Unbreakable Decomposition in Close-to-Linear Time

Aditya Anand* Euiwoong Lee† Jason Li‡ Yaowei Long§
Thatchaphol Saranurak¶

Abstract

Unbreakable decomposition, introduced by [CLP+19, CKL+20], has proven to be one of the most powerful tools for parameterized graph cut problems in recent years. Unfortunately, all known constructions require at least $\Omega_k\left(mn^2\right)$ time, given an undirected graph with n vertices, m edges, and cut-size parameter k. In this work, we show the first close-to-linear time parameterized algorithm that computes an unbreakable decomposition. More precisely, for any $0 < \epsilon \leq 1$, our algorithm runs in time $2^{O(\frac{k}{\epsilon}\log\frac{k}{\epsilon})}m^{1+\epsilon}$ and computes a $(O(k/\epsilon), k)$ unbreakable tree decomposition of G, where each bag has adhesion at most $O(k/\epsilon)$.

This immediately opens up possibilities for obtaining close-to-linear time algorithms for numerous problems whose only known solution is based on unbreakable decomposition.

*University of Michigan, Ann Arbor
†University of Michigan, Ann Arbor. Supported in part by NSF grant CCF-2236669 and Google.
‡Carnegie Mellon University
§University of Michigan, Ann Arbor
¶University of Michigan, Ann Arbor. Supported by NSF Grant CCF-2238138.

Contents

1 Introduction

For the past two decades, the study of graph cut problems has been a highly active subarea of fixed-parameter tractability (FPT) algorithms that has led to many powerful algorithmic techniques, including important separators and shadow removal [CLL+08, MR14, CHM13, LM13], matroid-based kernelization [KW12, KW14, CDK+21, HLW21, Wah22], treewidth reduction [MOR13], branching from half-integral solutions [CPPW13, Gui11, LNR+14, IWY16, Iwa16, IYY18], and flow augmentation [KKPW21, KKPW24, KMP+24].

Recently, *unbreakable decomposition* has emerged as one of the most powerful techniques used by numerous FPT algorithms [CLP+19, CKL+20, LSS22, PSS+22, SSS+24, SZ23, ILSS23, LSSZ19, AKP+22]. This decomposition also generalizes the highly influential tree decomposition of bounded width to general graphs, in the sense that unbreakable decomposition exists on arbitrary graphs, and the two concepts coincide on bounded-treewidth graphs. See Appendix C.

We briefly define unbreakable decomposition here. Given a graph G and a vertex set X, a (q,k)-*breakable witness* for X is a vertex cut (L,R) in G of size $|L\cap R|\leq k$ where $|L\cap X|,|R\cap X|>q$. If there is no (q,k)-breakable witness for X, then X is (q,k)-*unbreakable*. We say that X has *adhesion* σ if, for every connected component C in $G\setminus X$, the size of its neighborhood is at most $|N_G(C)|\leq\sigma$. A (q,k)-unbreakable decomposition with adhesion σ is a tree decomposition of G where every bag is (q,k)-unbreakable and has adhesion at most σ. The quality of the decomposition is measured by how small q and σ are compared to k.

This paper presents the first close-to-linear time FPT algorithm for computing unbreakable decomposition, thereby removing the core bottleneck to close-to-linear time FPT algorithms for many graph cut problems.[1] Below, we survey the development and impact of unbreakable decomposition.

History. In their breakthrough FPT algorithm for Minimum p-Way Cut[2], Kawarbayashi and Thorup [KT11] introduced the edge cut version of (q,k)-*breakable witnesses*. However, their algorithm for finding these witnesses is problem-specific. A year later, Chitnis et al. [CCH+16] presented a general algorithm for finding (q,k)-breakable witnesses for both the edge cut and vertex cut versions, leading to FPT algorithms for many problems including the terminal version of Minimum p-Way Cut and Unique Games, which in turn generalizes Subset Feedback Vertex Set and Odd Cycle Transversal. Unfortunately, the top-down divide-and-conquer technique used in [KT11, CCH+16] is ineffective in solving a prominent problem, namely, Minimum Bisection.

To address this, Cygan et al. [CLP+19] introduced the concept of *unbreakable decomposition*, which allows them to exploit (q,k)-unbreakable sets in a bottom-up manner using dynamic programming. Using this technique, they successfully developed an FPT algorithm for Minimum Bisection. Later, in [CKL+20], Cygan et al. significantly improved the construction in [CLP+19] by showing an FPT algorithm that constructs an unbreakable decomposition with optimal unbreakability and adhesion parameters. This improvement resulted in further applications.

Since then, unbreakable decomposition has become a core technique in various results within the field. For instance, Lokshtanov, Saurabh, and Surianarayanan [LSS22] provided a polynomial-time construction of unbreakable decomposition in the edge cut version and used it to settle a long line of work of [GLL18b, GLL18a, KL20] by showing a $(1+\epsilon)$-approximation FPT algorithm for Minimum p-Way Edge-Cut when parameterized by p (instead of the usual cut size). Other

[1] As per convention in the graph algorithms literature, given a graph with m edges, algorithms with *near-linear*, *almost-linear*, and *close-to-linear* time have running time of $O(m\log^{O(1)}m)$, $O(m^{1+o(1)})$, and $O(m^{1+\epsilon})$ for any constant $\epsilon>0$, respectively.

[2] Given a graph G, this problem asks if we can delete k edges to disconnect G into at least p connected components. This problem is often called Min k-Cut. But we choose to preserve k as a parameter in unbreakable decomposition.

Reference	Time	Unbreakability	Adhesion	Depth	Subtree?	Vertex or edge
[CLP+19]	$2^{O(k^2)}n^2m$	$(2^{O(k)}, k)$	$2^{O(k)}$	n	Yes	Vertex
[CKL+20]	$2^{O(k \log k)}n^{O(1)}$	$(i, i)\ \forall i \leq k$	k	n	No	Vertex
[LSS22]	$n^{O(1)}$	$((k+1)^5, k)$	k	n	No	Edge
[ILSS23]	$2^{O(k \log k)}n^{O(1)}$	$(9k, k)$	$8k$	$O(\log n)$	No	Vertex
Thm 6.1	$2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})}m^{1+\epsilon}$	$((2\lceil \frac{1}{\epsilon} \rceil + 3)k, k)$	$(2\lceil \frac{1}{\epsilon} \rceil + 2)k$	$O(\frac{k}{\epsilon} \log n)$	Yes	Vertex
	$2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})}m^{1+\epsilon}$	$(O(k/\epsilon), k)$	$O(k/\epsilon)$	$O(\log n)$	Yes	Vertex

Table 1: Known constructions of unbreakable decomposition. The "subtree?" column indicates whether the decomposition satisfies the *subtree unbreakability* property (see Definition 3.3). The "vertex or edge" column indicates whether each bag is unbreakable or (weaker) edge-unbreakable.

applications include model checking [PSS+22, LRSZ18, SSS+24], connectivity oracles under vertex failures [PSS+22], and FPT algorithms for Multiway Node Hubs [SZ23], Judicious Partitions [LSSZ19], Fair Bisection [ILSS23], and deletion to bounded degree graphs [AKP+22].

Bottleneck. Unfortunately, all known algorithms for constructing unbreakable decompositions share an important drawback: their running time is far from linear in the size of the graph, in contrast to other key algorithmic techniques such as important separators [Mar06], flow augmentation [KKPW24], treewidth reduction [MOR13], and branching from half-integral solutions [IYY18]. The fastest known algorithm in literature still takes at least $\Omega(mn^2)$ time in a graph with n vertices and m edges [CLP+19]. See Table 1. Unbreakable decomposition has thus become a common bottleneck for obtaining close-to-linear time FPT algorithms for many of its applications. Indeed, Cygan et al. [CKL+20] has stated that whether unbreakable decomposition admits a near-linear time construction is "an interesting and challenging open problem".

This paper gives an affirmative answer to this question up to an arbitrarily small polynomial factor. Furthermore, our unbreakability and adhesion parameters are optimal up to constant factors, almost matching the optimal guarantees of [CKL+20].

Theorem 1.1. *For any $0 < \epsilon \leq 1$, there is a randomized algorithm that runs in time $2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})}m^{1+\epsilon}$ and computes with high probability a $(O(\frac{k}{\epsilon}), k)$-unbreakable decomposition with adhesion $O(\frac{k}{\epsilon})$.*

Our decomposition also has $O(\log n)$ depth and satisfies the *subtree unbreakability property* (see Definition 3.3), both of which are not satisfied by the decomposition with the best unbreakability parameters [CKL+20]. Both properties are useful in applications. For example, the performance of the connectivity oracle of [PSS+22] crucially relies on the subtree unbreakability property, and its space depends on the depth of the decomposition. [ILSS23] also requires a low-depth decomposition.

As an almost immediate application of Theorem 1.1, we show the first close-to-linear time FPT algorithm for Minimum p-Way Cut.

Theorem 1.2. *For any $0 < \epsilon < 1$, there is a $2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})}m^{1+\epsilon}$ time algorithm that decides if a graph G has a p-Way Cut with at most k cut edges.*

Prior to our work, [KT11] showed an algorithm with a running time of $O(k^{k^{O(k)}}n^2)$. This was improved by [CCH+16] to $O(2^{O(k^2 \log p)}n^2)$ and by [CKL+20] to $O(2^{O(k \log k)}n^{O(1)})$, where the dependency on $n^{O(1)}$ is at least $\Omega(mn^2)$.

We believe that this is only the first of many such applications. Since there are numerous problems whose only solution so far is using unbreakable decomposition, Theorem 1.1 removes the main obstacle and opens up many exciting open problems listed in Section 8.

2 Technical Overview

Given parameters k and ϵ, we define $\sigma = k + k\left\lceil \frac{1}{\epsilon} \right\rceil$ and $q = k + 2\sigma$. Our strategy is to recursively call the following key subroutine: Given a graph H and a set of *boundary* vertices $B \subseteq V(H)$, satisfying $|B| \leq 2\sigma$, find a superset $X \supseteq B$ such that

1. X is (q, k)-unbreakable,
2. X has adhesion 2σ, i.e., $|N_H(C)| \leq 2\sigma$ for each connected component C in $H \setminus X$,
3. X is $\frac{1}{2}$-balanced, i.e., $|V(C)| \leq |V(H)|/2$ for each connected component C in $H \setminus X$.

Given X, we will then create a node t in the tree decomposition, set the bag $\beta(t) = X$, and recurse on the graphs $C \cup N_H(C)$ with the boundary $N_H(C)$ for every connected component C of $H \setminus X$. Each recursive call will then create a tree node with parent t.

By simply calling this subroutine on G and initial $B = \emptyset$, we would obtain an (q, k)-unbreakable decomposition with adhesion 2σ and finish. The recursion depth is $O(\log n)$ by the $\frac{1}{2}$-balanced condition, so it suffices to show a close-to-linear time algorithm for this key subroutine. In fact, we obtain the key subroutine only when $|B| \leq \sigma$. We will later show how to reduce the case of $|B| > \sigma$ to the case when $|B| \leq \sigma$ at the end of this overview.

Suppose $|B| \leq \sigma$. If we find a vertex set X_1 that is (σ, k)-unbreakable, has adhesion σ, and is $\frac{1}{2}$-balanced, then we can simply return $X := X_1 \cup B$, because X must be $(\sigma + |B|, k)$-unbreakable where $\sigma + |B| \leq q$, has adhesion $\sigma + |B| \leq 2\sigma$, and is $\frac{1}{2}$-balanced.

Our key technical contribution (Theorem 5.2) is a fast algorithm for computing X_1, i.e., an unbreakable low-adhesion balanced set. We do this in two main steps. The first step computes an unbreakable *balanced origin* X_0, defined below. The second step, called the *reducing adhesion* step, will return the desired set $X_1 \supseteq X_0$.

Balanced Origin (Section 4). We say that a vertex set X is a $\frac{1}{2}$-balanced σ-origin if every superset $X' \supseteq X$ with adhesion σ must be $\frac{1}{2}$-balanced. We will compute a $\frac{1}{2}$-balanced σ-origin that is also (k, k)-unbreakable.

We first observe that we can obtain a $\frac{1}{2}$-balanced σ-origin W, with constant probability, simply by sampling a random vertex set of size $O(\sigma)$. This holds because, as shown in [FM06], with constant probability, W is even a $(\frac{1}{2}, \sigma)$-net which is a stronger notion. Let us assume that this event holds with certainty for simplicity.

Now, if W is also (k, k)-unbreakable, then we can return $X_0 \leftarrow W$ and be done. Else, there is a (k, k)-breakable witness (L, R) for W. We can find such a witness simply by trying all partitions (W_L, W_R) of W and checking if there is a cut of size k separating (W_L, W_R). This takes $O(2^{|W|}mk) = 2^{O(k/\epsilon)}m$ time. Assume w.l.o.g. that $|L| \leq |R|$. We update $W \leftarrow (W \setminus L) \cup (L \cap R)$. Our key observation is that W remains a $\frac{1}{2}$-balanced σ-origin (because $|L| \leq |V(H)|/2$) and the size of W strictly decreases. Hence, this update can happen at most $O(\sigma)$ times before W becomes (k, k)-unbreakable. Thus, we obtain a (k, k)-unbreakable $\frac{1}{2}$-balanced σ-origin X_0 in $2^{O(k/\epsilon)}m$ time.

Reducing Adhesion (Section 5). Our *reducing adhesion* subroutine (Lemma 5.1) is such that, given any (k,k)-unbreakable set X_0, it finds a superset of X_0 that is (σ,k)-unbreakable and has adhesion σ.

Given this, we can obtain the desired set X_1 by feeding a (k,k)-unbreakable $\frac{1}{2}$-balanced σ-origin X_0 as the input to the reducing adhesion subroutine. Then, the output X_1 must be (σ,k)-unbreakable and have adhesion σ. Also, X_1 must be $\frac{1}{2}$-balanced, because $X_1 \supseteq X_0$ has adhesion σ and X_0 is a $\frac{1}{2}$-balanced σ-origin.

The high-level scheme of the reducing adhesion subroutine is as follows. We initialize $T \leftarrow V(H)$ and $X_1 \leftarrow X_0$ and keep updating T and X_1, while maintaining the following three invariants:

1. $X_0 \subseteq X_1 \subseteq T$,
2. T has adhesion σ, and
3. X_1 is (σ,k)-unbreakable.

We stop once $X_1 = T$. Thus, X_1 will be (σ,k)-unbreakable and has adhesion σ as desired.

Our approach for updating T and X_1 is centered around an object called a (X_1,T,k')-*witness*, which is a vertex cut (L,R) of size $|L\cap R| \le k'$ where $|L\cap T| > |L\cap R|$ and $X_1 \subseteq R$. This notion is similar to a (k',k')-breakable witness for T, but it requires $X_1 \subseteq R$ and omits $|R\cap T| > |L\cap R|$. Given a (X_1,T,k')-witness (L,R), our algorithm updates T by *carving* T along the witness: we set $T \leftarrow (T\setminus L)\cup(L\cap R)$. Before carving, a vertex $v \in L\setminus R$ is called (X_1,T,k')-*carvable*.[3]

The carving operation maintains $X_1 \subseteq T$ because $X_1 \subseteq R$. Thus, Invariants 1 and 3 are maintained. In fact, X_1 remains (k,k)-unbreakable because X_1 has never been updated. We can also maintain Invariant 2 by making the (X_1,T,k')-witness *lean* (see Definition 5.7). The leanness is easy to ensure using max-flow computation; we omit this detail here. The point of the carving operation is that it removes $|(L\setminus R)\cap T| \ge 1$ carvable vertices from $T\setminus X_1$, progressing towards the goal of $T = X_1$.

Our key structural lemma (Lemma 5.5) says that if X_1 is (q_1,k)-unbreakable for any q_1 and there is no (X_1,T,q_1+k)-carvable vertex in T, then T is (q_1+k,k)-unbreakable. This lemma suggests a natural algorithm: keep finding a $(X_1,T,2k)$-witness and carving T along it until $T = X_1$ or no such witness is left. If the former happens, we are done. Otherwise, if the latter happens, the lemma implies that T is $(2k,k)$-unbreakable, as X_1 is (k,k)-unbreakable. Since T also has adhesion σ by induction, we can set $X_1 \leftarrow T$ and return X_1.

While the above approach is correct, it is too slow. Indeed, a single carving operation might reduce the size of $T\setminus X_1$ by only one. Thus, the process could take as large as $\Omega(n)$ iterations.

Reducing Adhesion Fast. To speed up, our key algorithmic tool is the *disjoint-witness* algorithm (Lemma 5.12) that computes

- A vertex set Q that contains all (X_1,T,k')-carvable vertices in T,
- A collection $\mathcal{C}$ of disjoint (X_1,T,k')-witnesses $\mathcal{C}$ where $(L\setminus R)\cap T \subseteq Q$ for all $(L,R)\in\mathcal{C}$ and $\sum_{(L,R)\in\mathcal{C}} |(L\setminus R)\cap T| = \Omega_{k'}(|Q|/\log n)$.

Intuitively, the witnesses in $\mathcal{C}$ contains $\Omega_{k'}(1/\log n)$-fraction of all carvable vertices in T. The disjoint-witness algorithm takes $2^{O(k'\log k')}m^{1+o(1)}$ time and is based on the color coding technique

[3]The precise definition of carvable vertices (Definition 5.4) requires a technical condition that the witness is "connected" in some sense. We omit this detail here.

in [CKL+20] combined with the single source min-cut threshold algorithm introduced by [LP21], adapted to the vertex version by [PSY22].

For simplicity, we first explain how to compute X_1 in $O_k(m^{1+o(1)}\sqrt{n})$ running time using a 2-level algorithm. Our final algorithm reduces the factor $\sqrt{n}$ to n^ϵ by having $\lceil \frac{1}{\epsilon} \rceil$ levels.

Start by computing the set Q of $(X_1, T, 2k)$-carvable vertices in T and the collection $\mathcal{C}$ of disjoint $(X_1, T, 2k)$-witnesses. There are two cases depending on whether $|Q| \leq \sqrt{n}$ or not. If $|Q| > \sqrt{n}$, then we carve T along all witness of $\mathcal{C}$. Note that the size of $T \setminus X_1$ must decrease by at least $\sum_{(L,R)\in\mathcal{C}} |(L \setminus R) \cap T| = \Omega_k(\sqrt{n}/\log n)$. So, this can happen at most $O_k(\sqrt{n}\log n)$ times. The total running time in this case is at most $O_k(m^{1+o(1)}\sqrt{n})$.

Next, suppose $|Q| \leq \sqrt{n}$. At a first glance, one may expect at most $\sqrt{n}$ further iterations because there are at most $\sqrt{n}$ many $(X_1, T, 2k)$-carvable vertices left. Unfortunately, we cannot find a way to show this (and leave this as an open problem) since each carving operation might introduce new $(X_1, T, 2k)$-carvable vertices. This happens because the operation adds some new vertices into T. Even though $|T \setminus X_1|$ strictly decreases, there might be too many iterations because it is possible that $|T \setminus X_1| = \Omega(n)$ while $|Q| \leq \sqrt{n}$.

