]$ is connected by Fact III.3, this means $E[U] \cap \partial\mathcal{P} \neq \varnothing$.

Let $r(\cdot)$ be the rank function of the graphic matroid. Recall that r is submodular. So, for edge sets $A \subseteq B$ and X disjoint from B, we have

$$r(B \cup X) - r(B) \leq r(A \cup X) - r(A).$$

Substituting $A = E[U] \setminus \partial\mathcal{P}, B = E \setminus \partial\mathcal{P}, X = E[U] \cap \partial\mathcal{P}$ and denoting $C_O = \partial\mathcal{P} \setminus X$ (see Figure 1), we have

$$r(E \setminus C_O) - r(E \setminus \partial\mathcal{P}) \leq r(E[U]) - r(E[U] \setminus \partial\mathcal{P}). \quad (2)$$

In the min-ratio cut $\mathcal{P}$, every side is connected by Fact II.3. Applying Fact II.2 on $U = V$ gives $|\mathcal{P}| - 1 = r(E) - r(E \setminus \partial\mathcal{P})$. Denote π to be the ratio of $\mathcal{P}$, i.e.,

$$d(\mathcal{P}) = \pi \cdot (r(E) - r(E \setminus \partial\mathcal{P})).$$

We claim

$$c(C_O) \geq \pi \cdot (r(E) - r(E \setminus C_O)).$$

When C_O does not separate anything in G, RHS is trivially 0. Otherwise, a subset of C_O is the edge set of a multiway cut on G, whose ratio is at least π, establishing the claim.

Taking the difference between the above two inequalities and applying (2) gives

$$\begin{aligned} c(X) &= d(\mathcal{P}) - c(C_O) \\ &\leq \pi \cdot (r(E \setminus C_O) - r(E \setminus \partial\mathcal{P})) \\ &\leq \pi \cdot (r(E[U]) - r(E[U] \setminus \partial\mathcal{P})). \end{aligned}$$

That is, we get

$$c(X) \leq \pi \cdot (r(E[U]) - r(E[U] \setminus \partial\mathcal{P})). \quad (3)$$

We assumed for contradiction that U is separated into more than one side of $\mathcal{P}$. Let $\{U_i\}_{i=1}^k, k \geq 2$ be the connected components in $G[U]$ after deleting all cut edges $\partial\mathcal{P}$. Because U is a dense core and U_i are subsets of U, $\rho(U_i) \leq \rho(U)$ for each U_i. Combined with (1), we have for all i,

$$c(E[U_i]) = \rho(U_i) \cdot (|U_i| - 1) \leq \rho(U) \cdot (|U_i| - 1)$$

Summing over all i,

$$\begin{aligned} c(E[U] \setminus \partial\mathcal{P}) &= \sum_{i=1}^{k} c(E[U_i]) \\ &\leq \rho(U) \cdot (|U| - k) \\ &= \rho(U) \cdot r(E[U] \setminus \partial\mathcal{P}), \end{aligned}$$

where the last equality is by Fact II.2. So we have

$$c(E[U] \setminus \partial\mathcal{P}) \leq \rho(U) \cdot r(E[U] \setminus \partial\mathcal{P}). \quad (4)$$

Combining (3) and (4), we have

$$\begin{aligned} c(E[U]) &= c(E[U] \setminus \partial\mathcal{P}) + c(X) \\ &\leq \rho(U) \cdot r(E[U] \setminus \partial\mathcal{P}) \\ &\quad + \pi \cdot (r(E[U]) - r(E[U] \setminus \partial\mathcal{P})) \end{aligned}$$

Notice that $\pi \leq \rho(V)$ because $\rho(V)$ is the ratio of the all-singleton cut. Because U is a dense core and $V \supsetneq U$, we have $\rho(U) > \rho(V) \geq \pi$. By the assumption that $\mathcal{P}$ partitions $G[U]$ into multiple connected components, $r(E[U]) > r(E[U] \setminus \partial\mathcal{P})$. Therefore,

$$\begin{aligned} \rho(U) \cdot (|U| - 1) &= c(E[U]) \\ &< \rho(U) \cdot r(E[U] \setminus \partial\mathcal{P}) \\ &\quad + \rho(U) \cdot (r(E[U]) - r(E[U] \setminus \partial\mathcal{P})) \\ &= \rho(U) \cdot r(E[U]) \end{aligned}$$

Finally, noticing that $r(E[U]) = |U| - 1$ because $G[U]$ is connected, we have $p(U) \cdot (|U| - 1) < p(U) \cdot (|U| - 1)$, which is a contradiction. □

Next, we can show that, in fact, dense cores are star sets.

Lemma III.5. *If S is a dense core, then S is a star set of the canonical cut hierarchy.*

Proof. We prove by induction on $n = |V|$.

The base case is $n = |S|$ or equivalently $V = S$. It suffices to show that the all-singleton cut is a min-ratio cut of G. If this is true, then the recursive min-ratio cut procedure will choose the all-singleton cut which is the maximal min-ratio cut of G. Then S is partitioned into singletons, which means S is a star set in the hierarchy.

Because S is a dense core, for all nonempty $W \subseteq S$,

$$c(E[W]) = \rho(W) \cdot (|W| - 1) \leq \rho(S) \cdot (|W| - 1)$$

Consider any multiway cut $\mathcal{P} = \{W_i\}_{i=1}^k$ of $G[S]$. We have

$$\begin{aligned} d(\mathcal{P}) &= c(E[S]) - \sum_{i=1}^{k} c(E[W_i]) \\ &\geq \rho(S) \cdot (|S| - 1) - \rho(S) \cdot \left(\sum_{i=1}^{k} (|W_i| - 1) \right) \\ &= \rho(S) \cdot (k - 1). \end{aligned}$$

So the ratio of $\mathcal{P}$ is at least $\rho(S)$, which is the ratio of the all-singleton cut. In conclusion the all-singleton cut is a min-ratio cut.

Next we consider the inductive case where $S \subsetneq V$. Let $\mathcal{P}$ be the maximal min-ratio cut of G. By Lemma III.4, S is contained in one side $U \in \mathcal{P}$. Note that since S is a dense core in G, S is also a dense core in $G[U]$ for $U \subseteq V$ because the conditions on graph $G[U]$ is a subset of the conditions on G. Using the inductive hypothesis, we get that S is a star set in the canonical hierarchy of $G[U]$, which is a subtree of the canonical hierarchy of $G[V]$. In conclusion S is a star set in the canonical hierarchy of G. □

IV. Cut Hierarchy Algorithm

In this section, we design an algorithm (Algorithm 1) that constructs the canonical cut hierarchy w.h.p. in $\widetilde{O}(nm \log C)$ time plus $\widetilde{O}(n \log C)$ max-flow calls, thereby establishing Theorem I.1.

Theorem I.1. *There is a randomized algorithm that, given an undirected graph G on m edges and n vertices with integer*

edge weights in $\{1, 2, \ldots, C\}$*, outputs the cut hierarchy (and hence ideal edge loads) of* G *in* $\widetilde{O}(nm \log C)$ *time plus* $\widetilde{O}(n \log C)$ *calls to a maximum flow subroutine on directed graphs with* $O(m)$ *edges and* $O(m)$ *vertices and integer edge weights at most* $\mathrm{poly}(n, C)$.

In each iteration of the algorithm, we identify and contract a star set in the canonical cut hierarchy of the current graph (see Figure 2). Recall that a star set represents a node in the hierarchy whose children are all leaves of the hierarchy. In other words, it is a minimal non-singleton set in the laminar family corresponding to the hierarchy. We will show that contracting a star set does not affect the rest of the laminar family. Then, we can obtain the laminar family by collecting all sets found by the algorithm.

A. Contracting Star Sets

We first observe that contracting a subset of a side does not affect the maximal min-ratio cut.

Fact IV.1. *Suppose* $\mathcal{P}$ *is the maximal min-ratio cut of* G *with a side* $U \in \mathcal{P}$*, and* $S \subseteq U$*. Then,* $\mathcal{P}/S$ *is still the maximal min-ratio cut of* G/S *after contracting* S.

Proof. Let $\mathcal{C}_1$ be the family of multiway cuts in G that do not separate S, and $\mathcal{C}_2$ be the family of multiway cuts in G/S. The map $\phi : \mathcal{C}_1 \to \mathcal{C}_2$ defined by $\phi(\mathcal{Q}) = \mathcal{Q}/S$ is a bijection. Note that ϕ preserves the cut ratio because ϕ preserves the set of cut edges and the number of sides.

Because $\mathcal{P} \in \mathcal{C}_1$ and $\mathcal{P}$ is the maximal min-ratio cut of G, $\mathcal{P}$ maximizes the number of sides among the multiway cuts that minimize cut ratio in $\mathcal{C}_1$. After applying ϕ, this property is preserved by $\mathcal{P}/S$ in $\mathcal{C}_2$, which means $\mathcal{P}/S$ is the maximal min-ratio cut of G/S. □

A simple corollary of this fact is that contracting a star set does not affect the rest of the laminar family. This validates our bottom-up contraction algorithm.

Corollary IV.2. *If we contract a star set* S*, then in the canonical cut hierarchy, the only change is that the node corresponding to* S *is replaced by a leaf corresponding to the contracted vertex.*

Proof. We compare the recursive min-ratio cut process before and after the contraction. On a set U disjoint from S, it will also choose the same maximal min-ratio cut. On a set $U \supsetneq S$, it will choose the same maximal min-ratio cut up to contracting S by Fact IV.1. The only difference is that originally the process recurses on S, while after contraction, S is a singleton and hence a leaf in the hierarchy. □

B. Algorithm Framework

The algorithm runs in $O(n)$ iterations. Each iteration finds a star set and contracts it. The output is the laminar family of all sets found during the algorithm.

We give a high-level description before formally stating the algorithm. Recall that by Fact III.2, the maximum skew-densest set is a dense core, and hence a star set by Lemma III.5. We can design an algorithm for finding the maximum skew-densest subgraph (call it D) based on previously known approaches to finding the densest subgraph. The bottleneck is that this algorithm uses a directed (global) min-cut subroutine, for which the best known algorithm has a running time of $m^{1+o(1)}\sqrt{n}$ [19]. This is sufficient if $|D| \geq \Omega(\sqrt{n})$ since we can amortize the running time over the sets that we contract. But, for small D, we need a sharper running time bound. For this purpose, we show that a modification of the global directed min-cut algorithm can be used to identify D in $m^{1+o(1)}k$ time, for a given upper bound $k \geq |D|$. So, if we are given an upper bound k that (say) satisfies $k \leq 2|D|$, we would be done. But, how we find such an upper bound? We can use guess-and-double, but for this to work, we need to identify the complementary case: $k < |D|$. In other words, the above algorithm returns some set as the presumptive D, and we need to verify if that is indeed the case. Unfortunately, we do not know how to do this efficiently. Instead, we settle for the simpler goal: identify if the set returned by the above algorithm is a dense core. For this latter task, we give an algorithm that runs in $m^{1+o(1)}k$ time. Note that this is sufficient since any dense core (whether or not the maximum skew-densest set) is a star set in the cut hierarchy.

Now, we formally describe the algorithm for each iteration (see Algorithm 1). The algorithm guesses a size parameter k by doubling (starting from 2, i.e. $k = 2, 4, 8, \ldots$, because star sets are not singleton). We call a subroutine FIND-STAR described in Section IV-D to find a vertex set S_k. If $|S_k| \leq k/2$ or $|S_k| > k$, we reject it and go to the next value of k. Otherwise, we run a subroutine VERIFY-CORE described in Section IV-E to decide whether S_k is a dense core. If S_k is verified as a dense core, we finish the iteration and return S_k. Otherwise, we go to the next value of k.