Our solution is to use the extension of the key structural lemma (Lemma 5.5) above: for any set Y where $X_1 \subseteq Y \subseteq T$, if X_1 is (q_1, k)-unbreakable and there is no $(X_1, T, q_1 + k)$-carvable vertex in Y, then Y is $(q_1 + k, k)$-unbreakable. Since $T \setminus Q$ has no $(X_1, T, 2k)$-carvable vertex, we can set $X_1 \leftarrow T \setminus Q$ (satisfying Invariant 1). The lemma implies that the new X_1 is $(2k, k)$-unbreakable (satisfying Invariant 3) and $|T \setminus X_1| = |Q| \leq \sqrt{n}$. After this operation, the algorithm proceeds to the second *level*.

In the second level, we keep finding a $(X_1, T, 3k)$-witness and carve T along it, until $T = X_1$ or no such witness is left. At the beginning of this level, $|T \setminus X_1| \leq \sqrt{n}$, and so there are at most $\sqrt{n}$ further iterations. Once there is no $(X_1, T, 3k)$-witness, we return $X_1 \leftarrow T$. We conclude that X_1 is $(3k, k)$-unbreakable by the key structural lemma and X_1 has adhesion σ by induction, as desired.

We can extend the above 2-level algorithm to 3 levels simply by replacing the $\sqrt{n}$ threshold by two thresholds $n^{2/3}$ and $n^{1/3}$. By extending this approach to $\lceil \frac{1}{\epsilon} \rceil$ levels, we can obtain X_1 which is $(\lceil \frac{1}{\epsilon} \rceil k + k = \sigma, k)$-unbreakable and has adhesion σ in time $O_k(m^{1+o(1)}n^\epsilon)$ as desired.

When $|B| > \sigma$. We briefly discuss the case when $\sigma < |B| \leq 2\sigma$. Suppose that there exists a (k, k)-breakable witness (L, R) for B. Let $X = (L \cap R) \cup B$. Since $|X| \leq |B| + k \leq 2\sigma + k = q$, X is trivially (q, k)-unbreakable. Moreover, we can show that the adhesion of X is at most $|B| - 1$. That is, the size of the boundary strictly decreases, making progress towards the case where $|B| \leq \sigma$. So, while we have that X does satisfy Properties 1 and 2 but may not satisfy Property 3, this case may occur at most σ times before $|B| \leq \sigma$. This is why the depth of our decomposition has a factor of σ. In Section 6.1, we further improve the depth to $O(\log n)$.

The next case is when B is (k, k)-unbreakable. If we apply the reducing adhesion subroutine to B, we get a set X_2 which is (σ, k)-unbreakable and has adhesion σ. This means that each connected component in $H \setminus X_2$ has a boundary size at most σ. Therefore, this reduces the problem to the previous case of $|B| \leq \sigma$ in close-to-linear FPT time.

Organization: Section 3 contains basic definitions. Section 4 describes the construction of unbreakable balanced origins. Section 5 describes our fast algorithm for reducing adhesion. Finally, using the tools from these two sections as outlined above, Section 6 shows the construction of the unbreakable decomposition. As an example of applications, Section 7 shows the close-to-linear-time FPT algorithm for p-Way Cut. We list potential applications of our result and more open problems in Section 8.

3 Preliminaries

Throughout this paper, we use n to denote the number of vertices and m to denote the number of edges in a graph. All graphs are undirected, unweighted, and connected unless otherwise stated, and hence we shall assume $m \geq n-1$. Given a graph G and a subset of vertices $U \subseteq V(G)$, $G[U]$ denotes the subgraph induced on the set of vertices U and $E_G(U)$ denotes the set of edges in $E(G)$ whose both endpoints are in U, i.e. the set of edges in $G[U]$. The set of neighbours of a vertex v in G is denoted by $N_G(v)$. We denote by $N_G(U) := \bigcup_{v \in U} N_G(v) \setminus U$ the set of neighbours of U. We omit the subscripts when the graph is clear from the context.

Vertex Cuts. A *vertex cut* (L, R) is such that $L \cup R = V(G)$, $L \setminus R$ and $R \setminus L$ are not empty, and there is no edge between $L \setminus R$ and $R \setminus L$. The *size* of the vertex cut (L, R) is $|L \cap R|$. We emphasize that, we will view (L, R) as an *ordered pair*, because in some definitions (e.g. Definitions 3.4 and 5.7), the order of (L, R) does matter. Throughout Sections 4 to 6 and appendix A, we only consider vertex cuts (not edge cuts), so we usually write cuts as an abbreviation of vertex cuts.

Vertex-Capacitated Graphs and Mincuts. We use $\widetilde{G}$ to denote *capacitated graphs* with positive integral vertex capacity function $\rho : V(\widetilde{G}) \to \mathbb{Z}^+$. For an arbitrary vertex set $A \subseteq V(\widetilde{G})$, we define $\rho(A) = \sum_{v \in A} \rho(v)$. Given two disjoint vertex sets $A, B \in V(\widetilde{G})$ such that there is no edge connecting A and B, an A-B cut is a cut (L, R) with $A \subseteq L \setminus R$ and $B \subseteq R \setminus L$, and an A-B mincut is an A-B cut that minimizes the (capacitated) cut size $\rho(L \cap R)$. We use $\lambda_{\widetilde{G}}(A, B)$ to denote the (capacitated) size of an A-B mincut. To avoid clutter, when $A = \{a\}$ (resp. $B = \{b\}$) is a singleton vertex, we replace $\{a\}$ with a (resp. replace $\{b\}$ with b).

Balance, Adhesion and Unbreakability. For any vertex set $X \subseteq V(G)$, X is *α-balanced* if each connected component C of $G \setminus X$ has size $|C| \leq \alpha n$. The *adhesion* $\sigma_G(X)$ of X in G, or simply the adhesion of X, is the maximum, over connected components C of $G \setminus X$, of the quantity $|N_G(C)|$.

Definition 3.1 (Unbreakablility)**.** *A vertex set $X \subseteq V(G)$ is (q, k)-*unbreakable in G *if every vertex cut (L, R) of size at most k satisfies $|L \cap X| \leq q$ or $|R \cap X| \leq q$. A (q, k)-*breakable witness of X in G *is a vertex cut (L, R) of G of size as most k satisfying $|L \cap X| > q$ and $|R \cap X| > q$.*

By definition, X is (q, k)-unbreakable in G iff there is no (q, k)-breakable witness of X in G. Note that any set X of size at most q is vacuously (q, k)-unbreakable.

The core technical contribution of this paper (Theorem 5.2) is a close-to-linear time algorithm for computing a vertex set that is simultaneously 1/2-balanced, unbreakable, and has low-adhesion.

Unbreakable Decomposition. Next, we define the key object of this paper.

Definition 3.2 (Tree Decomposition)**.** *A* tree decomposition *of a graph G is a pair (T, β), where T is a tree and $\beta : V(T) \to 2^{V(G)}$ is a mapping that assigns to every tree node t a subset $\beta(t) \subseteq V(G)$, called a* bag*. Furthermore, (T, β) satisfies the following.*

- *For each vertex $v \in V(G)$, the set $\{t \mid v \in \beta(t)\}$ induces a connected subtree of T.*
- *For each edge $\{u, v\} \in E(G)$, there is a tree node t where $u, v \in \beta(t)$.*

A *rooted* tree decomposition is a tree decompositon (T, β) together with a designated root node $r \in V(T)$. For any node $t \in V(T)$ with parent t', the adhesion of a tree node t is $\sigma(t) = \beta(t) \cap \beta(t')$. We define $\sigma(r) = \emptyset$ for the root r. The *adhesion* of T is $|\max_{t \in T} \sigma(t)|$. For every $t \in V(T)$, we also define the sets

$$\gamma(t) = \bigcup_{\text{descendants } s \text{ of } t} \beta(s) \text{ and } G_t = G[\gamma(t)] - E_G(\sigma(t)).$$

Definition 3.3 (Unbreakable Decomposition)**.** *A (q, k)-unbreakable decomposition of G is a rooted tree decomposition (T, β) where each bag $\beta(t)$ is (q, k)-unbreakable in G. The decomposition admits the stronger* subtree unbreakability *property if each bag $\beta(t)$ is (q, k)-unbreakable in G_t.*

The main goal of this paper (Section 6) is a fast algorithm for computing an unbreakable decomposition with subtree unbreakability property and small adhesion.

For each tree node t, let $\alpha(t) = \gamma(t) \setminus \sigma(t)$. We say that a rooted tree decomposition (T, β) is *compact* if for every node $t \in V(T)$ for which $\sigma(t) \neq \emptyset$, $G[\alpha(t)]$ is connected and $N_G(\alpha(t)) = \sigma(t)$. This property is handy for performing dynamic programming on tree decomposition, and we will exploit it in Section 7.

Single Source Vertex Mincuts. We will use an algorithm for computing single-source vertex min-cuts. We start by defining the notion of *disjoint* cuts.

Definition 3.4 (Disjoint Cuts)**.** *Let $\mathcal{C}$ be a collection of cuts. The cuts in $\mathcal{C}$ are disjoint if for each pair of different cuts $(L, R), (L', R') \in \mathcal{C}$, $L \setminus R$ and L' are disjoint.*

A cut collection $\mathcal{C}$ is a set of cuts (L, R). Next, we define a *mincut cover*, which is the output of our single source vertex mincuts subroutine.

Definition 3.5 (Mincut Covers)**.** *Consider a capacitated graph $\widetilde{G}$ with a source vertex s and sink vertices T such that $\{s\} \cup T$ is an independent set. A* mincut cover *$\mathcal{K}$ with respect to s and T in $\widetilde{G}$ is a set of cut collections $\mathcal{C}$, which satisfies the following.*

1. *For each collection $\mathcal{C} \in \mathcal{K}$ and cut $(L, R) \in \mathcal{C}$, (L, R) is a t-s mincut for some sink $t \in T$.*
2. *For each sink $t \in T$, there exists a cut (L, R) in some collection $\mathcal{C} \in \mathcal{K}$ such that $t \in L \setminus R$.*
3. *Each collection $\mathcal{C}$ is a set of disjoint cuts.*

The width *of a mincut cover $\mathcal{K}$ is the number of collections $\mathcal{C}$ in $\mathcal{K}$. To avoid clutter, we also use $(L, R) \in \mathcal{K}$ to denote a cut $(L, R) \in \mathcal{C}$ for some collection $\mathcal{C} \in \mathcal{K}$.*

Roughly speaking, the single source vertex mincuts subroutine receives a capacitated graph $\widetilde{G}$ with one source s, a set T of several sinks, and a parameter k, and outputs a small-width mincut cover $\mathcal{K}$ with respect to s and T. In other words, we can obtain t-s mincuts for all $t \in T$, and these mincuts can be partitioned into a small number of collections of disjoint cuts. We defer the proof of this result to Appendix A.

Theorem 3.6. *Consider an m-edge capacitated graph $\widetilde{G}$ with vertex capacity function ρ, a parameter k, a single source vertex s and sink vertices T satisfying that $\{s\} \cup T$ is an independent set and each source/sink vertex has capacity ∞. Let T^* be the set of sink vertices t with $\lambda_{\widetilde{G}}(t, s) \leq k$. There is a randomized algorithm that, with high probability, computes a mincut cover $\mathcal{K}$ with respect to s and T^* which has width $O(k \log^3 n)$. The running time is $O(km^{1+o(1)})$.*

4 Unbreakable Balanced Origins from Nets

Let us define a *balanced origin*, a vertex set where every low-adhesion superset must be balanced.

Definition 4.1 (Balanced Origin)**.** *Given a graph G, an α-balanced σ-origin is a vertex set $X \subseteq V(G)$ such that, for any vertex set $X' \supseteq X$ with adhesion σ, X' must be α-balanced.*

The goal of this section is to compute an unbreakable balanced origin.

Lemma 4.2. *Given an m-edge graph G with parameters k, σ where $k \leq \sigma$, there is a randomized algorithm that computes a set X which is always (k,k)-unbreakable such that with constant probability, X is a $\frac{1}{2}$-balanced σ-origin. The running time is $2^{O(\sigma)}m$.*

The high-level idea of the algorithm for Lemma 4.2 is to first sample a *net* (Lemma 4.4) as the initial set and keep updating the set as long as there exists a breakable witness until it becomes unbreakable in a straightforward manner using Lemma 4.5. The properties of nets will then allow us to show that the final set is a $\frac{1}{2}$-balanced σ-origin.

We now formally define the notion of (α, σ)-nets. This concept was introduced by [FM06] and is closely related to the notion of *detection sets* [Kle04].

Definition 4.3 ((α,σ)-nets)**.** *A set W of vertices in a graph G is an (α,σ)-net if for every set of vertices S in G of size at most σ, and for every connected component D of $G \setminus S$,*

1. *If $|D| \geq \alpha n$, then D has at least one vertex from W.*
2. *If $|D| \leq (1-\alpha)n - |S|$, then the set $V(G) \setminus (D \cup S)$ has at least one vertex from W.*

We remark that we will only use the first property in our proof of Lemma 4.2 and in the whole paper.

Lemma 4.4 (Corollary 3.6 of [FM06])**.** *Given a graph G, there exists an absolute constant c such that any random subset $W \subseteq V(G)$ of size $c\frac{\sigma}{\alpha}\log\frac{1}{\alpha}$ is an (α,σ)-net with constant probability.*

Lemma 4.5. *Given an m-edge graph G with a set $W \subseteq V(G)$ and parameters q, k, there is a deterministic algorithm that either*

- *certifies that W is (q,k)-unbreakable in G, or*
- *outputs a (q,k)-breakable witness (L,R) of W in G.*

The running time is $O(2^{|W|}km)$.

Proof. For each $W_L \subseteq W$ and $W_R = W \setminus W_L$ s.t. $|W_L|, |W_R| > q$, we compute a vertex mincut (L,R) that separates W_L from W_R in G. If the cut size is at most k, then (L,R) is a (q,k)-breakable witness of W in G by definition, and we terminate the algorithm with output (L,R). If there is no such (L,R) after checking all (W_L, W_R), it must be the case that there is no (q,k)-breakable witness of W in G, so W is (q,k)-unbreakable in G by definition.

Now we analyze the running time. The number of partitions (W_L, W_R) of W is at most $2^{|W|}$. For each of them, we invoke a maxflow algorithm[4] on G, which takes $O(km)$ time. Hence the total running time is $O(2^{|W|}km)$. □

[4]We can use the classic Ford-Fulkerson algorithm because we only want to obtain a mincut of size at most k or decide such mincut does not exist.

We are now ready to prove Lemma 4.2.

Proof of Lemma 4.2. Set $\alpha = \frac{1}{2}$. Using Lemma 4.4, we first compute an (α, σ)-net W of G with size $|W| = O(\frac{\sigma}{\alpha}\log(\frac{1}{\alpha})) = O(\sigma)$. We initialize $X^{(0)}$ to be W, and then we will update this set iteratively as follows. Let $X^{(i)}$ be the set of vertices right after the i-th iteration. For all $i \geq 0$, we will maintain the invariant that for each vertex $v \in W \setminus X^{(i)}$, the connected component in $G \setminus X^{(i)}$ containing v has size at most $n/2$. Initially, the invariant vacuously holds because $X^{(0)} = W$ and so $W \setminus X^{(0)} = \emptyset$.

At the i-th iteration, we use Lemma 4.5 to check whether $X^{(i-1)}$ is (k,k)-unbreakable in G or not. If it is indeed (k,k) unbreakable, we set $X = X^{(i-1)}$ and terminate the whole algorithm with output X. Otherwise, Lemma 4.5 will return a (k,k)-breakable witness (L,R) of $X^{(i-1)}$ in G. Without loss of generality, we assume L is the smaller side, i.e. $|L| \leq |R|$. Then we update $X^{(i)} = (X^{(i-1)} \setminus L) \cup (L \cap R)$. Namely, we first remove the part in the smaller side L, and then add $L \cap R$.

Now, we show that the invariant holds for $X^{(i)}$, assuming that $X^{(i-1)}$ already satisfies the invariant. Consider a vertex $v \in W \setminus X^{(i)}$. First, if $v \in L \setminus R$, the connected component D_i of $G \setminus X^{(i)}$ containing v satisfies $D_i \subseteq L \setminus R$, so $|D_i| \leq |L \setminus R| \leq n/2$. From now, we consider the case that $v \in R$. In fact, we must have $v \in R \setminus L$ in this case because $L \cap R \subseteq X^{(i)}$ but $v \in W \setminus X^{(i)}$. Let D_i be the connected component of $G \setminus X^{(i)}$ containing v. Note that $D_i \subseteq R \setminus L$ because $v \in R \setminus L$ and $L \cap R \subseteq X^{(i)}$. Furthermore, by the update rule, $X^{(i-1)} \cap (R \setminus L) = X^{(i)} \cap (R \setminus L)$, so D_i is disjoint from $X^{(i-1)}$, which means D_i is inside a connected component D_{i-1} of $G \setminus X^{(i-1)}$ and in particular $|D_i| \leq |D_{i-1}|$. Again by $X^{(i-1)} \cap (R \setminus L) = X^{(i)} \cap (R \setminus L)$, we know $v \in W \setminus X^{(i-1)}$ combining $v \in R \setminus L$ and $v \in W \setminus X^{(i)}$. Therefore, the invariant of $X^{(i-1)}$ gives that $|D_{i-1}| \leq n/2$, which implies $|D_i| \leq |D_{i-1}| \leq n/2$ as desired.

At the end of the algorithm (we will discuss why it must end in the running time analysis), we obtain a (k,k)-unbreakable set X such that for each vertex $v \in W \setminus X$, the connected component in $G \setminus X$ containing v has size at most $n/2$. Note that the unbreakability of X and the invariant always hold regardless of the success probability of Lemma 4.4.

Next we show that X is a $\frac{1}{2}$-balanced σ-origin conditioned on the success of Lemma 4.4. Since Lemma 4.4 is satisfied with constant probability, it would then follow that X is a $\frac{1}{2}$-balanced σ-origin with constant probability. Let $X' \supseteq X$ be a vertex set with adhesion σ. Consider a connected component D' of $G \setminus X'$, and assume for contradiction that $|D'| > n/2$. We know that $|N_G(D')| \leq \sigma$ since the adhesion of X', $\sigma(X')$, is at most σ. Then since $|D'| > n/2$, $|N_G(D')| \leq \sigma$ and by the definition of (α, σ)-nets (property 1 of Definition 4.3), W has at least one vertex from D', say v. Then $v \in W \setminus X$. However, by the invariant, the connected component D of $G \setminus X$ containing v must have size $|D| \leq n/2$, which implies that $|D'| \leq |D| \leq n/2$, a contradiction.

Running Time. The bottleneck is applying Lemma 4.5 in each iteration. The number of iterations is at most $|W|$ since the initial set is $X^{(0)} = W$ and each update to the set decreases its size by at least 1. Therefore, the total running time is $O(2^{|W|} \cdot |W| \cdot km) = 2^{O(\sigma)}m$ (since $\sigma \geq k$). □

5 Reducing Adhesion of Unbreakable Sets

This subsection forms the main technical component of our unbreakable decomposition. Roughly speaking, Lemma 5.1 shows, given an unbreakable set X_0, in almost linear time, we can expand it to another set X which is appropriately unbreakable and has small adhesion.