Algorithm 1: CONSTRUCT-CANONICAL-MIN-RATIO-CUT-HIERARCHY(G)

```
1.1 F ← ∅
1.2 while G is not contracted into a singleton do
1.3     for k = 2, 4, 8, ..., 2^⌈log n⌉ do
1.4         S_k ← FIND-STAR(G, k)
1.5         if k/2 < |S_k| ≤ k and
              VERIFY-CORE(G, k, S_k) then
1.6             Add S_k to F
1.7             Contract S_k in G
1.8             Break for loop
1.9 Output the laminar family F.
```

In the rest of the section, we describe the two subroutines of finding a candidate set (Algorithm 2 in Section IV-D) and verifying a dense core (Algorithm 3 in Section IV-E). The results are summarized in Lemma IV.3 and Lemma IV.4. Informally, we guarantee that FIND-STAR can find the maximum skew-densest set when k is large enough, and VERIFY-CORE can correctly verify whether the input set is a dense core.

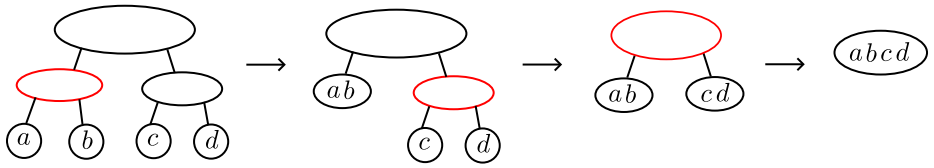

Fig. 2. Illustration of iterative contraction algorithm. Red nodes represent the contracted star set in the hierarchy.

Lemma IV.3 (Find-Star)**.** *Given a connected input graph G with integer edge capacities at most C and a parameter k, Algorithm 2 outputs the maximum skew-densest set D w.h.p. assuming $k \geq |D|$. The algorithm runs in $\widetilde{O}(mk \log C)$ time plus $\widetilde{O}(\log C)$ calls to max-flow on digraphs with $O(m)$ nodes and $O(m)$ edges.*

Lemma IV.4 (Verify-Core)**.** *For an input graph G, a set of vertices S, and a parameter k, Algorithm 3 returns true if and only if S is a dense core in G and $|S| \leq k$. The running time is $O(k)$ calls to max-flow on digraphs with $O(m)$ nodes and $O(m)$ edges.*

In the running time analysis, we use $F(n, m)$ to denote the time complexity of max-flow on digraph of n nodes and m edges. We assume $F(n, m)$ is monotone increasing and $F(n, m) \geq \Omega(m)$. The following is our main lemma, which establishes Theorem I.1, and follows from Lemma IV.3 and Lemma IV.4.

Lemma IV.5. *Algorithm 1 outputs the canonical cut hierarchy of G w.h.p. in $\widetilde{O}(nm \log C)$ time plus $\widetilde{O}(n \log C)$ calls to max flow on digraphs with $O(m)$ edges and $O(m)$ vertices and integer edge weights at most* $\mathrm{poly}(n, C)$.

Proof. We first show that each iteration can find and contract a set S_k w.h.p. Let D be the maximum skew-densest set of the current contracted graph, which is a dense core by Fact III.2. Let k_D be the smallest power of 2 greater or equal to $|D|$. If the inner for loop breaks before k_D, the claim holds. Otherwise, consider the inner for loop when $k = k_D$. By Lemma IV.3, FIND-STAR is guaranteed to find D w.h.p. when $k \geq |D|$. By Lemma IV.4, D will be verified as a dense core. Also, $k_D/2 < |D| \leq k_D$. So, the inner for loop will break when $k = k_D$. In conclusion, the inner for loop will break before or at k_D, which means some S_k is contracted.

Suppose the above high probability event happens. Then, each iteration can only terminate when VERIFY-CORE returns true for some set S_k. By Lemma IV.4, such an S_k must be a dense core. Then, S_k is a star set by Lemma III.5. Thus, each iteration contracts a star set, and this does not affect the rest of the hierarchy by Corollary IV.2. After $O(n)$ iterations, the whole graph is contracted into a singleton, and all sets visited by the algorithm form the canonical cut hierarchy.

Each iteration runs $O(\log n)$ steps of doubling k. For each k, the algorithm calls FIND-STAR and VERIFY-CORE, which take $\widetilde{O}(mk \log C + F(m, m)(k + \log C)) = \widetilde{O}((m + F(m, m))k \log C)$ time by Lemmas IV.3 and IV.4. The iteration contracts a set of size $> k/2$, hence decreases the vertex size of the contracted graph by at least $k/2$. So, the amortized time to contract a vertex is $\widetilde{O}((m + F(m, m)) \log C)$, and the total running time is $\widetilde{O}(nm \log C + n \log C \cdot F(m, m))$. (The infinite edge weights can be simulated by a large weight of $O(mC)$.) □

C. Goldberg's Network and Gabow's Variant

We need Goldberg's network and its variant due to Gabow as key building blocks of both the FIND-STAR and VERIFY-CORE subroutines.

Given an undirected graph G with edge capacity c and a parameter $\tau > 0$, we construct a directed graph H that we call *Goldberg's network* as follows [20] (see the first two diagrams of Figure 3). To avoid ambiguity, we use *nodes* and *arcs* in place of vertices and edges for Goldberg's network H.

- Start with a bipartite graph on node set $E \uplus V$ defined as follows: For each $e \in E$ and each endpoint v of e in G, add an arc in H from node e to node v of capacity ∞.
- Add a node s with an arc (s, e) of capacity c_e for each $e \in E$.
- Add a node t with an arc (v, t) of capacity τ for each $v \in V$.

Fact IV.6. $|V[H]| = O(m + n)$, $|E[H]| = O(m + n)$.

In this section, we require the concepts of (directed) s-t flows, s-t cuts, and t-cuts of directed graphs. An s-t flow assigns a value $f(u, v)$ to each arc (u, v) satisfying $0 \leq f(u, v) \leq c(u, v)$. For simplicity, assume that $f(u, v) = 0$ if (u, v) is not an arc of the graph. An s-t flow must satisfy the flow conservation constraints, namely $\sum_v f(u, v) = \sum_v f(v, u)$ for all vertices $u \notin \{s, t\}$. The value of the s-t flow equals $\sum_v f(s, v)$, and an s-t max flow is an s-t flow of maximum value. Given an s-t flow f, the residual graph of f has the following capacities on arcs: for each pair of arcs (u, v) and (v, u), where a nonexistent arc is considered an arc of capacity 0, the residual graph sets capacities $c(u, v) - f(u, v) + f(v, u)$ and $c(v, u) - f(v, u) + f(u, v)$ on the arcs (u, v) and (v, u), respectively.

We use $\partial^+ S$ to denote the set of arcs from S to $V \setminus S$. An s-t cut is the arc set $\partial^+ S$ for some vertex set S that contains s but not t. We call S the source side of the s-t cut. The value of an s-t cut $\partial^+ S$ is the total capacity $d^+(S) = \sum_{e \in \partial^+ S} c_e$. The s-t mincut is the s-t cut of minimum value, which is also the value of the s-t max flow. A t-cut is the arc set $\partial^+ S$ for some set of vertices S not containing t. We also call S the source side of the t-cut, and value and t-mincut are defined

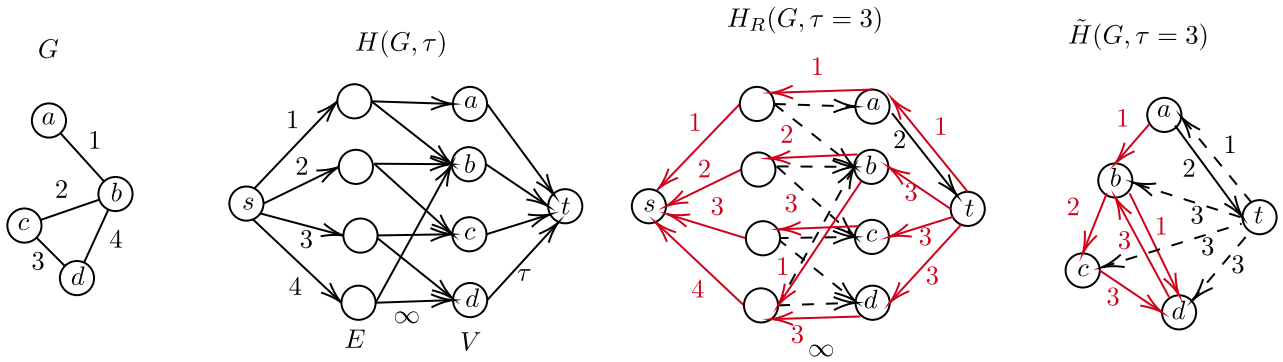

Fig. 3. Example of H and $\tilde{H}$ on input graph G and $\tau = 3$

analogously. We sometimes abuse notation and refer to the s-t cut or t-cut by the source side S instead of the set of arcs $\partial^+ S$.

We require the following fact about cuts in residual graphs, which follows from flow conservation and the construction of the residual graph.

Fact IV.7. *Consider a directed graph H, an s-t flow of value τ, and the residual graph H_R of f. For any vertex set S that contains s but not t, we have $d^+_{H_R}(S) = d^+_H(S) - \tau$.*

We can characterize the value of any (directed) s-t cut on Goldberg's network H as follows.

Fact IV.8. *For any s-t cut on H with source side $S \ni s$, let $S_E = S \cap E$ and $S_V = S \cap V$ be the nodes in the source side from E and from V respectively, i.e., $S = \{s\} \cup S_E \cup S_V$. Then, the cut value of S is*

$$d^+_H(S) = \begin{cases} c(E) - c(S_E) + \tau \cdot |S_V| & S_E \subseteq E[S_V] \\ \infty & S_E \not\subseteq E[S_V] \end{cases}$$

Proof. We first prove that the cut value is finite if and only if $S_E \subseteq E[S_V]$, i.e., S_E is a subset of edges in the induced subgraph of G on S_V.

Assume the cut value is finite. For every $e = (u, v) \in S_E$, it has two outgoing infinite arcs (e, u) and (e, v). Infinite arcs cannot appear in $\partial^+ S$, so $u, v \in S_V$. It follows that $(u, v) \in E[S_V]$. Since this holds for every $e \in S_E$, we have $S_E \subseteq E[S_V]$.

Assume the cut value is infinite, i.e., there exists an infinite arc (e, u) in $\partial^+ S$. Then, $e = (u, v)$ for some vertex v, and $e \in S_E, u \notin S_V$. This implies $e \notin E[S_V]$ and $S_E \not\subseteq E[S_V]$.

Next, we compute the cut value when it is finite. There are two types of cut arcs: from s to $E \setminus S_E$ and from S_V to t. The first type contributes $c(E \setminus S_E) = c(E) - c(S_E)$. The second type contributes $\tau \cdot |S_V|$. Taking their sum gives the statement. □

As a corollary, we can characterize the s-t min cut on H.

Lemma IV.9. *The s-t min cut on H has source side $S = \{s\} \cup S_V \cup E[S_V]$, where $S_V = \arg\max_{X \subseteq V}(c(E[X]) - \tau|X|)$.*

Proof. For any s-t cut on H with source side $S \ni s$, denote $S_E = S \cap E$ and $S_V = S \cap V$. From Fact IV.8, we have the following: For any fixed $S_V \subseteq V$, starting from $S_E = \varnothing$, the cut value $d^+_H(S)$ is decreased if we add an edge in $E[S_V]$ to S_E, and the cut value becomes infinity if we add an edge outside $E[S_V]$ to S_E. So, for fixed $S_V \subseteq V$, $d^+_H(S)$ is minimized when $S_E = E[S_V]$. So, the min cut value is $\min_{X \subseteq V} d^+_H(\{s\} \cup X \cup E[X])$. Because $d^+_H(\{s\} \cup X \cup E[X]) = c(E) - c(E[X]) + \tau|X|$ by Fact IV.8 and $c(E)$ is a constant, the cut value is minimized when $c(E[X]) - \tau|X|$ is maximized. □

Sometimes, we need to force some node $u \in V$ to be in the source side of the s-t min cut. This can be achieved by adding an infinite capacity arc from s to u.

Lemma IV.10. *Construct H' by adding to H an infinite capacity arc from s to $u \in V$. The s-t min cut on H' has source side $S = \{s\} \cup S_V \cup E[S_V]$, where $S_V = \arg\max_{u \in X \subseteq V}(c(E[X]) - \tau|X|)$.*

Proof. For any s-t cut on H with source side $S \ni s$, denote $S_E = S \cap E$ and $S_V = S \cap V$. When $u \in S_V$, $d^+_{H'}(S) = d^+_H(S)$ because the new arc (s, u) does not contribute to $\partial^+ S$. When $u \notin S_V$, $d^+_{H'}(S) = \infty$ due to the new arc.

We have shown in Lemma IV.9 that $d^+_H(S)$ is minimized when $S_E = E[S_V]$ for fixed S_V. So, $d^+_{H'}(S)$ is also minimized when $S_E = E[S_V]$ for fixed S_V, and $d^+_{H'}(\{s\} \cup X \cup E[X]) = c(E) - c(E[X]) + \tau|X|$ when $u \in X$. Since $c(E)$ is a constant, $d^+_{H'}(S)$ is minimized when $S_V = \arg\max_{u \in X \subseteq V}(c(E[X]) - \tau|X|)$. □

Next, we construct a modified network $\tilde{H}$ based on Goldberg's network H (see the latter two diagrams of Figure 3). The construction is due to [5].

We start from the residual network H_R of the s-t max flow on H. We first remove s and all its incident arcs from H_R. Then, we shortcut all edge nodes as follows. Consider any edge node $e = (u, v) \in E$. In H_R, e is incident to 4 arcs, among which (e, u) and (e, v) are infinite capacity arcs from H, and (u, e) and (v, e) are created by the residual network, possibly of capacity 0. We perform the following:

1) Replace arcs $(u,e),(e,v)$ by (u,v), and set capacity $c_{\tilde{H}}(u,v) = c_{H_R}(u,e)$.
2) Replace arcs $(v,e),(e,u)$ by (v,u), and set capacity $c_{\tilde{H}}(v,u) = c_{H_R}(v,e)$.
3) Remove the node e.

After shortcutting all edge nodes, we get a modified network $\tilde{H}$.

Fact IV.11. $|V[\tilde{H}]| = O(n)$, $|E[\tilde{H}]| = O(m+n)$.

Lemma IV.12. *Suppose the s-t max flow value on H is $c(E) = \sum_{e\in E} c(e)$. Then for any cut $X \subseteq V$ in $\tilde{H}$,*

$$d^+_{\tilde{H}}(X) = \tau|X| - c(E[X]).$$

Proof. Consider any $X \subseteq V$. Let $S = \{s\} \cup X \cup E[X]$. By Fact IV.8, $d^+_H(S) = c(E) - c(E[X]) + \tau|X|$. S is an s-t cut, so its cut value in the residual network is, by Fact IV.7,

$$d^+_{H_R}(S) = d^+_H(S) - c(E) = \tau|X| - c(E[X]).$$

Note that $d^+_H(s) = c(E)$ by construction of H. So, the assumption implies that all (s,e) edges are saturated in the flow. Then, (s,e) edges have no capacity in H_R, and removing s from S does not affect $d^+_{H_R}(S)$.

Next, we consider the effect of shortcutting $e = (u,v) \in E$. If $S \supseteq \{u,v\}$ or $S \cap \{u,v\} = \varnothing$, then $\partial^+_{H_R}(S)$ is not affected. Otherwise, assume $S \cap \{u,v\} = \{u\}$ (the case of $S \cap \{u,v\} = \{v\}$ is symmetric). Then e is not in $E[X]$. Among the 4 arcs incident to e, only (u,e) is in $\partial^+_{H_R}(S)$. After shortcutting, only (u,v) is in $\partial^+_{\tilde{H}}(S)$. So, the cut value does not change. In conclusion $d^+_{\tilde{H}}(X) = d^+_{H_R}(S) = \tau|X| - c(E[X])$. □

D. The FIND-STAR Subroutine

In this section, we design an algorithm that outputs some vertex set given G and k in $\widetilde{O}(mk \log C + F(m,m)\log C)$ time. Moreover, if k is larger than the size of maximum skew-densest set, then the algorithm must output the maximum skew-densest set.

The algorithm needs the following subroutines. They are adapted from the global directed min cut algorithm [19]. We include a proof of Lemma IV.14 in Appendix A for completeness.

Lemma IV.13 (Lemmas 2.6 and 2.7 of [19])**.** *There exists an algorithm that, given a capacitated digraph G with a fixed root vertex t, finds a t-mincut w.h.p. when the source side of a t-mincut has size $\leq k$. The algorithm runs in $\widetilde{O}(mk)$ time plus polylogarithmic calls to max flow on digraphs with $O(m)$ edges and $O(n)$ vertices and integer edge weights at most* $\mathrm{poly}(n, C)$.

Lemma IV.14 (Minimum Directed Cut)**.** *There exists an algorithm that, given a capacitated digraph G with a fixed root vertex t and a number $\tau > 0$, assuming there exists a t-cut S with $d^+(S) < \tau$ and $|S| \leq k$, finds a t-cut S with $d^+(S) < \tau$ w.h.p. The algorithm runs in $\widetilde{O}(mk)$ time plus polylogarithmic calls to max flow on digraphs with $O(m)$ edges and $O(n)$ vertices and integer edge weights at most* $\mathrm{poly}(n, C)$.

The algorithm performs a binary search for τ up to a precision of n^{-3}. Let τ^* be the maximum τ (in binary search) that the following procedure succeeds. The algorithm outputs the t-mincut of $\tilde{H}(G, \tau^*)$ using Lemma IV.13.

1) Construct Goldberg's network H from G with parameter τ.
2) Compute the s-t max flow on H. If the flow value $< c(E)$, then return success.
3) Construct the modified network $\tilde{H}$.
4) Call Lemma IV.14 on input $(\tilde{H}, t, \tau, k)$ to find a set U.
5) If $d^+_{\tilde{H}}(U) < \tau$, return success. Otherwise, return failure.

Algorithm 2: FIND-STAR(G, k)

```
2.1  τ_L ← 0, τ_R ← c(E)
2.2  while τ_R − τ_L ≥ n^{−3} do
2.3      τ ← (τ_L + τ_R)/2
2.4      Construct Goldberg's network H from G with
           parameter τ
2.5      if the s-t max flow value on H < c(E) then
2.6          τ_L = τ
2.7      else
2.8          Construct the modified network H̃ from H
2.9          Call Lemma IV.14 on input (H̃, t, τ, k) to find
               a set U
2.10         if d^+_H̃(U) < τ then
2.11             τ_L = τ
2.12         else
2.13             τ_R = τ
2.14 Output the t-mincut in H̃ of parameter τ_L using
       Lemma IV.13
```

Lemma IV.15. *Let D be the maximum skew-densest subgraph. Suppose $k \geq |D|$. The following holds w.h.p.: The binary search procedure on input τ succeeds if and only if $\tau < \rho(D)$.*

Proof. (Necessity) Assume the algorithm succeeds. There are two cases: success at step 2 or step 5. In both cases, we show $\tau < \rho(U)$ for some U, which implies $\tau < \rho(D)$ because D is the skew-densest set.

Suppose the algorithm succeeds at step 2. Let S be the source side of the s-t min cut in H computed at step 2. By Lemma IV.9 and Fact IV.8, $S = \{s\} \cup U_1 \cup E[U_1]$ for some $U_1 \subseteq V$, and $d^+_H(S) = c(E) - c(E[U_1]) + \tau|U_1|$. According to step 2, we have $d^+_H(S) < c(E)$ or equivalently $c(E[U_1]) - \tau|U_1| > 0$. Then we can apply (1) to get

$$c(E[U_1]) = \rho(U_1)(|U_1| - 1) > \tau|U_1| > \tau(|U_1| - 1)$$

which implies $\tau < \rho(U_1)$.

Suppose the algorithm succeeds at step 5. Since it did not succeed at step 2, the s-t max flow on H is $c(E)$. Let U_2 be the set found at step 4. By Lemma IV.12, $d^+_{\tilde{H}}(U_2) = \tau|U_2| - c(E[U_2])$. According to step 5, we have $d^+_{\tilde{H}}(U_2) < \tau$. Then

$|U_2| \geq 2$ and we can apply (1) to get

$$\tau(|U_2| - 1) < c(E[U_2]) = \rho(U_2)(|U_2| - 1)$$

which implies $\tau < \rho(U_2)$.

(Sufficiency) Assume $\tau < \rho(D)$. We need to show that the algorithm succeeds w.h.p.

If the algorithm succeeds at step 2, we are done. Next assume it did not, which means the s-t max flow on H is $c(E)$. By Lemma IV.12,

$$\begin{aligned} d^+_{\tilde{H}}(D) &= \tau|D| - c(E[D]) \\ &= \tau|D| - \rho(D)(|D| - 1) \\ &= \tau + (\tau - \rho(D))(|D| - 1) < \tau. \end{aligned}$$

So there exists a t-cut D in $\tilde{H}$ with value $< \tau$ and $|D| \leq k$. By Lemma IV.14, step 4 finds a cut of value $< \tau$ w.h.p., and the algorithm will succeed at step 5. □

Lemma IV.16. *Suppose the input graph G is nonempty and has integer edge capacities. Let D be the maximum skew-densest subgraph of G. Suppose $\rho(D) - n^{-3} < \tau < \rho(D)$. Let H and $\tilde{H}$ be Goldberg's network and the modified network of G with parameter τ, respectively. Then, D is the t-mincut of $\tilde{H}$.*

Proof. Let $S = \{s\} \cup U_1 \cup E[U_1]$ be the s-t min cut in H. We claim that $U_1 = \varnothing$. Assume for contradiction that U_1 is nonempty. Then by (1),

$$c(E[U_1]) = \rho(U_1)(|U_1| - 1) \leq \rho(D)(|U_1| - 1).$$

Combined with Fact IV.8,

$$\begin{aligned} d^+_H(S) &= c(E) - c(E[U_1]) + \tau|U_1| \\ &\geq c(E) - \rho(D)(|U_1| - 1) + (\rho(D) - n^{-3})|U_1| \\ &\geq c(E) + \rho(D) - n^{-2}. \end{aligned}$$

Since edge capacities are integral, the total edge capacity in G is ≥ 1, so $\rho(D) \geq \rho(V) \geq \frac{1}{n} > n^{-2}$. But $\partial^+_H(\{s\})$ has smaller cut value $c(E)$, which contradicts the assumption that S is a s-t min cut. We conclude that the s-t mincut in H is $\{s\}$, which has value $c(E)$.