Lemma 5.1. *Given an m-edge graph G with parameters $0 < \epsilon \leq 1$, $k \geq 1$ and $q \geq k$, and an initial set $X_0 \subseteq V(G)$ such that X_0 is (q,k)-unbreakable, there is an algorithm that computes with*

high probability that a set $X \supseteq X_0$ such that X is $(q + k\lceil\frac{1}{\epsilon}\rceil, k)$ unbreakable and has adhesion at most $\sigma(X) = q + k\lceil\frac{1}{\epsilon}\rceil$. The running time is $\exp(O((q + \frac{k}{\epsilon})\log(q + \frac{k}{\epsilon})))m^{1+\epsilon+o(1)}$.

Before proving Lemma 5.1, we first give Theorem 5.2 which may be of independent interest. Roughly speaking, Theorem 5.2 is a simple corollary of Lemma 4.2 and Lemma 5.1, which says we can compute a unbreakable balanced vertex set with low adhesion efficiently.

Theorem 5.2 (Unbreakable Balanced Low-Adhesion Sets)**.** *Given an m-edge graph G with parameters $0 < \epsilon \leq 1$ and $k \geq 1$, there is a randomized algorithm that with high probability computes a set $X \subseteq V(G)$ such that X is $(\lceil 1/\epsilon\rceil k + k, k)$-unbreakable, 1/2-balanced and has adhesion $\lceil 1/\epsilon\rceil k + k$. The running time is* $\exp(O(\frac{k}{\epsilon}\log\frac{k}{\epsilon}))m^{1+\epsilon+o(1)}$.

Proof. We do the following $O(\log n)$ times. First, let X_0 be an initial set by applying Lemma 4.2 on H with parameters k and $\sigma = \lceil 1/\epsilon\rceil k + k$. Second, compute X by applying Lemma 5.1 on graph H with X_0 as the initial set and parameters $\epsilon, k, q = k$. If X is 1/2-balanced, we terminate the whole algorithm, otherwise proceed to the next iteration.

To see the correctness, consider each of the $O(\log n)$ iterations. Lemma 4.2 guarantees that X_0 is always (k, k)-unbreakable and with constant probability, X_0 is a 1/2-balanced σ-origin. Therefore, Lemma 5.1 guarantees that, with high probability, $X \supseteq X_0$ is $(\lceil 1/\epsilon\rceil k + k)$-unbreakable and has adhesion at most $\sigma = \lceil 1/\epsilon\rceil k + k$. Furthermore, because X_0 is a 1/2-balanced σ-origin with constant probability, we have X is 1/2-balanced with constant probability. It follows that with high probability, at least one of the $O(\log n)$ iterations will give a 1/2-balanced X. □

In the following subsections, we will prove Lemma 5.1.

5.1 Witnesses and Carvable Vertices

We will introduce concepts around *witnesses* and *carvable vertices*, and then show several useful observations that eventually lead to the final algorithm for Lemma 5.1.

Definition 5.3. *Given a vertex set $T \subseteq V(G)$, the* torso H_T *of T in G is a graph with $V(H_T) = T$ and $E(H_T) = \{\{u, v\} \mid \{u, v\} \in G$ or $u, v \in N(D)$ for some connected component D of $G \setminus T\}$.*

Observe that H_T has at most $|E(G)|(\sigma(T))^2$ edges where $\sigma(T)$ is the adhesion of T. We will refer to H_T as H when T is clear from the context. This notion of the torso graph H_T was introduced by [CKL+20] for their color coding step, and we will exploit it for our color coding step as well.

Definition 5.4 (Witnesses and Carvable Vertices)**.** *Given a vertex set T and a set of vertices $X \subseteq T$, we say that a cut (L, R) in G is an* (X, T, k')-witness *if*

1. $|L \cap R| \leq k'$,
2. $|L \cap T| > |L \cap R|$, *and*
3. $X \subseteq R$.

We say that a (X, T, k')-witness (L, R) is connected *if $H_T[(L \setminus R) \cap T]$ is connected. A vertex v is* (X, T, k')-carvable *if there exists a connected (X, T, k')-witness where $v \in L \setminus R$.*

This definition of carvable vertices is similar to that in [KPS24]. The two differences are that instead of the condition $|L \cap T| > |L \cap R|$, [KPS24] require $|(L \setminus R) \cap T| > \alpha$ for some $\alpha \gg k'$, and they require that $L \setminus R$ is connected in G, instead of requiring that $(L \setminus R) \cap T$ is connected in $H_T[(L \setminus R) \cap T]$.

The following structural lemma is crucial. It says that, for any vertex set T, if $X \subseteq T$ is unbreakable and we remove all carvable vertices from T to obtain the set Y, then Y is also unbreakable.

Lemma 5.5. *For some arbitrary q, k where $q \geq k$ and a set Y such that $X \subseteq Y \subseteq T$, if X is (q,k)-unbreakable and there is no $(X, T, q+k)$-carvable vertex in Y, then Y is $(q+k, k)$-unbreakable.*

Proof. Suppose for contradiction that Y is not $(q+k, k)$ unbreakable. Then, there exists a vertex cut (L_0, R_0) of size $|L_0 \cap R_0| \leq k$ where $|L_0 \cap Y| > q + k$ and $|R_0 \cap Y| > q + k$. Since X is (q,k)-unbreakable, either $|L_0 \cap X| \leq q$ or $|R_0 \cap X| \leq q$. Without loss of generality, let us assume the former.

Observe that the vertex cut $(L_0, R_1 = R_0 \cup (L_0 \cap X))$ is a $(X, Y, q+k)$-witness. Let us verify the three conditions. First, the cut size is $|L_0 \cap R_1| = |(L_0 \cap R_0) \cup (L_0 \cap X)| \leq q + k$. Second, we have $|L_0 \cap Y| > q + k \geq |L_0 \cap R_1|$. Last, we have $X \subseteq R_0 \cup (L_0 \cap X) \subseteq R_1$ by construction.

Next, we introduce a new notion within this proof. A cut (L, R) is a *candidate-witness* if

1. $|L \cap R| \leq q + k$,
2. $|(L \setminus R) \cap Y| > |(L \cap R) \setminus T|$, and
3. $X \subseteq R$.

The notion of candidate-witness is "between" $(X, Y, q+k)$-witness and $(X, T, q+k)$-witness. More formally, any $(X, Y, q+k)$-witness is a candidate-witness, because for any cut (L, R) we have $|L \cap Y| > |L \cap R|$ iff $|(L \setminus R) \cap Y| > |(L \cap R) \setminus Y|$, and additionally $|(L \cap R) \setminus Y| \geq |(L \cap R) \setminus T|$ because $Y \subseteq T$. Similarly, any candidate-witness is a $(X, T, q+k)$-witness, because $|(L \setminus R) \cap T| \geq |(L \setminus R) \cap Y| > |(L \cap R) \setminus T|$ which is equivalent to $|L \cap T| > |L \cap R|$.

Next, let (L^*, R^*) be a candidate-witness where $|(L^* \setminus R^*) \cap T|$ is minimized. Note that (L^*, R^*) is well-defined by the existence of the $(X, Y, q+k)$-witness (L_0, R_1).

Claim 5.6. *$H_T[(L^* \setminus R^*) \cap T]$ is connected.*

Given this claim, we obtain a contradiction as follows. As observed above, (L^*, R^*) is a $(X, T, q+k)$-witness, and by Claim 5.6, it is a connected $(X, T, q+k)$-witness. Since $|(L^* \setminus R^*) \cap Y| > 0$, there exists a vertex $v \in (L^* \setminus R^*) \cap Y$. So, v is a $(X, T, q+k)$-carvable vertex in Y, which is a contradiction. It remains to prove Claim 5.6. The proof of this claim is similar to the proof of Claim 3.8 in [CKL+20].

Proof of Claim 5.6. Let $\mathcal{C} = \{C_1, C_2, ..., C_q\}$ denote the connected components of $G[L^* \setminus R^*]$. We first assume without loss of generality that each component C_i intersects T. Otherwise, let Γ denote the union of all components C_i that are disjoint from T. Note that Γ is also disjoint from $Y \subseteq T$. Observe that the cut (L', R') with $L' = L^* \setminus \Gamma$ and $R' = R^* \cup \Gamma$ is still a $(X, Y, q+k)$-witness where $|(L' \setminus R') \cap T| = |(L^* \setminus R^*) \cap T|$ is still minimum. So we can work with (L', R') instead.

Assume $H_T[(L^* \setminus R^*) \cap T]$ is not connected. We will reach contradiction by showing another candidate witness $(\hat{L}, \hat{R})$ where $|(\hat{L} \setminus \hat{R}) \cap T| < |(L^* \setminus R^*) \cap T|$.

Let D_1, D_2 be a partition of $(L^* \setminus R^*) \cap T$ such that there is no edge connecting D_1 and D_2 in $H_T[(L^* \setminus R^*) \cap T]$. Note that each $C_i \cap T$ induces a connected subgraph of H_T, so we can write D_1 and D_2 in the form $D_1 = \bigcup_{C_i \in \mathcal{C}_1} C_i \cap T$ and $D_2 = \bigcup_{C_i \in \mathcal{C}_2} C_i \cap T$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ partition $\mathcal{C} = \{C_1, C_2, ..., C_q\}$.

The key observation is that, for any $C_{i_1} \in \mathcal{C}_1$ and $C_{i_2} \in \mathcal{C}_2$, we must have $N(C_{i_1}) \cap N(C_{i_2}) \subseteq T$[5]. Otherwise, by the definition of H_T, the existence of a vertex in $(N(C_{i_1}) \cap N(C_{i_2})) \setminus T$ implies that

[5]We emphasize that $N(\cdot)$ denote neighbors in G.

there is an edge in $E(H_T)$ connecting some terminal $t_1 \in C_{i_1} \cap T \subseteq D_1$ and some $t_2 \in C_{i_2} \cap T \subseteq D_2$, contradicting the fact that there is no edge between D_1 and D_2 in $H_T[(L^* \setminus R^*) \cap T]$.

Let $\widetilde{C}_1 = \bigcup_{C_i \in \mathcal{C}_1} C_i$ and $\widetilde{C}_2 = \bigcup_{C_i \in \mathcal{C}_2} C_i$. From this key observation, we further have $N(\widetilde{C}_1) \cap N(\widetilde{C}_2) \subseteq T$, which implies

$$(N(\widetilde{C}_1) \setminus T) \cap (N(\widetilde{C}_2) \setminus T) = \emptyset.$$

Since $N(\widetilde{C}_1) \cup N(\widetilde{C}_2) = N(L^* \setminus R^*) \subseteq L^* \cap R^*$ (since $\widetilde{C}_1 \cup \widetilde{C}_2 = L^* \setminus R^*$), we also have

$$(N(\widetilde{C}_1) \setminus T) \cup (N(\widetilde{C}_2) \setminus T) \subseteq (L^* \cap R^*) \setminus T.$$

Therefore, we have $|N(\widetilde{C}_1) \setminus T| + |N(\widetilde{C}_2) \setminus T| \leq |(L^* \cap R^*) \setminus T|$. Also, $|\widetilde{C}_1 \cap Y| + |\widetilde{C}_2 \cap Y| = |(L^* \setminus R^*) \cap Y| > |(L^* \cap R^*) \setminus T|$. Combining both, we have either $|\widetilde{C}_1 \cap Y| > |N(\widetilde{C}_1) \setminus T|$ or $|\widetilde{C}_2 \cap Y| > |N(\widetilde{C}_2) \setminus T|$. Thus either one of the cuts $(\widetilde{C}_1 \cup N(\widetilde{C}_1), V(G) \setminus \widetilde{C}_1)$ or $(\widetilde{C}_2 \cup N(\widetilde{C}_2), V(G) \setminus \widetilde{C}_2)$ is a candidate witness. However, since both $\widetilde{C}_1$ and $\widetilde{C}_2$ intersect T, we have $|\widetilde{C}_1 \cap T|, |\widetilde{C}_2 \cap T| < |(L^* \setminus R^*) \cap T|$, which contradicts the minimality of (L^*, R^*). □

□

5.2 Carve Terminals with Lean Witnesses

In this section, we analyze the basic building block of our algorithm for Lemma 5.1. We define the notion of *lean witness*, describe how to carve a terminal set given a lean witness, and analyze how the adhesion and size of a terminal set changes. The notion of *lean witness* in this paper essentially serves the same purpose as the term single bag lean witness introduced by [CKL+20].

Definition 5.7. *An (X, T, k')-witness (L^*, R^*) is* lean *if there exists a set of $|L^* \cap R^*|$ vertex disjoint paths in $G[L^*]$ starting from every vertex in $L^* \cap R^*$ and ending at $L^* \cap T$.*

In this subsection, we will consider fixed X and T satisfying $X \subseteq T$. Therefore, we will use lean witnesses as an abbreviation of lean (X, T, k')-witnesses with arbitrary k'.

In Definition 5.8, we formally define the carve operation that carves the terminal set T with a lean witness. Next, Lemma 5.9 shows that, roughly speaking, when we carve T with a lean witness, the adhesion of T will not increase and the size of T will decrease.

Definition 5.8 (Carve with One Lean Witness)**.** *Given a terminal set T, we define the* carve *operation with respect to a lean witness (L^*, R^*) to return the set $T' = (T \setminus L^*) \cup (L^* \cap R^*)$.*

Lemma 5.9. *Suppose we carve T with a lean-witness (L^*, R^*). Then the adhesion of the new terminal set $T' = (T \setminus L^*) \cup (L^* \cap R^*)$ is at most $\max(|L^* \cap R^*|, \sigma(T))$ and $|T| - |T'| \geq \frac{|L^* \cap T|}{2 \max\{1, |L^* \cap R^*|\}}$.*

Proof. Consider a component C which is obtained after deleting T' from G, and let us bound $|N(C)|$. There are two cases. In the first case, the component C is a *new component*, i.e. $C \subseteq L^* \setminus R^*$. In this case, clearly $|N(C)| \leq |L^* \cap R^*|$. In the second case, $C \subseteq R^* \setminus L^*$. Observe that in this case, $C \subseteq D$, where D was a connected component of $G \setminus T$. Partition the boundary $N(C)$ into $N(C) \setminus N(D)$ and $N(C) \cap N(D)$.

First we claim that $N(C) \setminus N(D) \subseteq (L^* \cap R^*) \setminus T$, because $N(C) \setminus N(D) \subseteq L^* \cap R^*$ by the carving rule, and any T-vertex in $N(C)$ must belong to $N(D)$. Next, by Definition 5.7, in $G[L^*]$, there exists $|L^* \cap R^*|$ vertex disjoint paths from $L^* \cap R^*$ to $L^* \cap T$. This means in $G[L^* \setminus (L^* \cap R^* \cap T)]$, there exists vertex-disjoint paths starting from all vertices in $(L^* \cap R^*) \setminus T$ to some vertices in $(L^* \setminus R^*) \cap T$. Consider the vertex-disjoint paths starting from $N(C) \setminus N(D)$. We know each of them must go through $N(D) \setminus N(C)$ by the following reasons.

- Such paths end at T-vertices, so they must go through $N(D)$.
- They cannot go through $N(D) \cap N(C)$ because $C \subseteq R^* \setminus L^*$ and $N(C) \cap N(D) \subseteq R^* \cap T$ which is disjoint from $L^* \setminus (L^* \cap R^* \cap T)$.

Therefore, it means that $|N(C) \setminus N(D)| \leq |N(D) \setminus N(C)|$. This in turn yields

$$\begin{aligned} |N(C)| &= |N(C) \setminus N(D)| + |N(C) \cap N(D)| \\ &\leq |N(D) \setminus N(C)| + |N(C) \cap N(D)| \\ &= |N(D)| \leq \sigma(T). \end{aligned}$$

Lastly, observe that the reduction in the size of the terminal set $|T| - |T'|$ at least $|L^* \cap T| - |L^* \cap R^*|$. Let $a = |L^* \cap T|$ and $b = |L^* \cap R^*|$. If $a \geq 2b$, then $a - b \geq \frac{a}{2}$. Otherwise, because $(L^\star, R^\star)$ is a witness, $a - b \geq 1 \geq \frac{a}{2b}$. Therefore, $|T| - |T'| \geq \frac{|L^* \cap T|}{2\max\{1, |L^* \cap R^*|\}}$. This completes the proof. □

We need a slight extension of the carve operation, as instead of updating the terminal set with one lean-witness, we will carve it using a collection of disjoint lean witnesses at once (see Definition 3.4 to recall the definition of disjoint cuts).

Definition 5.10 (Carve with Disjoint Lean Witnesses)**.** *Given a terminal set T and a collection of disjoint lean-witnesses (L_i, R_i), $i \in [z]$, the carve operation with respect to this collection is defined as replacing T by $T \cup (\bigcup_{i\in[z]} L_i \cap R_i) \setminus (\bigcup_{i\in[z]} (L_i \setminus R_i) \cap T)$.*

Lemma 5.11. *Suppose we carve the terminal set T with a set of disjoint lean witnesses $\{(L_i, R_i)\}$, $i \in [z]$. Then the adhesion of the new terminal set T' is at most $\max\{\max_{i\in[z]} |L_i^* \cap R_i^*|, \sigma(T)\}$ and the size of T reduces by at least $\sum_{i\in[z]} \frac{|L_i^* \setminus R_i^*|}{2\max\{1, |L_i^* \cap R_i^*|\}}$*

5.3 Covering Carvable Vertices with Disjoint Witnesses

Lemma 5.12 is the key lemma we prove in this subsection. Broadly, it shows that one can (a) efficiently find the set of all carvable vertices, and (b) find a set of disjoint lean witnesses that *cover* a large fraction of carvable vertices (a vertex v is covered by cut (L, R) if $v \in L \setminus R$).

Lemma 5.12. *Given an m-vertex graph G, a terminal set T, a set $X \subseteq T$ and a parameter k' satisfying $\sigma(T) \leq k'$, there is an algorithm running in time $2^{O(k' \log k')} m^{1+o(1)}$ that computes*

- *a set $Q \subseteq T \setminus X$ that includes all (X, T, k')-carvable vertices in T, and*
- *a collection of disjoint lean (X, T, k')-witnesses $\mathcal{C}$ s.t. $(L \setminus R) \cap T \subseteq Q$ for each $(L, R) \in \mathcal{C}$, and*

$$\sum_{(L,R)\in\mathcal{C}} |(L \setminus R) \cap T| \geq |Q|/\gamma,$$

where $\gamma = 2^{O(k' \log k')} \log n$.

In fact, Lemma 5.12 is a simple corollary of the below Lemma 5.13 and Lemma 5.18, where the former will compute a collection disjoint witnesses (may not be lean) to cover a large fraction of carvable vertices, and the latter will convert disjoint witnesses to disjoint lean witnesses.