Since the s-t max flow in H also has value $c(E)$, we can apply Lemma IV.12 to obtain, for any nonempty $U \subseteq V$,

$$\begin{aligned} d^+_{\tilde{H}}(U) &= \tau|U| - c(E[U]) \\ &= \tau|U| - \rho(U)(|U| - 1) \\ &= \tau + (\tau - \rho(U))(|U| - 1). \end{aligned}$$

That is, we have

$$d^+_{\tilde{H}}(U) = \tau + (\tau - \rho(U))(|U| - 1). \tag{5}$$

In particular, $d^+_{\tilde{H}}(D) < \tau$ because $\rho(D) > \tau$ and $|D| \geq 2$.

Recall that we need to show that D is the t-mincut of $\tilde{H}$. For any set $U \subseteq V$ with $\rho(U) < \tau$ or $|U| = 1$, we have $d^+_{\tilde{H}}(U) \geq \tau$ by (5), so U is not the t-mincut. Next, we consider the remaining sets $U \subseteq V$ which satisfy $\rho(U) \geq \tau$ and $|U| \geq 2$. Because skew-density is in a set of fractional numbers with denominators in $[1, n-1]$, the gap between two different values of skew-density is at least n^{-2}. So, there cannot be another skew-density between $\rho(D) - n^{-3}$ and $\rho(D)$. It follows that $\rho(U) \geq \rho(D)$, and U is a skew-densest set. Among the skew-densest sets U with $\rho(U) = \rho(D) > \tau$, $d^+_{\tilde{H}}(U)$ decreases as $|U|$ increases by (5). So, $d^+_{\tilde{H}}(U)$ is minimized by the maximum skew-densest set, which is exactly D. □

We are now ready to prove Lemma IV.3.

Lemma IV.3 (Find-Star)**.** *Given a connected input graph G with integer edge capacities at most C and a parameter k, Algorithm 2 outputs the maximum skew-densest set D w.h.p. assuming $k \geq |D|$. The algorithm runs in $\widetilde{O}(mk \log C)$ time plus $\widetilde{O}(\log C)$ calls to max-flow on digraphs with $O(m)$ nodes and $O(m)$ edges.*

Proof. By Lemma IV.15, w.h.p. the binary search succeeds if and only if $\tau < \rho(D)$. The algorithm finds the maximum successful τ^* up to a precision of n^{-3}, so $\rho(D) - n^{-3} < \tau^* < \rho(D)$. By Lemma IV.16, for $\tilde{H}$ constructed from (G, τ^*), the t-mincut is D. The algorithm finds such a t-mincut w.h.p. by calling Lemma IV.13.

The algorithm runs $O(\log(nC))$ iterations of binary search because the total edge capacity is at most mC and the precision is n^{-3}. In each iteration, steps 1 and 3 take near-linear time to construct H and $\tilde{H}$. Step 2 runs a max flow on H with $O(m)$ vertices and edges, which takes $F(m, m)$ time. Step 4 takes $\widetilde{O}(mk + F(n, m))$ time according to Lemma IV.14. Finally, after the binary search terminates, the algorithm calls Lemma IV.13 which takes $\widetilde{O}(mk + F(n, m))$ time. In conclusion the running time of an iteration is $\widetilde{O}(mk + F(m, m))$. The total running time is $\widetilde{O}(mk \log C + F(m, m) \log C)$. □

E. The VERIFY-CORE *Subroutine*

In this section, given $S \subseteq V$ and k such that $|S| \leq k$, we give an algorithm that can verify whether S is a dense core in $O(k \cdot F(m, m))$ time.

The algorithm verifies whether S is denser than its subsets using a network $\tilde{H}_1$ constructed from the induced subgraph $G[S]$, and verifies whether S is denser than its supersets using a network H'_2 constructed from the contracted graph G/S.

Algorithm 3: VERIFY-CORE(G, k, S)

3.1 **if** $|S| > k$ **then**
3.2 ⌊ **return** False
3.3 Construct H_1 to be Goldberg's network on $G[S]$ with parameter $\tau = \rho(S)$.
3.4 Construct $\tilde{H}_1$ to be the modified network of H_1.
3.5 Construct H_2 to be Goldberg's network on G/S with parameter $\tau = \rho(S)$.
3.6 Construct H'_2 by adding an infinite-capacity edge from s to (contracted) S on H_2.
3.7 **return** (the s-t max flow value in $H_1 = c(E[S])$) AND (the t-mincut value in $\tilde{H}_1 \geq \rho(S)$) AND (the s-t max flow value in $H'_2 > |E[V/S]| + \rho(S)$)

Lemma IV.17. *Let H_1 be Goldberg's network of $G[S]$ and $\rho(S)$. Let $\tilde{H}_1$ be the modified network of H_1. Then, $\rho(W) \leq \rho(S)$ for all $W \subseteq S$ if and only if the s-t max flow value in H_1 is $c(E[S])$, and the t-mincut in $\tilde{H}_1$ is at least $\rho(S)$.*

Proof. Let $\tau = \rho(S)$. The condition for $W = \varnothing$ is trivial, so we only consider nonempty W. Note that $\rho(W) \leq \rho(S) \iff c(E[W]) \leq \tau(|W| - 1)$, so the condition is equivalent to

$$\forall W \subseteq S \text{ s.t. } W \neq \varnothing,\ c(E[W]) \leq \tau(|W| - 1). \tag{6}$$

(Sufficiency) Assume the s-t max flow value in H_1 is $c(E[S])$ and the t-mincut in $\tilde{H}_1$ is $\geq \tau$. By Lemma IV.12, for any nonempty $W \subseteq S$,

$$d^+_{\tilde{H}_1}(W) = \tau|W| - c(E[W]) \geq \tau$$

which implies (6).

(Necessity) Assume (6) holds. By Fact IV.8, for any nonempty $W \subseteq S$,

$$d^+_{H_1}(\{s\} \cup W \cup E[W]) = c(E[S]) - c(E[W]) + \tau|W| \geq c(E[S]) + \tau,$$

where the last inequality uses (6). Then by Lemma IV.9, the s-t min cut in H_1 is $\partial^+\{s\}$, which has value $c(E[S])$.

Now we can apply Lemma IV.12. For any nonempty $W \subseteq S$,

$$d^+_{\tilde{H}_1}(W) = \tau|W| - c(E[W]) \geq \tau$$

where the last inequality uses (6). So, the t-mincut in $\tilde{H}_1$ is $\geq \tau$.

□

Lemma IV.18. *Let H_2 be Goldberg's network of $G[V/S]$ and $\rho(S)$. Construct H_2' by adding an infinite-capacity edge from s to (the contracted vertex) S in H_2. Then, $\rho(U) < \rho(S)$ for all $U \supsetneq S$ if and only if the s-t max flow value in H_2' is $> c(E[V/S]) + \rho(S)$.*

Proof. We first show that for $U \supsetneq S$, $\rho(U) < \rho(S) \iff c(E[U/S]) < \rho(S)(|U| - |S|)$.

Assume $\rho(U) < \rho(S)$. Then $\rho(S) > 0$ and $|U| > |S| \geq 2$. By (1),

$$c(E[U]) = \rho(U)(|U| - 1) < \rho(S)(|U| - 1).$$

Then,

$$\begin{aligned} c(E[U/S]) &= c(E[U]) - c(E[S]) \\ &< \rho(S)(|U| - 1) - \rho(S)(|S| - 1) \\ &= \rho(S)(|U| - |S|). \end{aligned}$$

Next, assume $c(E[U/S]) < \rho(S)(|U| - |S|)$. Then $\rho(S) > 0$ and $|U| > |S| \geq 2$. By (1),

$$\begin{aligned} c(E[U]) &= c(E[S]) + c(E[U/S]) \\ &< \rho(S)(|S| - 1) + \rho(S)(|U| - |S|) \\ &= \rho(S)(|U| - 1). \end{aligned}$$

Combined with $c(E[U]) = \rho(U)(|U| - 1)$, we have $\rho(U) < \rho(S)$.

Let $\tau = \rho(S)$. Now, the condition is equivalent to

$$\forall U \supsetneq S,\ c(E[U/S]) < \tau(|U| - |S|). \tag{7}$$

(Sufficiency) Assume the s-t min cut value in H_2' is $> c(E[V/S]) + \tau$. For any $U \supsetneq S$, let $X = U/S$. By Fact IV.8,

$$\begin{aligned} d^+_{H_2'}(\{s\} \cup X \cup E[X]) &= c(E[V/S]) - c(E[X]) + \tau|X| \\ &= c(E[V/S]) - c(E[U/S]) \\ &\quad + \tau(|U| - |S| + 1). \end{aligned}$$

Combined with the assumption that $d^+_{H_2'}(\{s\} \cup X \cup E[X]) \geq (\text{min cut value in } H_2') > c(E[V/S]) + \tau$, we have $c(E[U/S]) < \tau(|U| - |S|)$.

(Necessity) Assume (7) holds. By Lemma IV.10, the s-t min cut on H_2' has source side $\{s\} \cup X \cup E[X]$ for some X that contains the contracted S. Expand the contracted S in X to form U.

By Fact IV.8, the min cut value is

$$\begin{aligned} c(E[V/S]) - c(E[X]) + \tau|X| &= c(E[V/S]) - c(E[U/S]) \\ &\quad + \tau(|U| - |S| + 1) \\ &> c(E[V/S]) + \tau, \end{aligned}$$

where the last inequality uses (7). □

We are now ready to prove Lemma IV.4.

Lemma IV.4 (Verify-Core)**.** *For an input graph G, a set of vertices S, and a parameter k, Algorithm 3 returns true if and only if S is a dense core in G and $|S| \leq k$. The running time is $O(k)$ calls to max-flow on digraphs with $O(m)$ nodes and $O(m)$ edges.*

Proof. The correctness follows from Lemmas IV.17 and IV.18.

Next, we analyze the running time. Constructing H_1, H_2 and H_2' takes $O(m)$ time. Constructing $\tilde{H}_1$ calls a max flow on H_1 and takes $O(F(m, m))$ time. Computing an s-t max flow in H_2 takes $O(F(m, m))$ time. Finally, to compute a t-mincut in $\tilde{H}_1$, we iterate over all $s \in S$, and take the minimum among all s-t mincuts. Because $|S| \leq k$, the running time is $O(k \cdot F(n, m))$. In conclusion the total running time is $O(k \cdot F(m, m))$. □

V. A Faster Algorithm for Arboricity

In this section, we give a faster algorithm for computing the arboricity $\Gamma(G)$ of a graph G, which builds on ideas presented in the previous section. Nash-Williams [21] showed that arboricity is equal to the integer ceiling of the skew density of the skew densest vertex set in G. We denote this maximum skew-density $\tilde{\Gamma}(G)$, i.e., $\Gamma(G) = \lceil \tilde{\Gamma}(G) \rceil$. We will show that a variant of Algorithm 2 computes $\Gamma(G)$ in $\widetilde{O}(1)$ calls to two subroutines – directed global mincut and max-flow – and $\widetilde{O}(m)$ additional time outside these oracle calls. We restate the formal theorem below.

Theorem I.2. *There is a randomized algorithm that, given an undirected graph G on m edges and n vertices with integer edge weights in $\{1, 2, \ldots, C\}$, outputs the arboricity of G in*

$O(m \log(nC))$ time plus $O(\log(nC))$ calls to a maximum flow subroutine on directed graphs with $O(m)$ edges and $O(m)$ vertices and $O(\log(nC))$ calls to a global minimum cut subroutine on directed graphs with $O(m)$ edges and $O(n)$ vertices, both with integer edge weights at most $\mathrm{poly}(n, C)$.