Lemma 5.13. *Given an m-vertex graph G, a terminal set T, a set $X \subseteq T$ and a parameter k' satisfying $\sigma(T) \leq k'$, there is an algorithm running in time $2^{O(k' \log k')} m^{1+o(1)}$ that computes*

- *a set $Q \subseteq T \setminus X$ that includes all (X, T, k')-carvable vertices in T, and*
- *a collection of disjoint (X, T, k')-witnesses $\mathcal{C}$ s.t. $(L \setminus R) \cap T \subseteq Q$ for each $(L, R) \in \mathcal{C}$, and*

$$\sum_{(L,R)\in\mathcal{C}} |(L \setminus R) \cap T| \geq |Q|/\gamma_0,$$

where $\gamma_0 = 2^{O(k' \log k')} \log n$.

Most of this subsection is devoted to prove Lemma 5.13. Before we start the proof, we explain on a high level how the algorithm works - for ease of understanding, we focus only on obtaining the set Q. Let $v \in T$ be an (X, T, k')-carvable vertex. Then, we know that there is a connected (X, T, k') witness (L, R) with $v \in L \setminus R$.

For simplicity, assume that $|(L \setminus R) \cap T| = k' + 1$ and $|(L \cap R) \cap T| = k'$. In Lemma 5.15 we show that there is a set Y of at most $O(k'^3)$ vertices, such that for every edge $(u, t) \in E(H)$ where $u \in (L \setminus R)$ and $t \in (R \setminus L)$, we have $t \in Y$. Here $H = H_T$ is the torso of T in G. Roughly speaking, if we use color coding such that every terminal in $(L \setminus R) \cap T$ is colored red, and every terminal in Y is colored blue, then when we contract all the red vertices together, we can detect the cut (L, R) by computing a min-cut between the contracted red vertices and the set X, which can be done using our single source min-cut subroutine, Theorem 3.6. The actual algorithm is slightly more complicated to account for the general case when $|(L \setminus R) \cap T| < k' + 1$ - in this case, we need to account for the terminals in $L \cap R \cap T$ as well. In this case we use color coding with 3 colors, and modify the graph slightly before applying our result on single source min-cuts.

Proof. The proof combines the color coding technique in [CKL+20] along with our single source mincuts subroutine, Theorem 3.6. We remark that Theorem 6.1 of [KPS24] gives a similar result for a different notion of carvable vertices: they give a deterministic algorithm, but our randomized algorithm based on Theorem 3.6 has a better dependence on k'. In this proof, we use H to denote the torso H_T of T in G.

Fix the sets of terminals to be colored. For the sake of analysis, for a vertex $v \in T$ which is (X, T, k')-carvable, we fix a *connected* (X, T, k')-witness (L_v^*, R_v^*) with $v \in L_v^* \setminus R_v^*$ (recall the definition of connected witness in Definition 5.4) satisfying $N_H((L_v^* \setminus R_v^*) \cap T) \supseteq L_v^* \cap R_v^* \cap T$. Such a witness must exist by Observation 5.14 (we defer its proof to Appendix B).

Observation 5.14. *If there exists a connected (X, T, k') witness (L, R), there exists an connected (X, T, k') witness (L', R') with $(L' \setminus R') \cap T = (L \setminus R) \cap T$ further satisfying that $N_H((L' \setminus R') \cap T) \supseteq L' \cap R' \cap T$.*

Because (L_v^*, R_v^*) is a connected (X, T, k')-witness, we know that $H[(L_v^* \setminus R_v^*) \cap T]$ is connected. We fix three vertex sets Z_v, Y_v, W_v w.r.t. v as follows.

- Let Z_v be an arbitrary subset of $(L_v^* \setminus R_v^*) \cap T$ of size $\min\{k' + 1, |(L_v^* \setminus R_v^*) \cap T|\}$ s.t. $H[Z_v]$ is connected (i.e. when $|(L_v^* \setminus R_v^*) \cap T| \leq k' + 1$, we take the entire $(L_v^* \setminus R_v^*) \cap T$ as Z_v).
- Let $W_v = L_v^* \cap R_v^* \cap T$ be the terminals inside $L_v^* \cap R_v^*$.
- Let $Y_v = N_H((L_v^* \setminus R_v^*) \cap T) \setminus W_v$ be the H-neighbors of $(L_v^* \setminus R_v^*) \cap T$ not falling in $L_v^* \cap R_v^*$.

Note that by definition, $|Z_v| \leq k'+1$, $|W_v| \leq k'$, and by Lemma 5.15, $|Y_v| \leq |Z_v|(k'^2) \leq (k'+1)k'^2 \leq 2k'^3$.

Lemma 5.15. *Let (L^*, R^*) be an (X, T, k')-witness. For every vertex $u \in (L^* \setminus R^*) \cap T$ there are at most k'^2 vertices $t \in (R^* \setminus L^*) \cap T$ such that $(u, t) \in E(H)$.*

Proof. Consider a vertex $t \in (R^* \setminus L^*) \cap T$ s.t. $(u, t) \in E(H)$. By the definition of H, it must be the case that there exists a path in G from u to t with no internal vertex from T. Also since $u \in L^* \setminus R^*$ and $t \in R^* \setminus L^*$, this path must go through some vertex $x \in (L^* \cap R^*) \setminus T$. Let C_x be the connected component of $G \setminus T$ containing x. Observe that $t \in N(C_x)$.

The number of possible t is at most k'^2 because there are $|(L^* \cap R^*) \setminus T| \leq k'$ possible choices of x and, for each x, $|N(C_x)| \leq \sigma(T) \leq k'$ by the assumption in Lemma 5.13. □

Color coding. We are now ready to apply the color coding step. The following result, also used in [CKL+20], is essentially a deterministic fast algorithm for color coding.

Lemma 5.16 (Lemma 2.2 of [CKL+20], extension of [CCH+16])**.** *Given a set U of size n and integers $0 \leq a_1 \leq a_2 \leq a_3 \leq a_\ell \leq n$, one can compute a family $\mathcal{F}$ of functions $f : U \to [\ell]$ of size $2^{O((a_1+a_2+\ldots a_{l-1})\log(a_1+a_2+\ldots a_\ell))}O(\log^2 n)$ in time $2^{O((a_1+a_2+\ldots a_{\ell-1})\log(a_1+a_2+\ldots a_\ell))}O(\log^2 n)$ such that for any pairwise disjoint sets $A_1, A_2, A_3 \ldots A_\ell$ of size at most $a_1, a_2, a_3 \ldots a_\ell$ respectively, there exists a function $f \in \mathcal{F}$ such that $f(x_i) = i$ for every $x_i \in A_i$, $i \in [\ell]$.*

We apply Lemma 5.16 on the universe $U = T$ with $a_1 = k' + 1$, $a_2 = k'$ and $a_3 = 2(k')^3$ to obtain a function family $\mathcal{F}$. For a function $f \in \mathcal{F}$, we use $f^{-1}(1), f^{-1}(2), f^{-1}(3) \subseteq T$ to denote the terminals with color 1, 2 and 3 respectively. Furthermore, for a function $f \in \mathcal{F}$ and a terminal $v \in T$, we say that v is *lucky* w.r.t f if $Z_v \subseteq f^{-1}(1)$, $W_v \subseteq f^{-1}(2)$, and $Y_v \subseteq f^{-1}(3)$. Note that by the guarantee of Lemma 5.16, for every (X, T, k')-carvable terminal $v \in T$, there must exist a function $f \in \mathcal{F}$ such that v is lucky with respect to f.

Compute single source mincuts. Now for every $f \in \mathcal{F}$, we do the following. We first construct a capacitated graph $\widetilde{G}$ with vertex capacity function ρ.

- Start with the original graph G. For each non-terminal vertex $v \in V(G) \setminus T$, we set its capacity $\rho(v) = 1$. For each vertex $v \in T$, if $f(v) = 1$, set its capacity $\rho(v) = \infty$, otherwise (i.e. $f(v) = 2$ or $f(v) = 3$) set its capacity $\rho(v) = 1$.
- For each connected component C of $H[f^{-1}(1)]$, we add a super vertex t_C with capacity $\rho(t_C) = \infty$ and add edges from t_C to each vertex in $C \cup (N_H(C) \cap f^{-1}(2))$.
- Add a super vertex s with capacity $\rho(s) = \infty$, and add edges from s to each vertex in X.

Note that a vertex cut $(\widetilde{L}, \widetilde{R})$ in $\widetilde{G}$ (with finite cut size) naturally corresponds to a vertex cut (L, R) in G where L and R are obtained by dropping super vertices from $\widetilde{L}$ and $\widetilde{R}$ respectively. Therefore, when we have a cut $(\widetilde{L}, \widetilde{R})$ (possibly with some subscripts and superscripts) in $\widetilde{G}$, we use (L, R) (with the same subscripts and superscripts) to denote its corresponding cut in G.

On the graph $\widetilde{G}$, we run the single source mincut algorithm, Theorem 3.6, with the vertex s as the source, the vertices $T_C = \{t_C \mid \text{connected components } C \text{ of } H[f^{-1}(1)]\}$ as sinks, and the same parameter k'. Let $T_C^* = \{t_C \in T_C \mid \lambda_{\widetilde{G}}(t_C, s) \leq k'\}$. The output is a mincut cover $\widetilde{\mathcal{K}}_f$ with respect to s and T_C^* in $\widetilde{G}$, and the width of $\widetilde{\mathcal{K}}_f$ is $O(k' \log^3 n)$. The subscript f of $\widetilde{\mathcal{K}}_f$ means $\widetilde{\mathcal{K}}_f$ is with respect to function f.

The final output for Lemma 5.13. Let $\mathcal{K}$ be obtained by replacing each cut $(\widetilde{L}, \widetilde{R}) \in \bigcup_{f \in \mathcal{F}} \widetilde{\mathcal{K}}_f$ with its corresponding cut (L, R) in G. Then obtain another mincut cover $\mathcal{K}_{\mathrm{wit}}$ by keeping only those cuts of $\mathcal{K}$ which are (X, T, k')-witnesses. The set $Q \subseteq T$ is defined as

$$Q = \bigcup_{(L,R) \in \mathcal{K}_{\mathrm{wit}}} (L \setminus R) \cap T.$$

The collection $\mathcal{C}$ that will be output is the $\mathcal{C} \in \mathcal{K}_{\mathrm{wit}}$ that maximizes $|\bigcup_{(L,R)\in\mathcal{C}}(L \setminus R) \cap T|$.

Correctness of Lemma 5.13. First we show that, for each color function $f \in \mathcal{F}$ and each (X, T, k')-carvable vertex $v \in T$ that is lucky with respect to f, there is an (X, T, k')-witness $(L, R) \in \mathcal{K}_f$ with $v \in L \setminus R$.

Lemma 5.17. *Let $v \in T$ be a (X, T, k')-carvable vertex, and let $f \in \mathcal{F}$ be a function s.t. v is lucky w.r.t. f. Let C_v be the connected component of $H[f^{-1}(1)]$ containing v. We have*

1. *the size of t_{C_v}-s mincut in $\widetilde{G}$, i.e. $\lambda_{\widetilde{G}}(t_{C_v}, s)$, is at most k', and*
2. *any t_{C_v}-s mincut $(\widetilde{L}, \widetilde{R})$ in $\widetilde{G}$ corresponds to a (X, T, k')-witness (L, R) in G with $v \in L \setminus R$.*

We exploit the above Lemma 5.17 to prove this (we will prove Lemma 5.17 soon). First, we have $t_{C_v} \in T^*_C$ because $\lambda_{\widetilde{G}}(t_{C_v}, s) \leq k'$ as statement 1 in Lemma 5.17 says. Because $\widetilde{\mathcal{K}}_f$ is a mincut cover with respect to s and T^*_C in $\widetilde{G}$, there is a t_{C_v}-s mincut $(\widetilde{L}, \widetilde{R})$ in $\widetilde{\mathcal{K}}_f$. By statement 2 in Lemma 5.17, the cut in G corresponding to $(\widetilde{L}, \widetilde{R})$ is a (X, T, k')-witness with $v \in L \setminus R$.

Next, because every (X, T, k')-carvable vertex $v \in T$ is lucky w.r.t. some $f \in \mathcal{F}$ by Lemma 5.16, we know Q includes all (X, T, k')-carvable vertices in T. Also Q is disjoint from X because $L \setminus R$ is disjoint from X for all $(L, R) \in \mathcal{K}$. Trivially $\mathcal{C}$ is a collection of disjoint cuts because each $\widetilde{\mathcal{K}}_f$ is a mincut cover. Lastly, we have $|\bigcup_{(L,R)\in\mathcal{C}}(L \setminus R) \cap T| \geq |Q|/O(|\mathcal{F}| \cdot k' \log^3 n) \geq |Q|/(2^{O(k' \log k') \log n})$, because the width of $\mathcal{K}$ is $O(k' \log^3 n) \cdot |\mathcal{F}|$.

Proof of Lemma 5.17. We prove statements 1 and 2 separately.

Statement 1. When v is lucky with respect to f, by definition, the connected (X, T, k')-witness (L^*_v, R^*_v) with $v \in L^*_v \setminus R^*_v$ we fixed above has $L^*_v \cap R^*_v \cap T = W_v \subseteq f^{-1}(2)$. By the construction of $\widetilde{G}$, vertices in $L^*_v \cap R^*_v$ are all have capacity 1 in $\widetilde{G}$, so $\rho(L^*_v \cap R^*_v) = |L^*_v \cap R^*_v| \leq k'$.

Next, observe that removing $L^*_v \cap R^*_v$ will disconnect t_{C_v} from s in $\widetilde{G}$ due to the following reasons. Recall that we only connect t_{C_v} to $C_v \cup (N_H(C_v) \cap f^{-1}(2))$ and only connect s to X, so it suffices to show $C_v \cup (N_H(C_v) \cap f^{-1}(2)) \subseteq L^*_v$ and $X \subseteq R^*_v$.

- We first prove $C_v \cup (N_H(C_v) \cap f^{-1}(2)) \subseteq L^*_v$. Note that $C_v \subseteq (L^*_v \setminus R^*_v) \cap T$ because we have blocked all H-neighbors of $(L^*_v \setminus R^*_v) \cap T$ using colors 2 and 3 (more precisely, we have $N_H((L^*_v \setminus R^*_v) \cap T) \subseteq W_v \cup Y_v \subseteq f^{-1}(2) \cup f^{-1}(3)$). Furthermore, we have $N_H(C_v) \cap f^{-1}(2) \subseteq L^*_v$ because $N_H(C_v) \cap (R^*_v \setminus L^*_v) \subseteq N_H((L^*_v \setminus R^*_v) \cap T) \cap (R^*_v \setminus L^*_v) = N_H((L^*_v \setminus R^*_v) \cap T) \setminus W_v = Y_v \subseteq f^{-1}(3)$.
- We have $X \subseteq R^*_v$ because (L^*_v, R^*_v) is an (X, T, k')-carvable witness.

Therefore, we can conclude that $\lambda_{\widetilde{G}}(t_{C_v}, s) \leq \rho(L^*_v \cap R^*_v) = |L^*_v \cap R^*_v| \leq k'$.

Statement 2. We will show that (L, R) is an (X, T, k')-witness by verifying the properties stated in Definition 5.4 one by one (recall that (L, R) is the cut in G corresponding to $(\widetilde{L}, \widetilde{R})$).

Properties 1 and 3 are easy to see.

1. $|L \cap R| \le k'$ is because $\lambda_{\widetilde{G}}(t_{C_v}, s) \le k'$

3. $X \subseteq R$ because when constructing $\widetilde{G}$, s is connected to all vertices in X.

To see property 2, i.e. $|L \cap T| > |L \cap R|$, we consider two cases.

Case 1. The first case is $|Z_v| = k' + 1$. Note that $Z_v \subseteq C_v$ (since $Z_v \subseteq f^{-1}(1)$ and $H[Z_v]$ is connected) and $C_v \subseteq (L \setminus R) \cap T$ (since all vertices in $f^{-1}(1) \supseteq C_v$ has infinite capacity in $\widetilde{G}$), so trivially we have $|(L \setminus R) \cap T| \ge |C_v| \ge |Z_v| = k' + 1 > k' \ge |(L \cap R) \setminus T|$, which is equivalent to $|L \cap T| > |L \cap R|$.

Case 2. The second case is $Z_v = (L_v^* \setminus R_v^*) \cap T$, which implies $C_v = (L_v^* \setminus R_v^*) \cap T$ by $C_v \subseteq (L_v^* \setminus R_v^*) \cap T$ (shown in the proof of statement 1) and $Z_v \subseteq C_v$. Recall that the witness (L_v^*, R_v^*) we fixed above satisfies $N_H((L_v^* \setminus R_v^*) \cap T) \supseteq L_v^* \cap R_v^* \cap T$, i.e., $W_v \subseteq N_H(C_v)$.Combining $W_v \subseteq f^{-1}(2)$, we have $W_v \subseteq N_H(C_v) \cap f^{-1}(2)$. Thus, by the construction of $\widetilde{G}$, we have $W_v \subseteq L$ (since t_{C_v} is connected to W_v) and $Z_v = C_v \subseteq L \setminus R$ (since t_{C_v} is connected to Z_v, and Z_v-vertices have infinite capacity), which implies

$$|L \cap T| \ge |W_v| + |Z_v| = |L_v^* \cap T|.$$

Therefore, we have

$$|L \cap T| \ge |L_v^* \cap T| > |L_v^* \cap R_v^*| \ge \lambda_{\widetilde{G}}(t_{C_v}, s) = |L \cap R|,$$

where $|L_v^\star \cap T| > |L_v^\star \cap R_v^\star|$ is because $(L_v^\star, R_v^\star)$ is an (X, T, k')-witness.

Finally, we have $v \in C_v \subseteq L \setminus R$. □

Running time of Lemma 5.13. The bottleneck is to call the single-source mincut subroutine Theorem 3.6 for each function $f \in \mathcal{F}$. The total running time is $|\mathcal{F}| \cdot k' m^{1+o(1)} = 2^{O(k' \log k')} m^{1+o(1)}$. This completes the proof of Lemma 5.13. □

Finally, we obtain Lemma 5.12 from Lemma 5.13 by making each witness from Lemma 5.13 to be lean using the following lemma.

Lemma 5.18. *Given an m-vertex graph G, a set of terminals T and a collection of disjoint (X, T, k')-witnesses $\mathcal{C}$, there is an algorithm that computes, for each $(L, R) \in \mathcal{C}$, a lean (X, T, k')-witness (L', R') satisfying that $L' \setminus R' \subseteq L \setminus R$ and $|(L' \setminus R') \cap T| \ge |(L \setminus R) \cap T| / (k' + 1)$. Furthermore, these new lean witnesses are disjoint. The running time is $O(k'm)$.*

Proof. For each cut $(L, R) \in \mathcal{C}$, we do the following. First, we construct a unit-capacitated graph $\widetilde{G}_{\text{local}}$. Initially, $\widetilde{G}_{\text{local}} = G[L] \setminus E(G[L \cap R])$ and assign each vertex with capacity 1. Next, we create a super vertex v_{src} as source and connect v_{src} to each vertex in $L \cap T$. Also, we create a super vertex v_{sink} as sink and connect v_{sink} to each vertex in $L \cap R$.

We compute an arbitrary v^{src}-v^{sink} mincut $(\widetilde{L}, \widetilde{R})$ in $\widetilde{G}_{\text{local}}$. To avoid clutter, in what follows we suppose $v^{\text{src}}, v^{\text{sink}}$ have been removed from $\widetilde{L}$ and $\widetilde{R}$. We define (L', R') by letting $L' = \widetilde{L}$ and $R' = \widetilde{R} \cup (R \setminus L)$. Note that by the construction of $\widetilde{G}_{\text{local}}$, the mincut size $|\widetilde{L} \cap \widetilde{R}| \le |L \cap R|$. We now show (L', R') satisfies the requirements. We start with some simple observation.

1. We have $L \cap T \subseteq \widetilde{L}$, because v_{src} connects to $L \cap T$.

2. We have $L \cap R \subseteq \widetilde{R}$, because v^{sink} connects to $L \cap R$.

3. We have $|\widetilde{L} \cap \widetilde{R}| \le |L \cap R|$ by the construction of $\widetilde{G}_{\text{local}}$.

Note that (L', R') is indeed a cut in G because $L' \setminus R'$ has no edge to $R' \setminus L'$. Concretely, $R' \setminus L' = (\widetilde{R} \setminus \widetilde{L}) \cup (R \setminus L)$, and $L' \setminus R'$ has no edge to $\widetilde{R} \setminus \widetilde{L}$ (because $L' \setminus R' = \widetilde{L} \setminus \widetilde{R}$) and $R \setminus L$ (because $\widetilde{L} \setminus \widetilde{R} \subseteq L \setminus R$ by observation 2).

We first show $L' \setminus R' \subseteq L \setminus R$. Again we have $L' \setminus R' = \widetilde{L} \setminus \widetilde{R}$ by the definition of (L', R'). Combining $\widetilde{L} \subseteq L$ and $L \cap R \subseteq \widetilde{R}$, we have $\widetilde{L} \setminus \widetilde{R} \subseteq L \setminus R$.

Second, we show that (L', R') is a (X, T, k')-witness. First, we have $|L' \cap R'| = |\widetilde{L} \cap \widetilde{R}| \leq |L \cap R| \leq k'$ by observation 3. Next, $|L' \cap T| = |\widetilde{L} \cap T| \geq |L \cap T| > |L \cap R| \geq |L' \cap R'|$, because (L, R) is an (X, T, k')-witness and observations 1 and 3. Finally, we have $X \subseteq R'$ because $L' \setminus R' \subseteq L \setminus R$.

Third, (L', R') is lean by the following reasons. Consider the maxflow in $\widetilde{G}_{\text{local}}$ that certifies $(\widetilde{L}, \widetilde{R})$ is a v^{src}-v^{sink} mincut. The flow paths correspond to $|\widetilde{L} \cap \widetilde{R}|$ vertex disjoint paths in $G[\widetilde{L}]$ from $L \cap T$ (recall that v^{src} connects to $L \cap T$) to $\widetilde{L} \cap \widetilde{R}$. By the definition of (L', R'), these vertex disjoint paths show that (L', R') is lean.

Fourth, we show $|(L' \setminus R') \cap T| \geq |(L \setminus R) \cap T|/(k' + 1)$. Because $\widetilde{L} \setminus \widetilde{R} = L \setminus R$, it suffices to show that $|(\widetilde{L} \setminus \widetilde{R}) \cap T| \geq |(L \setminus R) \cap T|/(k' + 1)$. By observation 1, we have $L \cap T = \widetilde{L} \cap T$. Note that $|(\widetilde{L} \setminus \widetilde{R}) \cap T| \geq |\widetilde{L} \cap T| - |\widetilde{L} \cap \widetilde{R}|$. Let $a = |\widetilde{L} \cap T| = |L \cap T|$ and $b = |\widetilde{L} \cap \widetilde{R}|$. We have $a = |L \cap T| > |L \cap R| \geq b$ and $b \leq |L \cap R| \leq k'$. Therefore, from $a \geq b + 1$ and $b \leq k'$, we have $a - b \geq a/(k' + 1)$. Namely, $|(\widetilde{L} \setminus \widetilde{R}) \cap T| \geq |L \cap T|/(k' + 1) \geq |(L \setminus R) \cap T|/(k' + 1)$

Finally, these new lean witnesses are disjoint because for each $(L, R) \in \mathcal{C}$, its new lean witness (L', R') has $L' = \widetilde{L} \subseteq L$ and $L' \setminus R' = \widetilde{L} \setminus \widetilde{R} \subseteq L \setminus R$. It is easy to check that the new lean witnesses satisfy Definition 3.4.

The whole algorithm just runs maxflow on the graphs $\widetilde{G}_{\text{local}}$ for each cut $(L, R) \in \mathcal{C}$. Because $\mathcal{C}$ is a collection of disjoint cuts, the graphs $\widetilde{G}_{\text{local}}$ are edge-disjoint, so we only runs maxflow on graphs of total size $O(m)$. Since all the maxflow value is bounded by k' by observation 3, the can use the classic Ford-Fulkerson maxflow algorithm. The total running time is $O(k'm)$. □

5.4 The Algorithm for Reducing Adhesion: Proof of Lemma 5.1

The algorithm for Lemma 5.1 is described in Algorithm 1. At a high level, at any time, we maintain a set of terminals T and a set X. Initialize $T = V(G)$ and $X = X_0$. During the algorithm, X keeps growing, and T is roughly shrinking. Additionally, we maintain the invariant that $T \supseteq X$, X is appropriately unbreakable, and T has small adhesion. Once $T = X$, we obtain an unbreakable X with small adhesion as desired.

Algorithm 1 Reducing Adhesion

Input: A graph G with a paramter $\epsilon > 0$ and a (q, k)-unbreakable set $X_0 \subseteq V(G)$.
Output: A $(k\lceil \frac{1}{\epsilon} \rceil + q, k)$ unbreakable set X s.t. $X_0 \subseteq X \subseteq V(G)$ and $\sigma(X) \leq k\lceil \frac{1}{\epsilon} \rceil + q$.

```
Initialize X ← X_0 and T ← V(G)
for ℓ = 1 to ⌈1/ε⌉ do
    Apply Lemma 5.12 on G, T, X and k' = q + ℓk. Let Q and C be the output.
    while |Q| ≥ n^{1−ℓε} do
        T ← carve T with C as in Definition 5.10.
        Recompute Q and C by applying Lemma 5.12 on G, the new T, X, and k' = q + ℓk.
    end while
    X ← T \ Q
end for
Return X
```

To prove the correctess of Algorithm 1, we will show the invariant that, at the beginning of

each phase ℓ,

1. $X_0 \subseteq X \subseteq T$,
2. X is $(q + (\ell - 1)k, k)$-unbreakable,
3. T has adhesion $\sigma(T) \leq q + (\ell - 1)k$, and
4. $|T \setminus X| \leq n^{1-(\ell-1)\epsilon}$.

Note that at the end of the algorithm, we have $X = T$ by the stopping condition of the inner loop. Therefore, once we prove the invariant, we immediately show the correctness of the final output X.

Let us give an intuition about the properties of X. On every phase, we always have $X_0 \subseteq X \subseteq T$. When the algorithm proceeds to the next phase, the unbreakability of X and adhesion of T are slightly relaxed by an additive factor of k. But the set $T \setminus X$ significantly shrinks by a factor of n^ϵ. Intuitively, $T \setminus X$ contains set of vertices that we might carve out from T. Once $T \setminus X = \emptyset$, we have $X = T$ and we are done.[6]

In what follows, we prove the invariant by induction. Trivially, the invariant holds at the beginning of the first phase $\ell = 1$. Now fix a phase $1 \leq \ell \leq \lceil 1/\epsilon \rceil$, assuming the invariant holds at the beginning of this phase, we show that it holds at the beginning of next phase $\ell + 1$.

Each time we compute Q and $\mathcal{C}$ by applying Lemma 5.12 (i.e. in Line 3 and Line 6), we have the following two cases and we first discuss some useful observations in these two cases.

Case 1: $|Q| \geq n^{1-\ell\epsilon}$. In this case we carve T with $\mathcal{C}$. By Lemma 5.12, $\mathcal{C}$ is a collection of disjoint lean $(X, T, q + \ell k)$-witnesses. Let T' be the new T by carving the old T with $\mathcal{C}$ according to Definition 5.10. Note that we have $\sum_{(L,R)\in\mathcal{C}} |(L \setminus R) \cap T| \geq |Q|/\gamma$ by Lemma 5.12 where

$$\gamma = 2^{O((q+\ell k)\log(q+\ell k))} \log n,$$

and each $(L, R) \in \mathcal{C}$ has $|L \cap R| \leq q + \ell k$ because (L, R) is a lean $(X, T, q + \ell k)$-witness. By Lemma 5.11, we have $T' \supseteq X$, $\sigma(T') \leq \max\{\sigma(T), q + \ell k\}$ and

$$|T'| - |T| \geq \sum_{(L,R)\in\mathcal{C}} \frac{|L \setminus R|}{2|L \cap R|} \geq \sum_{(L,R)\in\mathcal{C}} \frac{|(L \setminus R) \cap T|}{2|L \cap R|} \geq \frac{|Q|}{2\gamma(q + \ell k)} \geq \frac{n^{1-\ell\epsilon}}{2\gamma(q + \ell k)}.$$

Case 2: $|Q| < n^{1-\ell\epsilon}$. In this case, we grow X to be $T \setminus Q$. We use X to denote the old X before update, and let X' be the new X after the update. By Lemma 5.12, $Q \subseteq T \setminus X$ and Q includes all $(X, T, q + \ell k)$-carvable vertices in T, so there is no $(X, T, q + \ell k)$-carvable vertex in X'.

Now we verify the invariant at the end of phase ℓ (i.e. at the beginning of phase $\ell + 1$). Again, we let X denote the old X at the beginning of phase ℓ, and let X' denote the new X at the end of this phase.

1. We have $X' \subseteq T$ because $X' = T \setminus Q$. Also, $X \subseteq X'$ because $Q \subseteq T \setminus X$. Thus, by induction, $X_0 \subseteq X \subseteq X'$.
2. Because X' has no $(X, T, q + \ell k)$-carvable vertex and X is $(q + (\ell - 1)k, k)$-unbreakable by assumption, X' is $(q + \ell k, k)$-unbreakable by Lemma 5.5.

[6] This dynamics resembles how the expander decomposition algorithms in [NS17, WN17] work. At the beginning of each phase in these algorithms, the expansion parameter is slightly relaxed but the maximum balance of sparse cuts shrinks significantly.

3. By assumption, at the beginning of this phase, the adhesion of T is at most $q + (\ell - 1)k$. Each time we carve T into a new one (denoted by T'), we have $\sigma(T') \leq \max\{\sigma(T), q + \ell k\}$ by Lemma 5.11. Thus, the final T at the end of this phase has adhesion at most $q + \ell k$.

4. At the end of this phase, $|T \setminus X'|$ is exactly the size of Q in case 2, so $|T \setminus X'| \leq n^{1-\ell\epsilon}$.

Lemma 5.19. *Algorithm 1 runs in time* $\exp(O((q + \frac{k}{\epsilon}) \log(q + \frac{k}{\epsilon})))m^{1+\epsilon+o(1)}$.

Proof. In each phase ℓ, we claim that case 1 will occur at most $O(\gamma(q + \ell k)n^{\epsilon})$ times due to the following reasons. At the beginning of phase ℓ, we have $|T| - |X| \leq n^{1-(\ell-1)\epsilon}$. We have shown that each time we carve T in case 1, the size of T will drop by at least $\frac{n^{1-\ell\epsilon}}{2\gamma(q+\ell k)}$. Therefore, after at most $2\gamma(q + \ell k)n^{\epsilon}$ occurrences of case 1, the differences between the size of T and the size of (the old) X will drop below $n^{1-\ell\epsilon}$. Once this happens, the next iteration of the inner loop will go into case 2 because $Q \subseteq T \setminus X$.

Therefore, there are at most $2\gamma(q + \ell k)n^{\epsilon} = \exp(O((q + \frac{k}{\epsilon}) \log(q + \frac{k}{\epsilon})))n^{\epsilon}$ iterations of the inner loop in one phase. Each iteration runs in time $\exp(O((q+\frac{k}{\epsilon}) \log(q+\frac{k}{\epsilon})))m^{1+o(1)}$ (since the bottleneck is Lemma 5.13) and thus the total running time of one phase is at most $\exp(O((q + \frac{k}{\epsilon}) \log(q + \frac{k}{\epsilon})))mn^{\epsilon+o(1)}$. Because there are at most $O(1/\epsilon)$ phases, the total run-time of the procedure is $\exp(O((q + \frac{k}{\epsilon}) \log(q + \frac{k}{\epsilon})))mn^{\epsilon+o(1)}$ as well. □

6 The Unbreakable Decomposition Algorithm

In this section, we show the main result of our paper, a fast algorithm to construct unbreakable decomposition.

Theorem 6.1. *Given an n-vertex m-edge undirected graph G with a parameters k and $0 < \epsilon \leq 1$, there is a randomized algorithm that, with high probability, computes a $(2\lceil 1/\epsilon \rceil k+3k, k)$-unbreakable decomposition with adhesion $2\lceil 1/\epsilon \rceil k + 2k$, which further has the following properties.*

- *The unbreakable decomposition is compact and admits subtree unbreakability.*
- *The depth of the decomposition is $O(k \log n/\epsilon)$.*
- *The number of tree nodes is $O(n)$ and the total bag size is $O(kn/\epsilon)$.*

The running time is $2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})}m^{1+\epsilon+o(1)}$. Alternatively, the algorithm can compute an $(O(k/\epsilon), k)$-unbreakable decomposition with adhesion $O(k/\epsilon)$, depth $O(\log n)$, and all the other properties as above in asymptotically the same running time.

We now show the algorithm for computing an unbreakable decomposition with $O(k \log(n)/\epsilon)$ depth. Later in Section 6.1, we will discuss how to reduce the depth to $O(\log n)$ by modifying the algorithm slightly.

Proof of Theorem 6.1. We fix a parameter $\sigma = \lceil \frac{1}{\epsilon} \rceil k + k$ (which is an upper bound of the adhesion of the set returned by Lemma 5.1). We will design a subroutine $\textsc{Decomp}(H, B)$, which will receive as input an undirected graph H with a boundary vertex set $B \subseteq V(H)$ such that

1. $|B| \leq 2\sigma$,
2. $H \setminus B$ is connected, and
3. $N_H(V(H) \setminus B) = B$.

We remark that condition 1 is relatively important, while conditions 2 and 3 are only for avoiding some corner cases (so the reader may ignore them for now and come back when they are used). The output of $\textsc{Decomp}(H, B)$ is an unbreakable decomposition (T, β) of H in which the root bag $\beta(r)$ contains B. Then the desired unbreakable decomposition of G can be obtained by invoking $\textsc{Decomp}(G, \emptyset)$. Without loss of generality, we assume G is connected (otherwise, just work on each connected component separately). Thus it satisfies conditions 2 and 3.

In what follows, we first describe $\textsc{Decomp}(H, B)$ and then show the output guarantees of this subroutine.

Algorithm 2 $\textsc{Decomp}(H, B)$

Input: Parameters $k \geq 1, 0 < \epsilon \leq 1$ and $\sigma = \lceil 1/\epsilon \rceil k + k$. A graph H with a boundary vertex set $B \subset V(H)$ such that $|B| \leq 2\sigma$.

Output: A rooted tree decomposition (T, β) of H such that B is contained in the root bag.

if $\sigma + 1 \leq |B| \leq 2\sigma$ **then**
  Apply Lemma 4.5 on B and H to check if B is (k, k)-unbreakable in H.
  if Lemma 4.5 returns a (k, k)-breakable witness (L, R) of B in H **then**
    $X_1 \leftarrow L \cap R$ and $X \leftarrow B \cup X_1$.
  else ▷ B is (k, k)-unbreakable in H
    $X \leftarrow$ apply Lemma 5.1 on graph H with B as the initial set.
  end if
else ▷ $|B| \leq \sigma$
  $X_1 \leftarrow$ apply Theorem 5.2 on graph H with parameters ϵ, k.
  $X \leftarrow B \cup X_1$.
end if
if $X = B$ **then**
  Return a trivial decomposition (T, β) with a single bag $V(H)$.
else ▷ $X \supset B$.
  Initialize (T, β) with only the root node r whose bag is $\beta(r) = X$.
  for each connected component D of $H \setminus X$ **do**
    $(T_D, \beta_D) \leftarrow \textsc{Decomp}(H[D \cup N_H(D)] \setminus E(H[N_H(D)]), N_H(D))$
    Update (T, β) by appending (T_D, β_D) to the root node r.
  end for
  Return (T, β).
end if

Description of Decomp(H, B). See Algorithm 2 for a pseudocode. We now explain the algorithm below.

Compute a "separator" X. We first compute a set $X \subseteq V(G)$ according to the following two cases.

Case 1. If $\sigma + 1 \leq |B| \leq 2\sigma$, we will compute an unbreakable set $X \supseteq B$ with adhesion smaller than $|B|$ to reduce the boundary size of the subproblems.

(a) If B has a (k, k)-breakable witness in H, find such a witness (L, R) using Lemma 4.5 (let $X_1 = L \cap R$), and we let $X = X_1 \cup B$.

(b) Otherwise, B is (k, k)-unbreakable in H. We apply Lemma 5.1 on graph H with B as the initial set, and let X be the output set.

Claim 6.2. *In Case 1, the set X is $(2\lceil 1/\epsilon\rceil k+3k, k)$-unbreakable and has adhesion at most $|B|-1$.*

Proof. We first focus on case (a). X is trivially $(2\lceil 1/\epsilon\rceil k + 3k, k)$-unbreakable because $|X| \leq |X_1| + |B| \leq k + 2\sigma = 2\lceil 1/\epsilon\rceil k + 3k$. To see that X has adhesion $|B| - 1$, consider an arbitrary connected component D of $H \setminus X$. Recall that $X = X_1 \cup B$, so

$$|N_H(D)| = |N_H(D) \cap X_1| + |N_H(D) \cap (B \setminus X_1)|,$$

and we will bound $|N_H(D) \cap X_1|$ and $|N_H(D) \cap (B \setminus X_1)|$ separately. Observe that $|N_H(D) \cap X_1| \leq |X_1| \leq k$, where $|X_1| \leq k$ is trivially because X_1 is a (k, k)-breakable witness. To bound $|N_H(D) \cap (B \setminus X_1)|$, we look at the connected component $D' \supseteq D$ of $H \setminus X_1$ (note that D' is unique since $X_1 \subseteq X$). We have $N_H(D) \cap (B \setminus X_1) \subseteq D' \cap B$ because $N_H(D') \subseteq X_1$. Furthermore, $|D' \cap B| \leq |B| - (k+1)$ because X_1 is a (k, k)-breakable witness of B in H. Therefore, $|N_H(D) \cap (B \setminus X_1)| \leq |B| - (k+1)$. Putting it all together, $|N_H(D)| \leq k + |B| - (k+1) \leq |B| - 1$ as desired.