In the directed global mincut problem, the goal is to find a set of vertices $S \subset V$ that minimizes the value of the in-cut, i.e., the sum of capacities of directed edges from $V \setminus S$ to S. Thus far we have considered finding t-mincuts, not global directed mincuts (e.g., Lemma IV.9). These two problems are in fact essentially equivalent.

Fact V.1. *Finding a t-mincut requires one call to a directed global mincut oracle. Conversely, finding a directed global mincut requires two calls to a t-mincut oracle.*

Proof. To find a t-mincut, take the graph $G = (V \cup \{t\}, E, c)$ and add edges (v, t) for each $v \in V$ with capacities $c(v, t) = \infty$. (We can emulate an edge of infinite capacity by setting its capacity larger than the total capacity of the graph.) Then any mincut S must not include the vertex t, i.e., $S \subseteq V$, which corresponds exactly to the t-mincut. To find a directed global min cut in $G = (V, E, c)$, choose an arbitrary vertex $t \in V$ and compute the t-minimum cut in G and in the graph where every edge direction is reversed, and return the smaller of these two cuts. Since t is either on the side of the minimum cut or not, this finds the global minimum cut. □

We now give our algorithm for computing arboricity in Algorithm 4. There is an outer binary search procedure, where, in each step, the guessed value of arboricity is denoted τ. To test if arboricity exceeds τ or is smaller than it, we use two subroutine calls – max flow on Goldberg's network H with parameter τ and t-mincut on the modified Goldberg's network $\tilde{H}$, both of which are defined in Section IV-C.

Algorithm 4: COMPUTE-ARBORICITY(G)

```
4.1  τ_L ← 0, τ_R ← c(E)
4.2  while τ_R − τ_L ≥ n^{-3} do
4.3      τ ← (τ_L + τ_R)/2
4.4      Construct Goldberg's network H from G with
           parameter τ
4.5      if the s-t max flow value on H < c(E) then
4.6          τ_L = τ
4.7      else
4.8          Construct the modified network H̃ from H
4.9          if t-mincut of H̃ < τ then
4.10             τ_L = τ
4.11         else
4.12             τ_R = τ
4.13 Output ⌈τ_L⌉.
```

In the next lemma, we argue correctness of a single iteration of the binary search procedure in Algorithm 4:

Lemma V.2. *The binary search procedure on input τ sets $\tau_L = \tau$ if and only if $\tau < \tilde{\Gamma}(G)$.*

Proof. (Sufficiency) If the algorithm sets $\tau_L = \tau$, there are two cases: Line 4.6 and Line 4.10. In both cases, we show $\tau < \rho(U)$ for some U, which implies $\tau < \tilde{\Gamma}(G)$ because $\tilde{\Gamma}(G)$ is the skew-density of the skew-densest set.

Suppose the algorithm sets $\tau_L = \tau$ on Line 4.6. Let S be the source side of the s-t min cut in H. By Lemma IV.9 and Fact IV.8, $S = \{s\} \cup U_1 \cup E[U_1]$ for some $U_1 \subseteq V$, and $d^+_H(S) = c(E) - c(E[U_1]) + \tau|U_1|$. Since the if statement on Line 4.5 holds, we have $d^+_H(S) < c(E)$ or equivalently $c(E[U_1]) - \tau|U_1| > 0$ and $|U_1| \geq 2$. Then we can apply (1) to get

$$\rho(U_1)(|U_1| - 1) = c(E[U_1]) > \tau|U_1| > \tau(|U_1| - 1),$$

which implies $\tau < \rho(U_1)$.

Suppose the algorithm sets $\tau_L = \tau$ in Line 4.10. Since this did not happen at Line 4.6, the s-t max flow on H has value $c(E)$. Let U_2 be the t-mincut of $\tilde{H}$ found on Line 4.9. By Lemma IV.12, $d^+_{\tilde{H}}(U_2) = \tau|U_2| - c(E[U_2])$. Since the if statement on Line 4.9 succeeds, we have $d^+_{\tilde{H}}(U_2) < \tau$. Then we can apply (1) to get

$$\tau(|U_2| - 1) < c(E[U_2]) = \rho(U_2)(|U_2| - 1)$$

which implies $\tau < \rho(U_2)$.

(Necessity) Next, assume $\tau < \tilde{\Gamma}(G)$. We need to show that the algorithm sets $\tau_L = \tau$. If the algorithm does so at Line 4.6, we are done. Assume it did not, which means the s-t max flow on H is $c(E)$. Let D be a skew-densest set ($|D| \geq 2$). By Nash-Williams [21], we have $\rho(D) = \tilde{\Gamma}(G) > \tau$. By Lemma IV.12,

$$\begin{aligned} d^+_{\tilde{H}}(D) = \tau|D| - c(E[D]) &= \tau|D| - \rho(D)(|D| - 1) \\ &= \tau + (\tau - \rho(D))(|D| - 1) < \tau. \end{aligned}$$

Thus, there exists a t-cut D in $\tilde{H}$ with value $< \tau$, so the value of the t-mincut of $\tilde{H}$ will be found to be $< \tau$ in Line 4.9, as desired. □

We now conclude the proof of Theorem I.2:

Proof of Theorem I.2. By Lemma V.2, each iteration of the binary search sets $\tau_L = \tau$ if and only if $\tau < \tilde{\Gamma}(G)$. Let the values of τ_L and τ_R in the final iteration of binary search be denoted τ^*_L and τ^*_R; then, $\tau^*_R - \tau^*_L < n^{-3}$ and $\tilde{\Gamma}(G) \in [\tau^*_L, \tau^*_R]$. Note that $\tilde{\Gamma}(G)$ can be written as a ratio of two integers with denominator in the range $\{1, \ldots, n-1\}$. Any two distinct rational numbers that can be written in this form must differ by at least n^{-2}. It follows there is only one candidate value of $\tilde{\Gamma}(G)$ in the range $[\tau^*_L, \tau^*_R]$. Moroever, every integer is a candidate. Now, there are two cases. If $[\tau^*_L, \tau^*_R]$ contains an integer, then $\tilde{\Gamma}(G)$ must be equal to this integer since the latter is the unique candidate in this range. Therefore, $\Gamma(G) = \tilde{\Gamma}(G) = \lceil \tau^*_L \rceil$. In the other case, there is no integer in the range $[\tau^*_L, \tau^*_R]$. In this case, since $\tilde{\Gamma}(G) \in [\tau^*_L, \tau^*_R]$, we have

that $\Gamma(G) = \lceil \tilde{\Gamma}(G) \rceil = \lceil \tau_L^* \rceil$. Therefore, $\Gamma(G) = \lceil \tau_L^* \rceil$ in both cases.

Now, we bound the running time of Algorithm 4. Recall that we denote $F(n, m)$ to be the running time of max flow on a directed graph with n vertices and m edges. Similarly, denote $M(n, m)$ to be the running time of global min cut on a directed graph with n vertices and m edges. The algorithm runs $O(\log(nC))$ iterations of binary search because the total edge capacity is at most n^2C and the precision is n^{-3}. In each iteration, Lines 4.4 and 4.8 take $O(m + F(m, m))$ time to construct H and $\tilde{H}$, as shown in the proof of Lemma IV.4. Line 4.5 runs a max flow on H with $O(m)$ vertices and $O(m)$ edges by Fact IV.6. Line 4.9 can be computed by one calls to directed mincut on graphs with m edges and n vertices according to Facts IV.11 and V.1. In conclusion the running time of an iteration is $O(m+M(n, m)+F(m, m))$. The total running time is $O((m + M(n, m) + F(m, m)) \log(nC))$. □

VI. Ideal Tree Packing and Fractional Spanning Tree

In this section, we describe an application of the cut hierarchy to an ideal tree packing, and in turn, to the max-entropy fractional spanning tree.

A. Ideal Load and Ideal Tree Packing

Given a connected undirected graph G with edge weights c_e, Thorup [10], [11] defines the *ideal load* as a function on edges. We restate the definition below based on the cut hierarchy.

For each node p in the cut hierarchy, the node is associated with a vertex set S_p. Let $G_p = (V_p, E_p)$ be the graph formed from the induced subgraph $G[S_p]$ by contracting all vertex sets associated with children of p in the cut hierarchy. By the definition of the cut hierarchy, we have that the min-ratio cut in G_p is the all-singleton cut. For each edge $e \in E$, we associate e with the deepest node in the cut hierarchy whose induced subgraph contains e; we denote this node $p(e)$.

The ideal load is defined by

$$\ell(e) = \frac{c_e}{\sigma(G_{p(e)})} = \frac{c_e}{c(E_{p(e)})/(|V_{p(e)}| - 1)}.$$

The following lemma shows that the ideal load can be computed from the cut hierarchy in $O(m)$ time.

Lemma VI.1. *Given the cut hierarchy of an undirected graph G, the ideal load can be computed in $O(m)$ time.*

Proof. For each edge $e = (u, v)$, $p(e)$ is the least common ancestor (LCA) of u and v (as leaves of the cut hierarchy) in the cut hierarchy. We can use Tarjan's offline LCA to compute $p(e)$ for all $e \in E$ in $O(m)$ time [22].

Note that $\sigma(G_p)$ is the ratio of total weight of edges in G_p over the number of children of p minus 1. By scanning all edges and adding $c(e)$ to the sum of the weight of edges in $G_{p(e)}$, we can obtain the total weight of edges in G_p for all nodes p. In conclusion all $\sigma(G_p)$ and hence all $\ell(e)$ can be computed in $O(m)$ time. □

Importantly, the ideal loads on the edges define a fractional spanning tree [10]. So, there exists a convex combination of spanning trees whose average matches the ideal loads. We show that we can efficiently sample from such a distribution using the cut hierarchy.

Lemma VI.2. *Given the cut hierarchy of a connected undirected graph G, we can construct a data structure in $\widetilde{O}(n^3m)$ time and $O(nm)$ space from which we can sample in $O(n)$ time a spanning tree that belongs to a fixed ideal tree packing.*

Proof. For each node p of the cut hierarchy, we construct a maximal fractional tree packing $\mathbb{T}_p$ in G_p. (By Fact II.3, each G_p is connected, so $\mathbb{T}_p$ is a linear combination of spanning trees in G_p.) Let n_p, m_p be the number of vertices and edges in G_p respectively. The construction time is $\widetilde{O}(n_p^3 m_p)$ for each G_p by using the fractional tree packing algorithm of Gabow and Manu [23]. Since n_p equals the number of children of node p in the cut hierarchy, and the total number of nodes in the cut hierarchy is $\leq 2n$ since they represent a laminar family of subsets of V, we can conclude that $\sum_p n_p \leq 2n$. Similarly, since every edge appears in only one of the graphs G_p, we can conclude that $\sum_p m_p \leq m$. So, the total construction time is $\widetilde{O}(n^3m)$. Gabow-Manu algorithm guarantees that each $\mathbb{T}_p$ consists of at most m_p distinct trees, so the space is at most $\sum_p m_p n_p = O(nm)$.

To draw a sample, we simply draw a random tree from each $\mathbb{T}_p$ (each tree is drawn with probability proportional its weight in the packing), and take the union (in terms of the edges) of all trees drawn. We claim that the resulting graph is a spanning tree. First, by bottom-up induction on the cut hierarchy, we have that each S_p (the vertex set corresponding to node p) is connected by the sample. As a result, the sample spans the whole graph. Second, the number of edges in the union is the sum over p of rank difference of S_p and all children of p. The sum telescopes to be the rank difference of whole graph and all singletons, which is $n - 1$. Because the sample has $n - 1$ edges and is spanning, it must be a spanning tree.