For (b), the claim holds simply by the guarantees of Lemma 5.1, which guarantees that X is $(k\lceil \frac{1}{\epsilon}\rceil + k, k)$-unbreakable and has $\sigma(X) = k\lceil \frac{1}{\epsilon}\rceil + k$ (which means $\sigma(X) \leq \sigma \leq |B| - 1$). □

Case 2. If $|B| \leq \sigma$, we will compute a balanced unbreakable set $X \supseteq B$ (with adhesion at most 2σ) to significantly reduce the graph size of the subproblems. We first compute X_1 by applying Theorem 5.2 on graph H with parameters ϵ, k, and then let X be $B \cup X_1$.

Claim 6.3. *In Case 2, the set X of H is $(2\lceil 1/\epsilon\rceil k + 2k, k)$-unbreakable and $1/2$-balanced, and has adhesion at most $2\lceil 1/\epsilon\rceil k + 2k$.*

Proof. Theorem 5.2 guarantees that, with high probability, X is $(\lceil 1/\epsilon\rceil k + k, k)$-unbreakable with adhesion at most $\lceil 1/\epsilon\rceil k + k$. Because $X = X_1 \cup B$, X is an $(\lceil 1/\epsilon\rceil k + k + |B|, k)$-unbreakable and $1/2$-balanced set in H with adhesion at most $\lceil 1/\epsilon\rceil k + k + |B|$. Recall that $|B| \leq \sigma = \lceil 1/\epsilon\rceil k + k$, so the claim holds. □

Construct the decomposition. After computing the set X, we construct the decomposition (T, β) as follows. We first consider the corner case that $X = B$, in which we return a trivial decomposition with $V(H)$ as the only bag.

Claim 6.4. *In the corner case $X = B$, we have $|V(H)| \leq 2\sigma$, which implies $V(H)$ is $(2\lceil 1/\epsilon\rceil k + 2k, k)$-unbreakable in H.*

Proof. Note that only case 2 can lead to this corner case. Because from the input conditions 2 and 3, the adhesion of X is exactly $|B|$, contradicting Claim 6.2 for case 1. Therefore, $|X| = |B| \leq \sigma$ and Claim 6.3 tells that X is $1/2$-balanced. Combining that there is only one connected component D in $H \setminus X$ (by input condition 2), we have $|V(H)| \leq |D| + |X| \leq |V(H)|/2 + \sigma$, which means $|V(H)| \leq 2\sigma$. Hence $V(H)$ is trivially $(2\sigma, k)$-unbreakable in H. □

From now we assume $X \supset B$. We consider each connected component D of $H \setminus X$, and solve a subproblem $\textsc{Decomp}(H[D \cup N_H(D)] \setminus E(H[N_H(D)]), N_H(D))$[7] recursively which gives output (T_D, β_D) (note that this will ensure the graphs of the subproblems are edge-disjoint). After solving all the subproblems, we create a node r as the root of T corresponding to bag $\beta(r) = X$, and then append all (T_D, β_D) to r as *child-subtrees*.

[7] In other words, $H[D \cup N_H(D)] \setminus E(H[N_H(D)])$ is the subgraph of H induced by vertices $D \cup N_H(D)$ excluding edges with both endpoints in $N_H(D)$.

Correctness. Let (T, β) be the output of $\textsc{Decomp}(H, B)$.

(T, β) is an unbreakable decomposition. We now prove by induction that (T, β) from $\textsc{Decomp}(H, B)$ is a $(2\lceil 1/\epsilon \rceil k + 3k, k)$-unbreakable decomposition of H with adhesion $2\lceil 1/\epsilon \rceil k + 2k$, further satisfying that the root bag containing B.

We first consider the base case that $\textsc{Decomp}(H, B)$ causes no further recursion, which means the set X is trivially the whole $V(H)$. Then trivially the output (T, β) is a tree decomposition with only a single bag, and this bag is $(O(k/\epsilon), k)$-unbreakable in H by Claim 6.2 and Claim 6.3.

Next, we consider the recursive subproblems that $\textsc{Decomp}(H, B)$ invokes. Recall that each recursion $\textsc{Decomp}(H[D \cup N_H(D)] \setminus E(H[N_H(D)]), N_H(D))$ relates to a connected component D of $H \setminus X$, and we assume all these recursions output child-subtrees (T_D, β_D) correctly. We verify the properties in Definitions 3.2 and 3.3 one by one to show the correctness of $\textsc{Decomp}(H, B)$.

- Consider a vertex $v \in V(H)$. If $v \notin X$, then v is inside a unique connected component D_v of $H \setminus X$, so only the child-subtree (T_{D_v}, β_{D_v}) may have bags containing v, and by the correctness of recursions, bags containing v induce a connected subtree. If $v \in X$, any child-subtree (T_D, β_D) with bags containing v must relate to a component D s.t. $v \in N_H(D) \subseteq X$. Hence, v is inside the root bag of such child-subtrees (T_D, β_D) (since $N_H(D)$ are the boundary vertices of the next recursion). By the construction of (T, β), bags containing v induce a connected subtree.

- Each edge in $E(H)$ will be covered by at least a bag in (T, β), because edges in $H[X]$ are inside the root bag of (T, β), and each of the rest of edges is inside some $H[D \cup N_H(D)] \setminus E(H[N_H(D)])$.

- All bags of (T, β) are $(2\lceil 1/\epsilon \rceil k + 3k, k)$-unbreakable in H by the following reasons, The new bag X is $(2\lceil 1/\epsilon \rceil k + 3k, k)$-unbreakable in H by Claim 6.2 and Claim 6.3. each old bag is even $(2\lceil 1/\epsilon \rceil k + 3k, k)$-unbreakable in some subgraph of H.

- To see that (T, β) has adhesion $2\sigma = 2\lceil 1/\epsilon \rceil k + 2k$, it suffices to show that the root bag X of (T, β) has an intersection of size 2σ with the root bag of each child-subtree (T_D, β_D). Indeed, observe that the intersection is exactly $N_H(D)$, so it has size at most 2σ (since X has adhesion 2σ in H by Claim 6.2 and Claim 6.3).

The tree size and the total bag size. Next, we prove that (T, β) from $\textsc{Decomp}(H, B)$ satisfies (1) $|V(T)| \leq |V(H)|$, (2) the total bag size is at most $O(|V(H)|\sigma) = O(|V(H)|k/\epsilon)$. Recall that the definition of tree decomposition guarantees the $\beta(t) \setminus \sigma(t)$ for all $t \in V(T)$ are disjoint. Hence (1) holds because each $t \in V(T)$ has non-empty $\beta(t) \setminus \sigma(t)$ by Observation 6.5. Next, (2) is because $\sum_{t \in V(T)} |\beta(t)| = \sum_{t \in V(T)} |\beta(t) \setminus \sigma(t)| + \sum_{t \in V(T)} |\sigma(t)| \leq |V(H)| + |V(T)| \cdot (2\sigma) = O(|V(H)|\sigma)$.

Observation 6.5. *For each tree node $t \in V(T)$, its bag $\beta(t)$ is a strict super set of its adhesion $\sigma(t)$, i.e. $\beta(t) \supset \sigma(t)$.*

Proof. To see this, in the recursion step creating t, we have $\sigma(t) = B$. Furthermore, $\beta(t) = V(H)$ if $X = B$ (the corner case) and $\beta(t) = X$ if $X \supset B$, so always $\beta(t) \supset \sigma(t)$. □

The tree depth. We are now ready to show the depth of T is $O(k \log |V(H)|/\epsilon)$. Consider a path from the root (upside) to a leaf (downside) in the recursion tree. We divide this path into two parts, the upper part and the lower part, at the upper most step $\textsc{Decomp}(H', B')$ such that $|V(H')| \leq 10k\lceil 1/\epsilon \rceil$.

For the lower part, i.e. from the step $\textsc{Decomp}(H', B')$ to the leaf, trivially there are at most $|V(H')| = O(k/\epsilon)$ steps (note that the tree decomposition of $\textsc{Decomp}(H', B')$ has at most $|V(H')|$ nodes).

Consider the upper part. Each step $\textsc{Decomp}(H', B')$ on this part has $V(H') > 10k\lceil 1/\epsilon \rceil$. The number of case-2 steps on the upper part is at most $O(\log n)$ by the following reasons. Let $\textsc{Decomp}(H', B')$ be a case-2 step and $\textsc{Decomp}(H'', B'')$ be its sub-step on this path. Because X is 1/2-balanced and has adhesion 2σ by Claim 6.3, we have $|V(H'')| \leq |V(H')|/2 + 2\sigma$, which implies $|V(H'')| \leq 5|V(H)|/6$ combining $2\sigma \leq |V(H)|/3$ (recall that $2\sigma = 2\lceil 1/\epsilon \rceil k + 2k$ and $|V(H)| > 10\lceil 1/\epsilon \rceil k$). Next, the number of *consecutive* case-1 recursive steps is at most σ, because the earliest case-1 step has boundary size at most 2σ and the subsequent case-1 steps will keep reducing the boundary size by Claim 6.2, until the boundary size drops below $\sigma + 1$. Therefore, the total length of the upper part is at most $O(\log n) \cdot \sigma = O(k \log n/\epsilon)$.

In summary the recursion depth is at most $O(k \log n/\epsilon) + O(k/\epsilon) = O(k \log n/\epsilon)$.

Compactness and subtree unbreakability. Finally, the compactness and subtree unbreakability of (T, β) simply follow from the algorithm.

Running Time of Decomp$(G, \emptyset)$. Observe that, at each level i, the total graph size of level-i steps is $O(m)$, because the recursive steps $\textsc{Decomp}(H_i, B_i)$ at a level i are on edge-disjoint graphs H_i. Because there are $O(k \log n/\epsilon)$ levels, the total graph size of each level is $O(km \log n/\epsilon)$. In each recursive step $\textsc{Decomp}(H, B)$, invoking Lemmas 4.2, 4.5 and 5.1 takes time $2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})}|E(H)|^{1+o(1)+\epsilon}$. In summary, the total running time of $\textsc{Decomp}(G, \emptyset)$ is $2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})} m^{1+o(1)+\epsilon}$. □

6.1 Depth Reduction

Finally, we discuss how to reduce the depth of the unbreakable decomposition from $O(k \log n/\epsilon)$ to $O(\log n)$, at a cost of bringing constant factors to the unbreakability and adhesion. Precisely, we will show how to compute a $(O(k/\epsilon), k)$-unbreakable decomposition with adhesion $O(k/\epsilon)$ and depth $O(\log n)$.

We will change Algorithm 2 slightly as follows.

1. We will set $\sigma = 5\lceil 1/\epsilon \rceil k$ (instead of $\sigma = \lceil 1/k \rceil + k$).

2. Before Line 1, we add a termination condition: if $|V(H)| \leq 10\lceil 1/\epsilon \rceil k$, we return a trivial decomposition (T, β) with a single bag $V(H)$. We point out that after adding this new termination condition, the algorithm will never go into Line 13, because by Claim 6.4, there should be $|V(H)| \leq 2\sigma = 10\lceil 1/\epsilon \rceil k$ when we reach Line 13, which is impossible because of the new termination condition.

3. In Line 2, we will check if B is $(\lceil 1/\epsilon \rceil k + k, k)$-unbreakable (instead of (k, k)-unbreakable). This will lead to a new Claim 6.2:

   - In Case 1, the set X is $(O(k/\epsilon), k)$-unbreakable and has adhesion at most $|B| - \lceil 1/\epsilon \rceil k$.

   The new claim can be shown in the same way as Claim 6.2. Roughly speaking, in case (a), X is $(k + 2\sigma, k)$-unbreakable and has adhesion at most $|B| - \lceil 1/\epsilon \rceil k$. In case (b), X is $(2k\lceil 1/\epsilon \rceil + k, k)$-unbreakable and has adhesion $2k\lceil 1/\epsilon \rceil + k \leq \sigma \leq |B| - \lceil 1/\epsilon \rceil k$.

Following the same argument in the proof of Theorem 6.1, it is not hard to see that the resulting decomposition is $(O(k/\epsilon), k)$-unbreakable and has $O(k/\epsilon)$ adhesion. It also has all the other

properties stated in Theorem 6.1. In particular, the depth reduces to $O(\log n)$ by the following reasons. Consider a path from the root to an arbitrary leaf. The number of case-2 steps is at most $O(\log n)$ (note that there is no lower part because of the new termination condition). The number of consecutive case-2 steps is at most $\sigma/(\lceil 1/\epsilon \rceil k) = O(1)$, because every case-2 step will reduce the adhesion by $\lceil 1/\epsilon \rceil k$ (from modification 3).

7 Application: Minimum p-Way Cut in Close-to-Linear Time

The goal of this section is to prove Theorem 1.2. We essentially use the same algorithm as that for Auxiliary Multicut - Theorem 4.1 in [CKL+20], with some minor modifications. We remark that we give the full algorithm only to show that the algorithm indeed runs in close-to-linear time - indeed, the algorithm in [CKL+20] itself does run in close-to-linear time modulo the construction of the decomposition, though this is not analyzed explicitly. Our algorithm is simpler to state since our unbreakable decomposition admits the subtree unbreakability property (see Definition 3.3).

Given a graph G, parameters p and k, we need to decide if there exists a p-Way Cut of size at most k. Recall that a p-Way Cut is a set of edges whose deletion creates p connected components. We begin by computing a (q, k) unbreakable rooted tree decomposition with adhesion at most σ where $\sigma \leq q = O(\frac{k}{\epsilon})$ using Theorem 1.1, where we set ϵ to be some small constant. Let r be the root of this rooted tree decomposition. The running time for this step is $2^{O(\frac{k}{\epsilon} \log \frac{k}{\epsilon})} m^{1+\epsilon}$.

Equivalent formulation. Instead of checking if there exists a p-Way Cut of size at most k, equivalently, we can check if there is some p-coloring of the vertex set such that the number of edges with differently colored endpoints is at most k. Formally, in a graph H, a p-coloring of the vertex set is some function $h : V(H) \to [p]$. The cost of the p-coloring $cost(h)$ is the number of edges with differently colored endpoints. Given two functions $f : A \to C$ and $g : B \to C$ where $A \subseteq B$, we say that g respects f if g agrees with f on every $a \in A$. Recall (see Section 3) that given a node t in the rooted tree in the unbreakable decomposition, G_t is defined as $G[\gamma(t)] \setminus E(G[\sigma(t)])$. In words, G_t is the graph induced on all vertices contained in bags of the sub-tree rooted at t, excluding the edges between the vertices in the adhesion $\sigma(t)$.

Dynamic programming on tree decomposition. We use dynamic programming to solve the problem. For each tree node t in the (unbreakable) tree decomposition, every $I \subseteq [p]$ and every p-coloring $f : \sigma(t) \to [p]$ of the adhesion $\sigma(t)$, we store an entry $M[t, f, I]$ in our table M. $M[t, f, I]$ is an integer value that denotes the cost of the minimum cost p-coloring $h : V(G_t) \to [p]$ in G_t that (a) respects f (b) colors some vertex of G_t with color c for every $c \in I$ an (c) has cost at most k in G_t. If there is no such coloring h, then we set $M[t, f, I] = \infty$. Then since for the root r, $G_r = G$ and $\sigma(r) = \emptyset$, G has a p-Way Cut of size at most k if and only if $M[r, f_\phi, [p]] \leq k$, where f_ϕ is an empty function (note that any coloring h respects the empty function).

We now explain how to compute each entry $M[t, f, I]$. Since we compute these values bottom up with respect to the tree decomposition, we will inductively assume that we have already correctly computed the set of values $M[t', (.), (.)]$ for every child t' of t in the tree. Henceforth we assume that $M[t, f, I] < \infty$, if not, our algorithm will discover this and set $M[t, f, I] = \infty$. Fix a minimum cost p-coloring h satisfying the properties listed above, such that $M[t, f, I] = \text{cost}(h)$. Consider a restriction of this p-coloring h to $\beta(t)$, the bag corresponding to node t. Let $g : \beta(t) \to [p]$ be this coloring.

Exploiting unbreakability. The first observation is that since $\beta(t)$ is (q,k) unbreakable in G_t, the total number of vertices of $\beta(t)$ in all except one colored class of g is at most $3q$ - for otherwise, we would violate the definition of unbreakability. We show this formally in the next lemma.

Lemma 7.1. *There exists at most one $c \in [p]$ such that there exist more than q vertices $v \in \beta(t)$ for which $g(v) = c$. Further, the total number of vertices which have color $c' \neq c$ is at most $3q$.*

Proof. To prove the first part, suppose for contradiction that there exist $c_1, c_2 \in [p]$ with $c_1 \neq c_2$ such that there exist sets $D_1 \subseteq \beta(t)$ and $D_2 \subseteq \beta(t)$ with $g(v_1) = c_1$ for any $v_1 \in D_1$ and $g(v_2) = c_2$ for any $v_2 \in D_2$ with both $|D_1|, |D_2| \geq q + 1$.

Recall that we assumed that there exists a p-coloring h of G_t with cost at most k, whose restriction to $\beta(t)$ gives g. Let $S_1 = h^{-1}(c_1)$ and $S_2 = h^{-1}(c_2)$, where $h^{-1}(c_i)$ is the set of vertices $v \in V(G_t)$ for which $h(v) = c_i$, $i \in \{1, 2\}$. Let S_{12} be the set of vertices in $h^{-1}(c_1)$ which have some neighbor in $h^{-1}(c_2)$ in the graph G_t. Since the cost of h is at most k, we must have $|S_{12}| \leq k$. Now consider the vertex cut $(L = S_1, R = S_{12} \cup S_2)$ in G_t. Indeed, (L, R) is a vertex cut since $L \cap R = S_{12}$ separates $S_1 \setminus S_{12}$ from S_2 in G_t. Also we have $|L \cap R| = |S_{12}| \leq k$. Finally, g is a restriction of h so it follows that $D_1 \subseteq L$ and $D_2 \subseteq R$, and both $|D_1|, |D_2| \geq q+1$. Hence (L, R) is a (q, k) breakable witness for $\beta(t)$ in G_t, which is a contradiction since $\beta(t)$ is (q, k) unbreakable in G_t (recall that our unbreakable decomposition admits subtree unbreakability, see Definition 3.3).Thus every colored class except one, say c, has at most q vertices.

Next, we show that the total number of vertices in $\beta(t)$ colored differently from c is at most $3q$. If this is not the case, since $|h^{-1}(c') \cap \beta(t)| \leq q$ for each $c' \neq c$, we first observe that we can partition the set of colors $[p] \setminus \{c\}$ into two (disjoint) groups $X_1 \cup X_2$, such that there are at least $q + 1$ vertices of $\beta(t)$ having colors from X_1 and X_2.

For a set of colors X, let us define $h^{-1}(X)$ to be the set of vertices $v \in V(G_t)$ such that $h(v) \in X$. Then it follows that $|h^{-1}(X_1)|, |h^{-1}(X_2)| \geq q + 1$. Similar to the previous part, let us define $S'_1 = h^{-1}(X_1)$, $S'_2 = h^{-1}(X_2)$, and S'_{12} to be the set of vertices in S'_1 that have a neighbor in S'_2. Again, it must be the case that $(L = S'_1, R = S'_{12} \cup S'_2)$ is a (q, k) breakable witness for $\beta(t)$ in G_t, which is a contradiction. □

We guess the label of the color class $c \in [p]$ which has the maximum number of vertices $v \in \beta(t)$ satisfying $g(v) = c$, then we can potentially have more than q vertices of that color, but not for any other color. Without loss of generality and for simplicity of exposition let us assume that this color c is the color p.

Computing the table entries. To compute $M[t, f, I]$, our goal will be to come up with a coloring $g^* : \beta(t) \to [p]$, and a set $I_{t'} \subseteq I$ for each child t' of t in the tree such that the following hold.

1. g^* respects f
2. For every $i \in I$, either $i \in I_{t'}$ for some child t' of t, or there exists some $v \in \beta(t)$ such that $g^*(v) = i$.

Define the cost of g^* as follows. Every edge $e = \{u, v\} \in E(\beta(t)) \setminus E(\sigma(t))$ contributes 1 to the cost. For every child t' of t, t' contributes $M[t', g^*|_{\sigma_{t'}}, I_{t'}]$ to the cost, where $g^*|_{\sigma_{t'}}$ denotes the restriction of g^* to the set $\sigma(t')$. Then observe that since we correctly computed the values $M[t', (.), (.)]$ for each child t' of t, such a coloring g^* with minimum cost must have cost equal to $M[t, f, I]$.

Notice that in particular, g itself is a coloring of minimum cost that satisfies these conditions. To find a coloring of minimum cost (possibly different from g), we use color coding together with another dynamic programming step. We start by describing the color coding step.

Color coding. For every edge $e = \{u, v\}$ in $E(\beta(t)) \setminus E(\sigma(t))$, we say that e is *crossing* if $g(u) \neq g(v)$. Similarly, for every child node t' of t in the tree decomposition tree and adhesion $\sigma(t') \subseteq \beta(t)$, we say that $\sigma(t')$ is *crossing* if there exists two vertices $u, v \in \sigma(t')$ such that $g(u) \neq g(v)$.

Lemma 7.2. *The total number of crossing edges and crossing adhesions together is at most k.*

Proof. The proof follows from the compactness (see Section 3) property of the decomposition. Compactness implies that for each child t' of t, $G[\alpha(t')] = G[\gamma(t') \setminus \sigma(t')]$ is connected and $N_{G_{t'}}(\alpha(t')) = \sigma(t')$. Also, the graphs $G_{t'} = G[\gamma(t')] \setminus E(\sigma(t'))$ are disjoint across all children t' of t. If $\sigma(t')$ is crossing, then there must exist some two vertices $u, v \in \sigma(t')$ such that $g(u) \neq g(v)$, and hence each crossing adhesion contributes 1 to the cost of h.

Similarly, every crossing edge $e \in E(\beta(t)) \setminus E(\sigma(t))$ contributes 1 to the cost of h. But the cost of h in G_t is at most k, thus the total number of crossing edges and adhesions together is at most k. □

Let $A_1, A_2 \dots A_p$ be the set of all vertices of $\beta(t)$ colored $1, 2 \dots p$ respectively in g. Let $A^* = \bigcup_{i=1}^{p-1} A_i$ be the set of vertices colored with colors $1, 2 \dots p-1$. Define $B^* \subseteq A_p$ as the set of all vertices $v \in A_p$ for which there is either (a) an edge $\{u, v\} \in E(\beta(t)) \setminus E(\sigma(t))$ with $u \in A^*$ or (b) some child node t' of t such that $\sigma(t')$ is crossing, and $v \in \sigma(t')$.

Apply Lemma 5.16 to the sets $A_1, A_2 \dots A_{p-1}, B^*$ to obtain a family of functions $\mathcal{F}$. The running time is at most $2^{O(q \log q)} n \log^2 n$, since $\sum_{i=1}^{k-1} A_i \leq 3q$ by Lemma 7.1 and $|B^*| \leq qk\sigma \leq q^3$, where σ is the adhesion of the unbreakable decomposition. Then we know that there exists some function $g' \in \mathcal{F}$ that agrees with g on these sets. Henceforth we fix this g' (we try every function from $\mathcal{F}$). Before we proceed further, for every $v \in \sigma(t)$, we make g' agree with f by setting $g'(v) = f(v)$. Let us also assume that at least one vertex of $\beta(t)$ is colored p in g' (note that this is true for g since p is the color with maximum frequency; hence this vertex can be included in the color coding analysis to ensure this, but we avoid explicity stating this to avoid confusion). Now we set $I = I \setminus \{p\}$.

Next, consider the graph H obtained as follows. The vertex set $V(H) = \beta(t)$. We add into $E(H)$ every edge of $E(G_t) \setminus E(\sigma_t)$. Further for every child node t' of t, we turn the adhesion $\sigma(t')$ into a clique. Formally, for every $u, v \in \sigma(t')$ with $u \neq v$ we add the edge $\{u, v\}$ to $E(H)$.

Let H^* be the subgraph of H induced on the vertices which are colored $\{1, 2 \dots p-1\}$ by g', and let $\{C_1, C_2 \dots C_\ell\}$ be the connected components of H^*. We have the following observation.

Observation 7.3. *For connected component C_i, $i \in [\ell]$, either g' agrees with g on vertices of C_i or $g(v) = p$ for each $v \in C_i$.*

Proof. Note that if an edge $\{u, w\} \in E(H)$ is such that $g(u) \neq g(w)$, then by our color coding requirements, $g'(u) = g(u)$ and $g'(w) = g(w)$. Also if $g(u), g(w) \in [p-1]$, then $g'(u) = g(u)$ and $g'(w) = g(w)$.

Consider two cases. The first case is when in g, each vertex of C_i is colored from the set of colors $[p-1]$. In this case, g and g' agree on every vertex of C_i.

Otherwise, there exists a vertex of C_i that is colored with color p in g. We have two subcases now. Either every vertex of C_i is colored p, or there is some edge $\{u, w\} \in E(H)$ such that $g(u) \in [p-1]$ and $g(w) = p$. But in the latter case, by construction, we have $w \in B^*$, and hence $g'(w) = g(w) = p$, which is a contradiction since every vertex of C_i is colored from the set $[p-1]$ in g'. □

Thus there exists an optimal coloring (the coloring g), which for each connected component C_i, $i \in [\ell]$, either colors C_i consistently with g' or colors every vertex of C_i with the color p. Equivalently, in order to compute a coloring g^* of minimum cost satisfying items 1 and 2, we start with the coloring $g^* = g''$ which colors every vertex in $\beta(t)$ with the color p, and then decide to *flip* some connected components C_i, $i \in [\ell]$, so that we change the colors of vertices of C_i in g^* to color them consistently with g'.

For each component C_i, $i \in [\ell]$, define the *cost of flipping* as

$$\text{flip}(C_i) = \sum_{\{u,v\} \in E(\beta(t)) \setminus E(\sigma(t))} \mathbb{I}(u, v \in C_i \text{ and } g'(u) \neq g'(v)) + \sum_{\{u,v\} \in E(\beta(t)) \setminus E(\sigma(t))} \mathbb{I}(u \in C_i, v \notin C_i).$$

Essentially this is the cost for flipping C_i: we charge for all edges in the bag $\beta(t)$ with differently colored endpoints inside C_i, and for all edges which have exactly one endpoint in C_i (note that in this case the other endpoint is colored p). Before we proceed further, we note that if a component C_i contains a vertex $v \in C_i \cap \sigma(t)$, then since $g'(v) = f(v) = g(v) \neq p$, this already fixes the decision for this component: we flip C_i.

Flipping components using dynamic programming. It now suffices to decide which subset of components to flip. To accomplish this we will yet again use dynamic programming. For this, we need a few more observations and definitions.

First, observe that every adhesion of a child node is a clique in H, and hence can intersect at most one C_i, $i \in [\ell]$. For each component C_i, let x_i be the number of children t' of the node t whose adhesion $\sigma(t')$ intersects C_i. Arbitrarily order these child nodes by numbering them $(i, 1)$, $(i, 2)$, $(i, 3) \ldots (i, x_i)$, so that $t'_{i',j'}$ refers to the child node labelled (i', j'). Thus $\sigma(t'_{i',j'})$ refers to the adhesion of the child node labeled (i', j'). For simplicity, let us assume that for every child node t' of t, $\sigma(t')$ intersects some C_i, $i \in [\ell]$ - if not, this is easy to handle using a simple modification to the dynamic programming.

Given a coloring $\psi : \beta(t) \to [p]$, $i \in [\ell]$ and $j \in [x_i]$, a *flip set* $F \subseteq [i]$ and assignments $I_{t'_{i',j'}} \subseteq I$ to each child node numbered (i', j') satisfying either (a) $i' < i$, or (b) $i' = i$ and $j' \leq j$, the i, j*-partial cost* of ψ is defined as follows:

- Each component $C_{i'}$, such that $i' \in F$, contributes a cost of $\text{flip}(C_i)$.
- Each child node $t'_{i',j'}$ such that either (a) $i' < i$, or (b) $i' = i$ and $j' \leq j$ contributes a cost $M[t', \psi|_{\sigma_{t'_{i',j'}}}, I_{t'_{i',j'}}]$ where $\psi|_{\sigma_{t'_{i',j'}}}$ is the restriction of ψ to $\sigma(t'_{i',j'})$.

We create a dynamic programming table, whose each entry is of the form $T[i][j][I_0][b]$, $i \in \{0, 1, 2 \ldots \ell\}, j \in \{0, 1, \ldots x_i\}, I_0 \subseteq I, b \in \{0, 1\}$. $T[i][j][I_0][b]$ denotes the minimum possible i, j-partial cost across all choices of colorings $\psi : \beta(t) \to [p]$, flip sets $F \subseteq [i]$ and sets $I_{t'_{i',j'}} \subseteq I_0$ for all (i', j') such that either $i' < i$ or $i' = i, j' \leq j$, while maintaining the following properties.

- ψ is obtained from g'' by flipping some subset of components with indices equal to $F \subseteq [i]$.
- If $b = 1$, then $i \in F$, else $i \notin F$.
- For each $i' \leq i$, if $C_{i'} \cap \sigma(t) \neq \emptyset$, then $i' \in F$.
- For every $i_0 \in I_0$, either there exists some vertex $v \in \beta(t)$ with $\psi(v) = i_0$, or there exists a child node of t labelled i', j' with either (a) $i' < i$ or (b) $i' = i$ and $j' \leq j$ such that $v \in I_{t'_{i',j'}}$.

The updates to $T[i][j][I_0][b]$ are natural. For the base cases, we set $T[0][0][\emptyset][0] = T[0][0][\emptyset][1] = 0$. For any other set I_0 and $b' \in \{0,1\}$ we set $T[0][0][I_0][b'] = \infty$. If $i > 0$ and $j = 0$, there are two cases. If $C_i \cap \sigma(t) \neq \emptyset$, then we set $T[i][j][I_0][0] = \infty$. Else we just set $T[i][j][I_0][0] = \max\{T[i-1][x_{i-1}][I_0][0], T[i-1][x_{i-1}][I_0][1]\}$. In either case we set $T[i][j][I_0][1] = \max\{T[i-1][x_{i-1}][I_0 \setminus I_i][0], T[i-1][x_{i-1}][I_0 \setminus I_i][1]\} + \text{flip}(C_i)$, where I_i is the set of colors given to vertices in C_i in g'.

For $j > 0$, if $\sigma(t) \cap C_i \neq \emptyset$ we set $T[i][j][I_0][0] = \infty$. Otherwise, we set

$$T[i][j][I_0][0] = \min_{I_0'' \subseteq I_0} M[t'_{i,j}][f_0][I_0''] + T[i][j-1][I_0 \setminus I_0''][0].$$

In either case, we set

$$T[i][j][I_0][1] = \min_{I_0'' \subseteq I_0} M[t'_{i,j}][f_1][I_0''] + T[i][j-1][I_0 \setminus I_0''][1].$$

Here f_0 is the constant function that assigns to every vertex of $\sigma(t'_{i,j})$ the color p, and f_1 is the function that colors each vertex of $\sigma(t'_{i,j})$ the same color as that in g'.

Finally, we set $M[t, f, I]$ to be $\min(T[\ell][x_\ell][I_0][0], T[\ell][x_\ell][I_0][1]\}$. Recall that in reality, we repeat this process for every function $g' \in \mathcal{F}$ from the color coding step: therefore we define $M[t, f, I]$ to be the minimum over this result for every function in $\mathcal{F}$. If this quantity is more than k for every choice of the coloring from $\mathcal{F}$ obtained, we just set $M[t, f, I]$ to ∞.

Running time. The running time analysis is rather straightforward. Let us first analyse the run-time given the unbreakable decomposition. Fix an entry $M[t, f, I]$. There are at most $p^{O(\sigma)}$ such entries for every tree node t. To compute each entry, we first do a color coding step and a dynamic programming step which constructs the table T. Let x_t be the total number of children of t in the tree decomposition. Let $n_{tv} = |\beta(t)|$ be the number of vertices in the bag and n_{te} be the number of edges in $E(\beta(t)) \setminus E(\sigma(t))$.

The color coding step takes time $O(2^{O(\frac{k}{\epsilon}\log\frac{k}{\epsilon})}(n_{tv} + n_{te})\log^2 n)$.

There are at most $(x_t + n_{tv}) \cdot 2^p \cdot 2 = O((x_t + n_{tv}) \cdot 2^p)$ entries in the table T. Each entry is computed in time $O(2^p)$. Thus the total running time to compute T is $O((x_t + n_{tv})2^p)$. Summing across all nodes t, using the facts that $\sum_t (n_{tv} + n_{te}) = O(m + kn/\epsilon)$ ($\sum_t n_{tv} \leq O(kn/\epsilon)$ is from Theorem 6.1 and $\sum_t n_{te} = O(m)$ is from the nature of tree decomposition), and further noting that $\sum_t x_t = O(n)$ by Theorem 6.1, the total running time is at most $2^{O(\frac{k}{\epsilon}\log\frac{k}{\epsilon})}(m+n)\log^2 n$.

Finally, the unbreakable decomposition itself is computed using Theorem 6.1, and this is the bottleneck for the running time. The running time of the entire algorithm is therefore $2^{O(\frac{k}{\epsilon}\log\frac{k}{\epsilon})}m^{1+\epsilon}$. This concludes the run-time analysis.

8 Conclusion and Open Problems

Theorem 6.1 gives the first close-to-linear time FPT algorithm for unbreakable decomposition. Our decomposition also has optimal unbreakability and adhesion parameters up to a constant factor. This removes the bottleneck to fast FPT algorithms for numerous problems.

Below, we list some exciting potential applications that relied on unbreakable decomposition:

1. Can we find a Minimum Bisection of size k in $O_k(m^{1+\epsilon})$ time, if it exists?

2. Can we improve the preprocessing time of the $O(1)$-update-time connectivity oracle under $O(1)$ vertex failures from [PSS+22] to close-to-linear time? What is the optimal trade-off for space and preprocessing time when the update time is $O(1)$?

3. Let G be a graph that excludes a fixed topological minor. Can we decide whether G satisfies an $\mathsf{FO} + \mathsf{conn}$ sentence or an $\mathsf{FO} + \mathsf{DP}$ sentence in $O(m^{1+\epsilon})$ time? This would be an improvement over the $O(n^3)$ bound given by [PSS+22, SSS+24]. It is worth noting that there are already a linear time algorithm for deciding an MSO_2 sentence in bounded-treewidth graphs [Cou90], and a close-to-linear time algorithm for deciding an FO sentence in nowhere dense graphs [GKS17].

Can our running time be improved further? More specifically, can one compute $(O(k), k)$-unbreakable decomposition with adhesion $O(k)$ in $O_k(m \log^{O(1)} m)$ or $O_k(m^{1+o(1)}))$ time?

Moreover, we can hope to remove the exponential dependency on k in the edge-cut version. As shown in [LSS22], a $(\mathrm{poly}(k), k)$-*edge*-unbreakable decomposition can be computed in polynomial time with no exponential dependency on k. A near-linear time algorithm for computing an $(\mathrm{poly}(k \log n), k)$-edge-unbreakable decomposition will likely imply a nice application, i.e., a near-linear time $(1+\epsilon)$-approximation FPT algorithm for Minimum p-Way Edge-Cut parameterized by p, improving the polynomial running time of the algorithm in [LSS22].