We now compute the probability that ane edge e is chosen in the sample. Notice that e only appears in one of the graphs G_p, namely $G_{p(e)}$. So, we only need to consider the packing $\mathbb{T}_{p(e)}$, which has value $\sigma(G_{p(e)})$. Because the min-ratio cut of $G_{p(e)}$ is the all-singleton cut, all edges are fully used in the maximum fractional tree packing, i.e., e is used in a subset of trees with total value c_e. So, the probability that e is chosen in $\mathbb{T}_{p(e)}$ is $c_e/\sigma(G_{p(e)}) = \ell(e)$, as desired. □

B. Ideal Load as Maximum Entropy Fractional Tree

In this section, we show that the fractional spanning tree defined by the ideal loads maximizes the entropy function in the spanning tree polytope. This could be a natural alternate definition of ideal load.

To validate this observation, we view each integer-weighted edge e as $c(e)$ parallel, unweighted edges. Then, each unweighted edge has ideal load $\ell(e) = 1/\sigma(G_{p(e)})$. Denote x_e to be the value of edge e in the spanning tree polytope. The Shannon entropy of the (normalized) edge values is $H(x) :=$

$\sum_e \frac{x_e}{n-1} \ln \frac{n-1}{x_e} = \frac{1}{n-1}\ln(n-1) - \frac{1}{n-1}\sum_e x_e \ln x_e$. So, maximizing entropy is equivalent to minimizing $\sum_e x_e \ln x_e$ in the spanning tree polytope.

Theorem VI.3. *In an unweighted multigraph, the ideal load is the maximizer of entropy of the edge values in the spanning tree polytope.*

We prove this theorem in the rest of this section. We want to maximize entropy given by $\min \sum_{e\in E} x_e \ln x_e$ subject to the constraints of the spanning tree polytope:

$$x_e \in [0,1]\ \forall e \in E \text{ such that}$$
$$x(E[S]) := \sum_{e\in E[S]} x_e \le |S| - 1\ \forall S \subseteq V \text{ nonempty},$$
$$\sum_{e\in E} x_e = n-1.$$

The first group of constraints can be restricted to $2 \le |S| \le n-1$, because $|S| = 1$ is trivial and $|S| = n$ is included in the normalization constraint. Note also that the constraints $x_e \le 1$ are automatically implied by the subset constraints for pairs of vertices, so we only need to explicitly enforce $x_e \ge 0$ for all $e \in E$.

We introduce Lagrangian multipliers μ_S for $S \subseteq V, |S| \ge 2$. The Lagrangian is

$$L(x,\mu) = \sum_e x_e \ln x_e + \sum_{|S|\ge 2} \mu_S \cdot (x(E[S]) - |S| + 1).$$

The primal program is equivalent to $\max_\mu \min_x L(x,\mu)$ where $\mu_S \ge 0$ for sets with $2 \le |S| \le n-1$, and μ_V is unconstrained. Next we solve $g(\mu) := \min_{x\ge 0} L(x,\mu)$ to obtain the dual program $\max_\mu g(\mu)$. Let x^* denote $\arg\min_{x\ge 0} L(x,\mu)$. Taking the partial derivatives at $x = x^*$, we get that for every edge e,

$$1 + \ln x_e^* + \sum_{S: e\in E[S]} \mu_S = 0,$$

which implies that

$$x_e^* = \exp\left(-1 - \sum_{S:e\in E[S]} \mu_S\right).$$

Define

$$y_S = \begin{cases} \mu_S & S \subseteq V,\ 2 \le |S| \le n-1 \\ \mu_S + 1 & S = V, \end{cases}$$

so that $x_e^* = \exp(-\sum_{S:e\in E[S]} y_S)$. Substituting in the Lagrangian L, we get the dual objective

$$\begin{aligned} &= \sum_e x_e^* \cdot \left(-\sum_{S:e\in E[S]} y_S\right) + \sum_S y_S \cdot \left(\sum_{e\in E[S]} x_e^* - |S| + 1\right) \\ &\quad - (x^*(E) - n + 1) \\ &= \left(\sum_S y_S \cdot (-|S| + 1)\right) - (x^*(E) - n + 1) \\ &= n - 1 - \sum_e \exp\left(-\sum_{S:e\in E[S]} y_S\right) - \sum_S y_S(|S| - 1). \end{aligned}$$

The dual program is

$$\begin{aligned} \max\ g(y) := n - 1 &- \sum_e \exp\left(-\sum_{S:e\in E[S]} y_S\right) \\ &- \sum_S y_S(|S| - 1) \\ \text{subject to}\ \ y_S \ge 0\ \forall S \subseteq V,\ 2 \le |S| &\le n-1. \end{aligned}$$

Because the ideal load is in the spanning tree polytope, it is a feasible primal solution. We now construct a dual solution y^* and show its dual objective matches the primal objective for ideal load. It follows that both solutions are optimal.

For each non-root node p in the cut hierarchy with parent q, set $y^*_{S_p} := \ln \sigma(G_p) - \ln \sigma(G_q)$, where q is the parent of p in the cut hierarchy. For the root node corresponding to V, set $y^*_V := \ln \sigma(G)$. Other sets S that do not appear in the cut hierarchy are assigned $y^*_S = 0$. It is easy to verify the next property, which implies dual feasibility:

Fact VI.4. *Consider any non-root node p of the cut hierarchy and let q denote its parent. Then, $\sigma(G_p) \ge \sigma(G_q)$, which implies $y^*_{S_p} \ge 0$.*

Proof. Consider any non-root node p and let q be p's parent. Assume for contradiction that $\sigma(G_p) < \sigma(G_q)$. Let $\mathcal{P}, \mathcal{Q}$ be the maximal min-ratio cuts in $G[S_p], G[S_q]$ respectively. Note that S_p is a side in $\mathcal{Q}$. Let $\mathcal{Q}' = (\mathcal{Q} \setminus \{S_p\}) \cup \mathcal{P}$. The cut ratio of $\mathcal{Q}'$ is

$$\begin{aligned} \frac{d_{G[S_q]}(\mathcal{Q}')}{|\mathcal{Q}'| - 1} &= \frac{d_{G[S_q]}(\mathcal{Q}) + d_{G[S_p]}(\mathcal{P})}{|\mathcal{Q}| - 1 + |\mathcal{P}| - 1} \\ &= \frac{\sigma(G_q)(|\mathcal{Q}| - 1) + \sigma(G_p)(|\mathcal{P}| - 1)}{|\mathcal{Q}| - 1 + |\mathcal{P}| - 1} < \sigma(G_q), \end{aligned}$$

which contradicts the fact that $\mathcal{Q}$ is a min-ratio cut in $G[S_q]$. □

Because the sets form a laminar family, we have $\sigma(G_p) = \exp\left(\sum_{q:S_q \supseteq S_p} y^*_{S_q}\right)$. Therefore,

$$\ell(e) = \frac{1}{\sigma(G_{p(e)})} = \exp\left(-\sum_{S:e\in E[S]} y^*_S\right). \tag{8}$$

Applying (8) to $g(y^*)$, we have

$$g(y^*) = n-1-\sum_e \ell(e)-\sum_S y^*_S(|S|-1) = -\sum_S y^*_S(|S|-1).$$

Here, the last step uses $\sum_{e\in E}\ell(e) = n-1$ because ℓ is a fractional spanning tree. On the other hand,

$$\begin{aligned}\sum_e \ell(e)\ln\ell(e) &= -\sum_e \ell(e)\sum_{S:e\in E[S]} y^*_S \\ &= -\sum_S y^*_S \sum_{e\in E[S]} \ell(e) \\ &= -\sum_S y^*_S \cdot \ell(E[S]).\end{aligned}$$

We can ignore the terms with $y^*_S = 0$. The remaining S with $y^*_S \neq 0$ are sets in the cut hierarchy. It remains to show that for such sets, $\ell(E[S]) = |S|-1$.

Note that if we define the ideal load in a subgraph $G[S_p]$ for a node p in the cut hierarchy, the definition is identical to $\ell(e)$ because the cut hierarchy in $G[S_p]$ is a sub-family of the cut hierarchy in G. So, ℓ is also a fractional spanning tree in $G[S_p]$, and we have $\sum_{e\subseteq S_p}\ell(e) = |S_p|-1$. In conclusion $g(y^*) = \sum_e \ell(e)\ln\ell(e)$, which implies that $x^*_e = \ell(e)$ is an optimal solution for the primal program.

VII. Closing Remarks

In this paper, we have presented new algorithms for constructing the cut hierarchy of a weighted, undirected graph in $nm^{1+o(1)}$ time, and for determining its arboricity in $\sqrt{n}m^{1+o(1)}$ time. These algorithms improve the state of the art for both problems, previously due to Gabow [5], from $\widetilde{O}(n^2m)$ (weighted) and $\widetilde{O}(nm^{3/2})$ (unweighted) for the cut hierarchy problem and $\widetilde{O}(nm)$ (weighted) and $\widetilde{O}(m^{3/2})$ (unweighted) for the arboricity problem. Our arboricity algorithm is bottlenecked by the running time of a directed minimum cut subroutine – if that improves to $m^{1+o(1)}$, the entire algorithm will run in $m^{1+o(1)}$ time. Nevertheless, it would be interesting to explore whether this dependence on the directed minimum cut subroutine is necessary – apriori, there is no reason to believe that it is. Improving the running time for the cut hierarchy problem beyond that shown in this paper is also an interesting goal. Among other implications, this would result in a faster algorithm for computing the strength of a graph beyond the current best running time of $\widetilde{O}(nm)$.

Acknowledgment

RC and DP were supported in part by NSF grants CCF-1955703 and CCF-2329230. Part of this work was done when RC, JL, and DP were visiting the Simons Institute for Theory of Computing, UC Berkeley, as part of the semester program on "Data Structures and Optimization for Fast Algorithms". HF and GL are supported by the National Science Foundation Graduate Research Fellowship Program under NSF grant DGE2140739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

DP also gratefully acknowledges the support of Google Research where he was a visitor during part of this work. GL thanks Jeff Giliberti for mentioning the problem of computing arboricity to him.

References

[1] A. Schrijver, *Combinatorial Optimization*. Springer, 2003.

[2] J. Picard and M. Queyranne, "A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory," *Networks*, vol. 12, no. 2, pp. 141–159, 1982. [Online]. Available: https://doi.org/10.1002/net.3230120206

[3] M. Padberg and L. A. Wolsey, "Fractional covers for forests and matchings," *Math. Program.*, vol. 29, no. 1, pp. 1–14, 1984. [Online]. Available: https://doi.org/10.1007/BF02591725

[4] H. N. Gabow and H. H. Westermann, "Forests, frames, and games: Algorithms for matroid sums and applications," *Algorithmica*, vol. 7, no. 5&6, pp. 465–497, 1992. [Online]. Available: https://doi.org/10.1007/BF01758774

[5] H. N. Gabow, "Algorithms for graphic polymatroids and parametric s-sets," in *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '95, USA, 1995, p. 88–97. [Online]. Available: https://dl.acm.org/doi/10.5555/313651.313667

[6] D. Eppstein, "Arboricity and bipartite subgraph listing algorithms," *Information processing letters*, vol. 51, no. 4, pp. 207–211, 1994. [Online]. Available: https://doi.org/10.1016/0020-0190(94)90121-X

[7] S. A. Plotkin, D. B. Shmoys, and É. Tardos, "Fast approximation algorithms for fractional packing and covering problems," *Mathematics of Operations Research*, vol. 20, no. 2, pp. 257–301, 1995. [Online]. Available: https://doi.org/10.1287/moor.20.2.257

[8] B. M. T. Worou and J. Galtier, "Fast approximation for computing the fractional arboricity and extraction of communities of a graph," *Discrete Applied Mathematics*, vol. 213, pp. 179–195, 2016. [Online]. Available: https://doi.org/10.1016/j.dam.2014.10.023

[9] M. Blumenstock and F. Fischer, "A constructive arboricity approximation scheme," in *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 2020, pp. 51–63. [Online]. Available: https://doi.org/10.1007/978-3-030-38919-2_5

[10] M. Thorup, "Fully-dynamic min-cut," in *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, ser. STOC '01, 2001, p. 224–230. [Online]. Available: https://doi.org/10.1145/380752.380804

[11] ——, "Minimum k-way cuts via deterministic greedy tree packing," in *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, ser. STOC '08, 2008, p. 159–166. [Online]. Available: https://doi.org/10.1145/1374376.1374402

[12] T. de Vos and A. B. Christiansen, "Tree-packing revisited: Faster fully dynamic min-cut and arboricity," in *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2025, pp. 700–749. [Online]. Available: https://doi.org/10.1137/1.9781611978322.21

[13] V. Trubin, "Strength of a graph and packing of trees and branchings," *Cybernetics and Systems Analysis*, vol. 29, no. 3, pp. 379–384, 1993. [Online]. Available: https://doi.org/10.1007/BF01125543

[14] F. Barahona, "Separating from the dominant of the spanning tree polytope," *Operations Research Letters*, vol. 12, no. 4, pp. 201–203, 1992. [Online]. Available: https://doi.org/10.1016/0167-6377(92)90045-5

[15] C. S. A. Nash-Williams, "Edge-Disjoint Spanning Trees of Finite Graphs," *Journal of the London Mathematical Society*, vol. s1-36, no. 1, pp. 445–450, 01 1961. [Online]. Available: https://doi.org/10.1112/jlms/s1-36.1.445

[16] W. T. Tutte, "On the Problem of Decomposing a Graph into n Connected Factors," *Journal of the London Mathematical Society*, vol. s1-36, no. 1, pp. 221–230, 01 1961. [Online]. Available: https://doi.org/10.1112/jlms/s1-36.1.221

[17] L. Chen, R. Kyng, Y. P. Liu, S. Meierhans, and M. P. Gutenberg, "Almost-linear time algorithms for incremental graphs: Cycle detection, sccs, *s-t* shortest path, and minimum-cost flow," *STOC'24*, 2024. [Online]. Available: https://doi.org/10.1145/3618260.3649745

[18] J. Van Den Brand, L. Chen, R. Peng, R. Kyng, Y. P. Liu, M. P. Gutenberg, S. Sachdeva, and A. Sidford, "A deterministic almost-linear time algorithm for minimum-cost flow," in *2023 IEEE 64th Annual Symposium on Foundations of Computer Science*

(FOCS). IEEE, 2023, pp. 503–514. [Online]. Available: https://doi.org/10.1109/FOCS57990.2023.00037

[19] R. Cen, J. Li, D. Nanongkai, D. Panigrahi, T. Saranurak, and K. Quanrud, "Minimum cuts in directed graphs via partial sparsification," in *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022, pp. 1147–1158. [Online]. Available: https://doi.org/10.1109/FOCS57990.2023.00037

[20] A. V. Goldberg, "Finding a maximum density subgraph," University of California at Berkeley, USA, Tech. Rep., 1984. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/1984/CSD-84-171.pdf

[21] C. S. A. Nash-Williams, "Decomposition of finite graphs into forests," *Journal of the London Mathematical Society*, vol. s1-39, no. 1, pp. 12–12, 1964. [Online]. Available: https://doi.org/10.1016/j.jctb.2011.04.001

[22] R. E. Tarjan, "Applications of path compression on balanced trees," *Journal of the ACM*, vol. 26, no. 4, pp. 690–715, 1979. [Online]. Available: https://doi.org/10.1145/322154.32216

[23] H. N. Gabow and K. S. Manu, "Packing algorithms for arborescences (and spanning trees) in capacitated graphs," in *Integer Programming and Combinatorial Optimization*. Springer Berlin Heidelberg, 1995, pp. 388–402. [Online]. Available: https://doi.org/10.1007/3-540-59408-6_67

[24] N. E. Young, "Randomized rounding without solving the linear program," in *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, 1995, pp. 170–178. [Online]. Available: https://dl.acm.org/doi/10.5555/313651.313689

[25] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan, "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs," *Combinatorica*, vol. 6, no. 2, pp. 109–122, 1986. [Online]. Available: https://doi.org/10.1007/BF02579168

APPENDIX A
MINIMUM DIRECTED CUT (PROOF OF LEMMA IV.14)

In this section, we design an algorithm to solve the following subproblem: Given a digraph G with a root vertex t, cut value parameter τ, and size parameter k, assuming there exists a t-cut S with $d^+(S) < \tau$ and $|S| \le k$, find any t-cut S' with $d^+(S') < \tau$.

The cut we find will be a minimum 1-respecting t-cut. We define a t-arborescence to be a directed subgraph where each vertex except t has out-degree 1, t has out-degree 0, and each vertex has a directed path to t. We say a t-cut 1-respects an arborescence if the intersection of cut edges and arborescence edges is 1.

Lemma A.1. *There exists an algorithm that, given a directed graph G with integer edge capacity in $\{1, 2, \ldots, C\}$ and a t-arborescence T, outputs the minimum t-cut that 1-respects T in $\widetilde{O}(m)$ time plus polylogarithmic calls to max flow on digraphs with $O(m)$ edges and $O(n)$ vertices and integer edge weights at most* $\mathrm{poly}(n, C)$.

Proof. The lemma is a corollary of the following result:

Lemma A.2 (Theorem 2.7 of [19])**.** *There exists an algorithm that, given a capacitated digraph G and a t-arborescence T that 1-respects a t-mincut, output a (not necessarily* 1*-respecting) t-mincut in $\widetilde{O}(m)$ time plus polylogarithmic calls to max flow.*

(We remark that our direction of t-cut and t-arborescence is opposite to [19], but the results are equivalent after reversing orientation of all arcs.)

The only difference to the lemma above is that we require the output t-mincut to be 1-respecting. By adding a uniform large capacity on all edges of T, we can force the t-mincut to be 1-respecting, while keeping the relative value of all 1-respecting cuts. It follows that the original minimum 1-respecting cut is now the global t-mincut, and we can apply Lemma A.2 to find it. □

It remains to construct a t-arborescence that 1-respects a cut of value $< \tau$. Our approach is adapted from [19].

Lemma A.3. *Given a capacitated digraph G with a fixed root vertex t and a number $\tau > 0$, suppose there exists a t-cut S with $d^+(S) < \tau$ and $|S| \le k$. There exists a randomized algorithm that can compute a sparsifier $G_0 = (V_0, E_0)$ in $\widetilde{O}(m)$ time, such that $t \in V_0 \subseteq V$ and the following properties hold.*

1) *G_0 has integer edge capacities and the t-mincut has value $O(k \log(n)/\varepsilon^2)$.*
2) *There exists some S' with $d^+_G(S') < \tau$ such that S' is a $(1+\varepsilon)$-approximate t-mincut in G_0.*

Proof. Let $\mu = c_\mu \varepsilon^2 \tau/(k \log n)$ for a sufficiently small $c_\mu = \Theta(1)$ such that τ and $\varepsilon\tau/2k$ are integer multiples of μ (we may decrease ε by a constant factor). Let G_0 be a capacitated graph obtained as follows.

1) Randomly round each edge capacity independently up or down to the nearest multiple of μ, such that the expectation is w_e (i.e., if $w_e = K\mu + r$, $K \in \mathbb{Z}$, $r \in [0, \mu)$, sample $w'_e = K\mu$ w.p. $1 - \frac{r}{\mu}$, and $w'_e = (K+1)\mu$ w.p. $\frac{r}{\mu}$).
2) Add an edge of capacity $\varepsilon\tau/2k$ from each $v \in V \setminus \{t\}$ to t.
3) Scale down all the edge capacities by μ.

For each set $S \subseteq V$, let $d_1(S)$ be the original cut value, $d_2(S)$ be the cut value after step 1, and $d_3(S)$ be the cut value after step 2.

Consider any fixed $S \subseteq V \setminus \{t\}$. $d_2(S)$ is the sum of $|\partial^+ S|$ independent random variables, and has expectation $d_1(S)$. Each random variable in the sum is nonnegative and varies by at most μ. By a variation of Chernoff bound,[3]

$$\Pr[d_2(S) \le (1-\varepsilon) d_1(S) - \gamma] \le e^{-\varepsilon\gamma/\mu} = n^{-\gamma k/(c_\mu \varepsilon \tau)},$$

$$\Pr[d_2(S) \ge (1+\varepsilon) d_1(S) + \gamma] \le e^{-\varepsilon\gamma/\mu} = n^{-\gamma k/(c_\mu \varepsilon \tau)}.$$

Say that set S is (ϵ, γ)-approximated if $(1-\epsilon) d_1(S) - \gamma \le d_2(S) \le (1+\epsilon) d_1(S) + \gamma$. For $\gamma = \varepsilon\tau|S|/2k$, the probability that a set S is not (ε, γ)-approximated is at most $2n^{-c_0|S|}$, where $c_0 = \frac{1}{2c_\mu}$. Note that there are $O(n^{|S|})$ sets of size $|S|$. For sufficiently large c_0 (sufficiently small c_μ), we can apply

[3] Here we apply the following bounds (appropriately rescaled) which follow from the same proof as the standard multiplicative Chernoff bound.

Let $X_1, \ldots, X_n \in [0,1]$ independent random variables. Then for all $\varepsilon > 0$ sufficiently small and all $\gamma > 0$,

$$\Pr[X_1 + \cdots + X_n \le (1-\varepsilon)\mathbb{E}[X_1 + \cdots + X_n] - \gamma] \le e^{-\varepsilon\gamma},$$

and

$$\Pr[X_1 + \cdots + X_n \ge (1+\varepsilon)\mathbb{E}[X_1 + \cdots + X_n] + \gamma] \le e^{-\varepsilon\gamma}.$$

union bound to conclude that (ε, γ)-approximation holds for all subsets S, i.e., w.h.p. for all $S \subseteq V$,

$$|d_2(S) - d_1(S)| \leq \varepsilon d_1(S) + \frac{\varepsilon\tau|S|}{2k}.$$

Observe that $d_3(S) = d_2(S) + |S| \cdot \frac{\varepsilon\tau}{2k}$ for all t-cuts S by construction, so this implies

$$(1-\varepsilon)d_1(S) \leq d_3(S) \leq (1+\varepsilon)d_1(S) + \frac{\varepsilon\tau|S|}{k}.$$

Next, we prove properties assuming the above high probability event holds. We assumed there exists a t-cut S with $d_1(S) = d^+(S) < \tau$ and $|S| \leq k$. Then,

$$d_3(S) \leq (1+\varepsilon)d_1(S) + \frac{\varepsilon\tau|S|}{k} \leq (1+2\varepsilon)\tau.$$

After scaling down by μ, the min cut value is $\leq d_3(S)/\mu = O(\tau/\mu) = O(k \log n)$. This establishes property 1.