References

[AKP+22] Akanksha Agrawal, Lawqueen Kanesh, Fahad Panolan, MS Ramanujan, and Saket Saurabh. A fixed-parameter tractable algorithm for elimination distance to bounded degree graphs. *SIAM Journal on Discrete Mathematics*, 36(2):911–921, 2022. 1, 2

[AKT21] Amir Abboud, Robert Krauthgamer, and Ohad Trabelsi. Subcubic algorithms for gomory–hu tree in unweighted graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1725–1737, 2021. 34

[CCH+16] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. Designing fpt algorithms for cut problems using randomized contractions. *SIAM Journal on Computing*, 45(4):1171–1229, 2016. Announced at FOCS'12. 1, 2, 15

[CDK+21] Parinya Chalermsook, Syamantak Das, Yunbum Kook, Bundit Laekhanukit, Yang P Liu, Richard Peng, Mark Sellke, and Daniel Vaz. Vertex sparsification for edge connectivity. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1206–1225. SIAM, 2021. 1

[CHM13] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM Journal on Computing*, 42(4):1674–1696, 2013. 1

[CKL+20] Marek Cygan, Paweł Komosa, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, Saket Saurabh, and Magnus Wahlström. Randomized contractions meet lean decompositions. *ACM Transactions on Algorithms (TALG)*, 17(1):1–30, 2020. , 1, 2, 5, 10, 11, 12, 14, 15, 25, 37

[CLL+08] Jianer Chen, Yang Liu, Songjian Lu, Barry O'sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 177–186, 2008. 1

[CLP+19] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Minimum bisection is fixed-parameter tractable. *SIAM Journal on Computing*, 48(2):417–450, 2019. , 1, 2

[Cou90] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990. 30

[CPPW13] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Transactions on Computation Theory (TOCT)*, 5(1):1–11, 2013. 1

[CQ21] Chandra Chekuri and Kent Quanrud. Isolating cuts,(bi-) submodularity, and faster algorithms for connectivity. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021. 34

[FM06] Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 375–384, 2006. 3, 8

[GKS17] Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM (JACM)*, 64(3):1–32, 2017. 30

[GLL18a] Anupam Gupta, Euiwoong Lee, and Jason Li. Faster exact and approximate algorithms for k-cut. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 113–123. IEEE, 2018. 1

[GLL18b] Anupam Gupta, Euiwoong Lee, and Jason Li. An fpt algorithm beating 2-approximation for k-cut. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2821–2837. SIAM, 2018. 1

[Gui11] Sylvain Guillemot. Fpt algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011. 1

[HLW21] Zhiyang He, Jason Li, and Magnus Wahlström. Near-linear-time, optimal vertex cut sparsifiers in directed acyclic graphs. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021. 1

[ILSS23] Tanmay Inamdar, Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. Parameterized complexity of fair bisection:(fpt-approximation meets unbreakability). In *31st Annual European Symposium on Algorithms (ESA 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023. 1, 2

[Iwa16] Yoichi Iwata. Linear-time kernelization for feedback vertex set. *arXiv preprint arXiv:1608.01463*, 2016. 1

[IWY16] Yoichi Iwata, Magnus Wahlstrom, and Yuichi Yoshida. Half-integrality, lp-branching, and fpt algorithms. *SIAM Journal on Computing*, 45(4):1377–1411, 2016. 1

[IYY18] Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 0/1/all csps, half-integral a-path packing, and linear-time fpt algorithms. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 462–473. IEEE, 2018. 1, 2

[KKPW21] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation i: directed graphs. *arXiv preprint arXiv:2111.03450*, 2021. 1

[KKPW24] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Flow-augmentation ii: Undirected graphs. *ACM Transactions on Algorithms*, 20(2):1–26, 2024. 1, 2

[KL20] Ken-ichi Kawarabayashi and Bingkai Lin. A nearly 5/3-approximation fpt algorithm for min-k-cut. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 990–999. SIAM, 2020. 1

[Kle04] Jon Kleinberg. Detecting a network failure. *Internet Mathematics*, 1(1):37–55, 2004. 8

[KMP+24] Eun Jung Kim, Tomáš Masařík, Marcin Pilipczuk, Roohani Sharma, and Magnus Wahlström. On weighted graph separation problems and flow augmentation. *SIAM Journal on Discrete Mathematics*, 38(1):170–189, 2024. 1

[KPS24] Tuukka Korhonen, Michał Pilipczuk, and Giannos Stamoulis. Minor containment and disjoint paths in almost-linear time. *arXiv preprint arXiv:2404.03958*, 2024. 10, 14

[KT11] Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 160–169. IEEE, 2011. 1, 2

[KW12] Stefan Kratsch and Magnus Wahlström. Representative sets and irrelevant vertices: New tools for kernelization. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 450–459. IEEE, 2012. 1

[KW14] Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. *ACM Transactions on Algorithms (TALG)*, 10(4):1–15, 2014. 1

[LM13] Daniel Lokshtanov and Dániel Marx. Clustering with local restrictions. *Information and Computation*, 222:278–292, 2013. 1

[LNP+21] Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 317–329, 2021. 34

[LNPS23] Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. Near-linear time approximations for cut problems via fair cuts. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 240–275. SIAM, 2023. 34

[LNR+14] Daniel Lokshtanov, NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms (TALG)*, 11(2):1–31, 2014. 1

[LP20] Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 85–92. IEEE, 2020. 34

[LP21] Jason Li and Debmalya Panigrahi. Approximate gomory–hu tree is faster than n–1 max-flows. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1738–1748, 2021. 5, 34

[LRSZ18] Daniel Lokshtanov, MS Ramanujan, Saket Saurabh, and Meirav Zehavi. Reducing cmso model checking to highly connected graphs. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2018. 2

[LSS22] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-cut. *SIAM Journal on Computing*, (0):FOCS20–205, 2022. 1, 2, 30

[LSSZ19] Daniel Lokshtanov, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Balanced judicious bipartition is fixed-parameter tractable. *SIAM Journal on Discrete Mathematics*, 33(4):1878–1911, 2019. 1, 2

[Mar06] Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006. 2

[MOR13] Dáaniel Marx, Barry O'sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms (TALG)*, 9(4):1–35, 2013. 1, 2

[MR14] Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM Journal on Computing*, 43(2):355–388, 2014. 1

[NS17] Danupon Nanongkai and Thatchaphol Saranurak. Dynamic spanning forest with worst-case update time: adaptive, las vegas, and $O(n^{1/2-\epsilon})$-time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1122–1129, 2017. 19

[PSS+22] Michał Pilipczuk, Nicole Schirrmacher, Sebastian Siebertz, Szymon Toruńczyk, and Alexandre Vigny. Algorithms and data structures for first-order logic with connectivity under vertex failures. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022. 1, 2, 30

[PSY22] Seth Pettie, Thatchaphol Saranurak, and Longhui Yin. Optimal vertex connectivity oracles. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 151–161, 2022. 5, 34

[Ree97] Bruce Reed. Tree width and tangles: a new connectivity measure and some applications. *Surveys in combinatorics*, 241:87–162, 1997. 37

[SSS+24] Nicole Schirrmacher, Sebastian Siebertz, Giannos Stamoulis, Dimitrios M Thilikos, and Alexandre Vigny. Model checking disjoint-paths logic on topological-minor-free graph classes. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–12, 2024. 1, 2, 30

[SZ23] Saket Saurabh and Meirav Zehavi. Parameterized complexity of multi-node hubs. *Journal of Computer and System Sciences*, 131:64–85, 2023. 1, 2

[Wah22] Magnus Wahlström. Quasipolynomial multicut-mimicking networks and kernels for multiway cut problems. *ACM Transactions on Algorithms (TALG)*, 18(2):1–19, 2022. 1

[WN17] Christian Wulff-Nilsen. Fully-dynamic minimum spanning forest with improved worst-case update time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1130–1143, 2017. 19

A Single Source Vertex Mincuts

In this section, we show how to compute single source vertex mincuts. Our proof is based on previous algorithms for computing min-cut thresholds in graphs. Given a graph G, a single source $s \in V(G)$, a set of targets $T \subseteq V(G)$ and a parameter μ, the single source min-cut threshold problem asks to find the set of all targets $t \in T$ such that the size of the (t, s) edge/vertex min-cut is at most λ. [LP21] showed an algorithm for the edge version using minimum isolating cuts [LP20, AKT21]. Later, [PSY22] generalized this to the vertex version using minimum isolating vertex cuts [LNP+21, CQ21]. Their algorithm works on unit-capacity graphs, but it is easy to see that this generalizes to capacitated graphs. For our setting, we need a slightly different notion in that we need exact min-cuts instead of approximate min-cuts: but this can be ensured easily by running the threshold algorithm for each parameter μ from 1 to k by paying a factor of k in the running time, which we can indeed afford. For completeness, we give the complete proof for our setting, and we follow the simpler proof shown in [LNPS23].

Theorem 3.6. *Consider an m-edge capacitated graph $\widetilde{G}$ with vertex capacity function ρ, a parameter k, a single source vertex s and sink vertices T satisfying that $\{s\} \cup T$ is an independent set and each source/sink vertex has capacity ∞. Let T^* be the set of sink vertices t with $\lambda_{\widetilde{G}}(t, s) \leq k$. There is a randomized algorithm that, with high probability, computes a mincut cover $\mathcal{K}$ with respect to s and T^* which has width $O(k \log^3 n)$. The running time is $O(km^{1+o(1)})$.*

We will use the isolating vertex cuts subroutine stated in Theorem A.2. Recall that a $\mathcal{C}$ is collection of disjoint cuts if the cuts $(L, R) \in \mathcal{C}$ have mutually disjoint $L \setminus R$.

Definition A.1 (Isolating Cuts)**.** *Consider a capacitated graph $\widetilde{G}$ with terminals $W \subseteq V(\widetilde{G})$ that form an independent set. For each terminal $w \in W$, an* isolating cut *of w with respect to W is an arbitrary w-$(W \setminus \{w\})$ mincut.*

Theorem A.2 (Isolating Vertex Cuts [LNP+21, CQ21])**.** *Given a capacitated graph $\widetilde{G}$ with terminals $W \subseteq V(\widetilde{G})$ that form an independent set, there is an algorithm that computes a collection $\mathcal{C}$ of disjoint cuts satisfying that*

- *for each $w \in W$, there is a $(L, R) \in \mathcal{C}$ which is an isolating cut of w;*
- *each $(L, R) \in \mathcal{C}$ is an isolating cut of some $w \in W$.*

The running time is $m^{1+o(1)}$.

We note that the disjointness of the isolating cuts from Theorem A.2 can be seen easily following the algorithm in [LNP+21].

Algorithm 3 Single Source Vertex mincut

Input: a capacitated graph $\widetilde{G}$ with a parameter k, a source vertex s and sink vertices T.
Output: A set $\mathcal{C}$ of vertex cuts.

Initialize Γ to be an empty vertex set and $\mathcal{K}$ to be an empty mincut cover.
for $k' = 1, 2, ..., k$ **do**
  for $j = 1, 2 \dots O(\log^2 n)$ **do**
    Initialize $\mathcal{K}'$ to be an empty mincut cover.
    for $i = 0, 1, 2 \dots \lfloor \log n \rfloor, r = 2^i$ **do**
      Sample each sink vertex in $T \setminus \Gamma$ with probability $\frac{1}{r}$ independently into a set T'.
      $W \leftarrow T' \cup \{s\}$
      Let $\mathcal{C}_{\text{iso}}$ be the collection of isolating cuts of W in $\widetilde{G}$ (use Theorem A.2)
      Let $\mathcal{C}$ collect the cuts $(L, R) \in \mathcal{C}_{\text{iso}}$ which has size k' and is an isolating cut of some sink $t \in T' \subseteq W$
      Add $\mathcal{C}$ into $\mathcal{K}'$.
    end for
    For each cut $(L, R) \in \mathcal{K}'$, $\Gamma \leftarrow \Gamma \cup (L \setminus R) \cap (T \setminus \Gamma)$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \mathcal{K}'$
  end for
end for
return $\mathcal{K}$

Proof of Theorem 3.6. The single source vertex mincut algorithm is described in Algorithm 3 in details. In what follows, we prove that $\mathcal{K}$ is a mincut cover with respect to s and T.

Instead of directly verifying $\mathcal{K}$ satisfies the properties in Definition 3.5, we will first show Lemma A.3.

Lemma A.3. *With high probability, at the end of each phase k' of the outer loop, Γ exactly collects all sink vertices t with $\lambda_{\widetilde{G}}(t, s) \le k'$, i.e., at the beginning of each phase k', $T \setminus \Gamma$ exactly contains sink vertices with $\lambda_{\widetilde{G}}(t, s) \ge k'$.*

Proof. We will use induction. Note that initially this claim holds at the beginning of the first phase $k' = 1$. To enable the induction, fix an arbitrary phase $1 \le k' \le k$, and we will show that this claim holds at the end of phase k' with high probability, assuming that it holds at the beginning of phase k'.

For each single iteration j of the middle loop, let $\widetilde{T}_j$ collect all sink vertices $t \in T \setminus \Gamma$ s.t. $\lambda(t, s) = k'$ at the beginning of iteration j, and let ℓ_j be a random variable denoting the size of $\widetilde{T}_j$. Claim A.4 says that $E[\ell_j]$ will drop by a factor of $\Omega(1/\log n)$ in each iteration j of the middle loop. Therefore, after $O(\log^2 n)$ (with sufficiently large hidden constant) iterations j, in expectation the number of sink vertices $t \in T \setminus \Gamma$ s.t. $\lambda(t, s) = k'$ is at most $1/\text{poly}(n)$. By Markov's inequality, with high probability, the number of sink vertices $t \in T \setminus \Gamma$ s.t. $\lambda(t, s) = k'$ at the end of phase k' is 0.

Claim A.4. *For each iteration j, we have $E[\ell_{j+1}] \le \ell_j(1 - \Omega(1/\log n))$.*

Proof. At the end of each iteration j, we define a random variable

$$\alpha_j = \sum_{\mathcal{C} \in \mathcal{K}'} \sum_{(L,R) \in \mathcal{C}} |(L \setminus R) \cap (T \setminus \Gamma)|$$

We will see in a moment that $E[\alpha_j] \geq \Omega(\ell_j)$. Providing this, the number of new sink vertices added to Γ in Line 12 at the end of iteration j is at least $\Omega(\ell_j/\log n)$ in expectation, because each vertex in $T \setminus \Gamma$ will be counted at most $O(\log n)$ times in α_j (at most once in each collection $\mathcal{C}$ because Theorem A.2 return a collection of disjoint cuts). Note that these new sink vertices added into Γ are all $\widetilde{T}_j$-vertices (since all cuts in $\mathcal{K}'$ have size k'), we conclude that $E[\ell_{j+1}] \leq \ell_j - \Omega(\ell_j/\log n)$.

It remains to show $E[\alpha_j] \geq \Omega(\ell_j)$. Fix a sink vertex $t \in \widetilde{T}_j$, and let (L^*, R^*) denote the t-s mincut that minimizes $|L^* \setminus R^*|$. By the submodularity of vertex cuts, we have Observation A.5.

Observation A.5. *Any t-s mincut (L, R) has $L \setminus R \supseteq L^* \setminus R^*$*

We consider the size scale $r_t = 2^{\lfloor |(L^* \setminus R^*) \cap \widetilde{T}_j| \rfloor}$ (i.e. r_t is roughly the size of $(L^* \setminus R^*) \cap \widetilde{T}_j$). Regarding the sample process in Line 6 of this size scale r_t, we define an event Q_t in which t is sampled but the other sink vertices in $(L^* \setminus R^*) \cap \widetilde{T}_j$ are not. Trivially Q_t happens with probability

$$\frac{1}{r_t} \cdot (1 - \frac{1}{r_t})^{|(L^* \setminus R^*) \cap \widetilde{T}_j| - 1} = \Omega(1/r_t) = \Omega(1/|(L^* \setminus R^*) \cap \widetilde{T}_j|).$$

Note that $(L^* \setminus R^*) \cap (T \setminus \Gamma) = (L^* \setminus R^*) \cap \widetilde{T}_j$ because each $t \in (L^* \setminus R^*) \cap (T \setminus \Gamma)$ has $\lambda_{\widetilde{G}}(t, s)$ by the induction hypothesis and the fact that (L^*, R^*) has size k'. Providing this, whenever Q_t happens, we must have $(L^* \setminus R^*) \cap T' = \{t\}$, which means (L^*, R^*) is a t-$(W \setminus \{t\})$ cut. Therefore, the isolating cut $(L, R) \in \mathcal{C}_{\text{iso}}$ of $t \in T' \subseteq W$ computed in Line 8 has size exactly k', which means in Line 9, (L, R) will be added into $\mathcal{C}'$. By Observation A.5,

$$|(L \setminus R) \cap (T \setminus \Gamma)| \geq |(L^* \setminus R^*) \cap (T \setminus \Gamma)| = |(L^* \setminus R^*) \cap \widetilde{T}_j|.$$

In summary, with probability $\Omega(1/|(L^* \setminus R^*) \cap \widetilde{T}_j|)$, the cut (L, R) will contribute $|(L \setminus R) \cap (T \setminus \Gamma)| \geq |(L^* \setminus R^*) \cap \widetilde{T}_j|$ to α_j. In other words, (L, R) contributes $\Omega(1)$ to $E[\alpha_j]$. Repeating this argument for all $t \in \widetilde{T}_j$, we can conclude that $E[\alpha_j] \geq \Omega(|\widetilde{T}_j|) = \Omega(\ell_j)$. □

□

We are ready to verify that $\mathcal{K}$ is a mincut cover with respect to s and T.

- To prove property 1, we argue that at phase k', each cut (L, R) in each collection $\mathcal{C}$ (obtained in Line 9) is a t-s mincut for some $t \in T^*$.

  Recall that the cut (L, R) we add into $\mathcal{C}$ has size k' and is an isolating cut of some sink $t \in T' \subseteq W$. Observe that $t \in T' \subseteq T \setminus \Gamma$ from Line 6. Combining it with Lemma A.3 and the fact that Γ only grows, we have $\lambda_{\widetilde{G}}(t, s) \geq k'$. Because (L, R) has size k', (L, R) is a t-s mincut and $t \in T^*$.

- Consider property 2. By Lemma A.3, at the end of the algorithm, $\Gamma = T^*$. Property 2 follows the update rule of Γ (i.e. Line 12).

- Property 3 is because Theorem A.2 returns a collection of disjoint cuts.

Lastly, the width of $\mathcal{K}$ is bounded by the number of inner loops, which is at most $O(k \log^3 n)$.

Regarding the running time, the bottleneck is calling Theorem A.2. We invoke Theorem A.2 $O(k \log^3 n)$ times, each of which takes $m^{1+o(1)}$ time. Hence the total running time is $O(km^{1+o(1)})$. □

B Omitted Proof

Proof of Observation 5.14. Our strategy is to improve (L, R) to (L', R') with $(L' \setminus R') \cap T = (L \setminus R) \cap T$ with additional guarantees: (1) each connected component C of $G[L' \setminus R']$ intersects T, and (2) $N_G(L' \setminus R') = L' \cap R'$. As we show next, this is indeed possible. Then for each $v \in L' \cap R' \cap T$, $v \in N_G(C)$ for some component C of $G[L' \setminus R']$. Furthermore, because C intersects T, we have $v \in N_H(C \cap T)$, which implies $v \in N_H((L' \setminus R') \cap T)$ and we are done.

We can obtain (L', R') with the two additional guarantees as follows. First, let Γ denote the union of all components in $G[L \setminus R]$ that are disjoint from T. The cut (L_1, R_1) with $L_1 = L \setminus \Gamma$ and $R_1 = R \cup \Gamma$ is obviously a witness satisfying the first extra property, and we have $(L_1 \setminus R_1) \cap T = (L \setminus R) \cap T$. Next, consider the cut (L_2, R_2) with $L_2 = (L_1 \setminus R_1) \cup N(L_1 \setminus R_1)$ and $R_2 = V(G) \setminus (L_1 \setminus R_1)$. It is still a witness satisfying both the first and second extra properties, and trivially $(L_2 \setminus R_2) \cap T = (L_1 \setminus R_1) \cap T$. Finally, take (L_2, R_2) as the desired (L', R'). □

C Connection to Tree Decomposition with Bounded Width

Here, we argue that unbreakable decomposition generalizes tree decomposition with bounded width to general graphs.

By definition, a graph G has bounded treewidth if and only if there exists a tree decomposition of G with bounded width, i.e., all bags have bounded size. So we cannot hope for this decomposition to exist in an arbitrary graphs. In contrast, for any $k \leq q$, (q, k)-unbreakable decomposition exists for every graph [CKL+20].

We will show that, on a graph with bounded treewidth, every unbreakable decomposition is precisely a tree decomposition with bounded width. To prove this, we need the following well-known fact (see e.g. [Ree97]).

Proposition C.1. *Let G be a graph with treewidth at most k. Let X be any vertex set of size s. Then, there exists a vertex cut (L, R) of size k where $|L \cap X|, |R \cap X| > s/3$.*

Now, we formally prove our claim.

Corollary C.2. *Let G be a graph with treewidth at most k. Let (T, β) be a (q, k)-unbreakable decomposition of G. Then, each bag of T has size $|\beta(t)| \leq 3q$ for all tree node t. That is, the width of T must be at most $3q$.*

Proof. Suppose for contradiction that there is a tree node t whose bag $\beta(t)$ contains more than $3q$ vertices. By Proposition C.1, there is a vertex cut (L, R) of size k where $|L \cap \beta(t)|, |R \cap \beta(t)| > q$. This contradicts that $\beta(t)$ is (q, k)-unbreakable. □