For property 2, we divide into two cases. We discuss the cut value in d_3, which is equivalent to G_0 up to scaling by μ. Let λ be the t-mincut value.

Case 1: $\lambda \geq (1-\varepsilon)\tau$. Then, S is an $(1+O(\varepsilon))$-approximate t-mincut.

Case 2: $\lambda < (1-\varepsilon)\tau$. Suppose S' is the t-mincut in G_0 with $d_3(S') = \lambda$. We have $d_1(S') \leq \frac{1}{1-\varepsilon} d_3(S') < \tau$. So S' satisfies property 2. □

Lemma A.4. *Given a digraph G with integer edge capacities, a fixed root vertex t and a number k such that the (unknown) t-mincut value is $\lambda \leq k$, we can construct in $\widetilde{O}(mk)$ time a fractional packing of t-arborescences of value $\geq (1-\varepsilon)\lambda$.*

Proof. We can write the fractional t-arborescences packing problem as a packing LP. Denote $\mathcal{A} = \{A_1, \ldots, A_N\}$ as the family of all t-arborescences. We represent a fractional packing of t-arborescences as a vector $x \in \mathbb{R}^N$ where x_i is the coefficient of A_i. Define the value of an arborescence packing as the sum of all coefficients $\mathrm{val}(x) = \sum_{i=1}^N x_i$. The goal is to maximize $\mathrm{val}(x)$, under the constraints that each edge j can be fractionally used up to its capacity $c(j)$.

Equivalently, we can scale down all coefficients by $\mathrm{val}(x)$ for a solution x, so that the value of scaled packing becomes 1. Then, the goal becomes to minimize among all edges j the ratio between usage of j and $c(j)$. Our problem can be stated in the framework of a standard packing problem:

Definition A.5 (Packing problem [24])**.** For convex set $P \subseteq \mathbb{R}^n$ and nonnegative linear function $f : P \to \mathbb{R}^m$, the packing problem aims to find $\gamma^* = \min_{x \in P} \max_{j \in [m]} f_j(x)$ i.e., the solution in P that minimizes the maximum value of $f_j(x)$ over all j. The width of the packing problem (P, f) is defined as $\omega = \max_{j \in [m], x \in P} f_j(x) - \min_{j \in [m], x \in P} f_j(x)$.

In our setting, $P = \{x \in \mathbb{R}^N : \mathrm{val}(x) = 1, x \geq 0\}$ is the convex hull of all t-arborescences, and $f_j(x) = \frac{\sum_{i \in [N]: j \in A_i} x_i}{c(j)}$ is the ratio of usage over capacity of an edge j. The width is at most $1/c_{min}$, where c_{min} is the minimum edge capacity. Because the edge capacities are integers, the width is at most 1.

Next we describe the packing algorithm [24]. Maintain a vector $y \in \mathbb{R}^m$; initially set to $y = 1$. In each iteration, find $x = \arg\min_{x \in P} \sum_j y_j f_j(x)$, and then add x to set S and replace y by the vector y' defined by $y'_j = y_j(1 + \varepsilon f_j(x)/\omega)$. After a number of iterations, return $\bar{x} \in P$, the average of all the vectors x over the course of the algorithm. The lemma below upper bounds the number of iterations that suffice:

Lemma A.6 (Corollary 6.3 of [24])**.** *After $\lceil \frac{(1+\varepsilon)\omega \ln m}{\gamma^*((1+\varepsilon)\ln(1+\varepsilon) - \varepsilon)} \rceil$ iterations of the packing algorithm, $\bar{\gamma} = \max_j f_j(\bar{x}) \leq (1+\varepsilon)\gamma^*$.*

By duality between t-arborescence packing and minimum t-cut, we have the value of maximum t-arborescence packing is λ (Corollary 2.1 of [5]), and the optimal value of scaled packing problem is $\gamma^* = \frac{1}{\lambda}$. We run $C \cdot k \log n \geq C \cdot \lambda \log n$ iterations of the packing algorithm for a sufficiently large constant C. Then by Lemma A.6, the value of the output fractional packing is $\leq (1+\varepsilon)\gamma^*$. After scaling back the solution to a fractional packing, its value is $\geq \frac{1}{1+\varepsilon}\lambda$.

Next we analyze the time complexity. Each iteration computes $x = \arg\min_{x \in P} \sum_j y_j f_j(x)$, which can be solved by a minimum cost t-arborescence algorithm in $O(m + n \log n)$ time [25]. We run $\widetilde{O}(k)$ iterations, so the total running time is $\widetilde{O}(mk)$. □

Lemma A.7. *Given a capacitated digraph G with a fixed root vertex t and a number $\tau > 0$, suppose there exists a t-cut S with $d^+(S) < \tau$ and $|S| \leq k$. In $\widetilde{O}(mk)$ time, we can find $O(\log n)$ t-arborescences on vertex set $V_0 \supseteq S^*$, such that the following holds w.h.p.: There exists a t-cut S'' with $d^+(S'') < \tau$ and a t-arborescence T in the packing, such that T 1-respects S''.*

Proof. We first apply Lemma A.3 to obtain sparsifier G_0 and guarantee some S' with $d^+_G(S') < \tau$ has $d^+_{G_0}(S') \leq (1+\varepsilon)\lambda$, where λ is the t-mincut value in G_0, and ε is set to be 0.1. Then, we apply Lemma A.4 to obtain a fractional arborescence packing $\mathcal{T}$ with value $\geq (1-\varepsilon)\lambda$. The expected intersection size $\mathbb{E}_{T \sim \mathcal{T}}[d^+_T(S')] \leq \frac{(1+\varepsilon)\lambda}{(1-\varepsilon)\lambda} \leq 1.5$. By Markov's inequality, with constant probability a random tree $T \in \mathcal{T}$ has intersection size < 2, which must be 1 as an integer. So, we have constant probability to sample a 1-respecting t-arborescence, and we can obtain a 1-respecting t-arborescence w.h.p. by sampling $O(\log n)$ arborescences from the packing. Finally, let S'' be the minimum t-cut 1-respected by T, then $d^+_G(S'') \leq d^+_G(S') < \tau$. □

By applying Lemma A.1 on each of the $O(\log n)$ t-arborescences sampled and output the minimum t-cut found, we obtain a t-cut S' with $d^+(S') < \tau$.

Next, we analyze the running time. The partial sparsification step in Lemma A.3 takes $\widetilde{O}(m)$ time. The tree packing step in Lemma A.4 takes $\widetilde{O}(mk)$ time. Finally, computing 1-respecting min cut on $O(\log n)$ trees takes polylogarithmic

calls to max flow by Lemma A.1. In conclusion, the running time is $\widetilde{O}(mk)$ plus polylogarithmic calls to max flow.

We have established Lemma IV.14.

APPENDIX B
DISCUSSION OF TRUBIN'S MIN-RATIO CUT ALGORITHM

Trubin's min-ratio cut algorithm [13] has aesthetic similarities to ours. Our algorithm proceeds by finding a dense core, contracting, and repeating this process until we build the whole cut hierarchy. Trubin's algorithm iteratively finds a so-called "maximal strength HS-induced subgraph," contracts that subgraph, and repeats to build a cut hierarchy. Here a maximal strength HS-induced subgraph is an induced subgraph of G with maximal strength and with strength equal to its fractional arboricity. (The fractional arboricity of a graph is the skew-density of its skew-densest subgraph.) Trubin attempts to prove that the top level of the hierarchy exhibits the same cost min-ratio cut as that of the original graph via the following lemma.

Lemma B.1 ([13])**.** *Let $S \subseteq V$. The strength of G is equal to the strength of G/S if the strength of $G[S]$ is at least the strength of G.*

The maximum strength of an HS-induced subgraph is in fact the fractional arboricity of G. It is easy to show this using our results. In particular, let $S \subseteq V$ be of maximum size such that

$$\rho(S) = \max_{U \subseteq V \text{ nonempty}} \rho(U).$$

Then, trivially S is a maximum skew-densest subset of G. As such, by Fact III.2 and Lemma III.5, we get that S is a star set of the canonical cut hierarchy. Consequently, $\rho(S)$ is equal to the strength of $G[S]$—its min-ratio cut is the trivial multi-way cut into singletons as it is a star set. By our choice of S, the strength of $G[S]$ is then equal to its fractional arboricity.

Finally, since the fractional arboricity of G is at least $\rho(V)$ and in turn at least its strength (by considering the trivial multi-way cut), we obtain correctness of the non-optimized version of Trubin's algorithm.

However, the more optimized algorithm given in [13] is erroneous. We sketch the algorithm and then give a counterexample disproving its correctness. First, for each $v \in V$ and $\lambda \in [0, \infty)$ solve

$$\max_{S \ni v} \{c(E[S]) - \lambda(|S| - 1)\}. \tag{9}$$

Solutions for a given v can be represented by an increasing sequence of sets

$$\{v\} = X(\lambda_1^v, v) \subseteq X(\lambda_2^v, v) \cdots X(\lambda_{\ell_v}^v, v) = V,$$

where $\lambda_1^v \geq \lambda_2^v \geq \cdots \geq \lambda_{\ell_v}^v = 0$ and, for $i \geq 2$, $X(\lambda_i^v, v)$ maximizes eq. (9) for $\lambda \in [\lambda_i^v, \lambda_{i-1}^v]$. We also have $X(\lambda_1^v, v)$ maximizes eq. (9) for $\lambda \geq \lambda_1$.

Next, adhere to the following algorithm:

1) Let $u \in V$ with λ_1^u maximum over vertices with corresponding sequence of sets of length at least 2 (if none exist, terminate).
2) Contract $X(\lambda_2^u, u)$ in G (setting G to $G/X(\lambda_2^u, u)$), and replace all instances of elements in $X(\lambda_2^u, u)$ in computed sets with u. For u's increasing sequence of sets, remove the first set and reduce the indices of the λ_i^u's by 1. Return to step 1.

At the end, the final contracted graph should have each vertex a side of a min-ratio cut of G.

To show that this algorithm is incorrect, we consider a small graph, construct the sequences of sets corresponding to each vertex by hand, and then show that the algorithm does not terminate in a min-ratio cut of the original graph. Consider a path on vertices a, b, c, d with edges of weight 2, 1, and 100 between a, b; b, c; and c, d, respectively. We can compute the increasing sequences of sets by hand:

- For a, we get $\{a\} \subseteq \{a, c, d\} \subseteq \{a, b, c, d\}$ with $\lambda_1^a = 50, \lambda_2^a = 3$, and $\lambda_3^a = 0$.
- For b, we get $\{b\} \subseteq \{b, c, d\} \subseteq \{a, b, c, d\}$, with $\lambda_1^b = 50.5, \lambda_2^b = 2$, and $\lambda_3^b = 0$.
- For c, we get $\{c\} \subseteq \{c, d\} \subseteq \{a, b, c, d\}$ with $\lambda_1^c = 100, \lambda_2^c = 1.5$, and $\lambda_3^c = 0$.
- Finally, for d, we get $\{d\} \subseteq \{c, d\} \subseteq \{a, b, c, d\}$ with $\lambda_1^d = 100, \lambda_2^d = 1.5$, and $\lambda_3^d = 0$.

Observe that the min-ratio cut of this graph has sides $\{a, b\}$ and $\{c, d\}$. However, we can see that it is not possible to follow the algorithm and contract $\{a, b\}$ into a single node: $\{a, b\}$ never appears among the sequences over the course of the algorithm.