


# Parameterized Complexity of Fair Bisection



## FPT-Approximation meets Unbreakability

Tanmay Inamdar  

University of Bergen, Norway




Daniel Lokshtanov  

University of California Santa Barbara, CA, USA

Saket Saurabh   

The Institute of Mathematical Sciences, HBNI, Chennai, India

University of Bergen, Norway

Vaishali Surianarayanan   

University of California Santa Barbara, CA, USA

---

### Abstract

In the Minimum Bisection problem input is a graph  $G$  and the goal is to partition the vertex set into two parts  $A$  and  $B$ , such that  $||A| - |B|| \leq 1$  and the number  $k$  of edges between  $A$  and  $B$  is minimized. The problem is known to be NP-hard, and assuming the Unique Games Conjecture even NP-hard to approximate within a constant factor [Khot and Vishnoi, J.ACM'15]. On the other hand, a  $\mathcal{O}(\log n)$ -approximation algorithm [Räcke, STOC'08] and a parameterized algorithm [Cygan et al., ACM Transactions on Algorithms'20] running in time  $k^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$  is known.

The Minimum Bisection problem can be viewed as a clustering problem where edges represent similarity and the task is to partition the vertices into two equally sized clusters while minimizing the number of pairs of similar objects that end up in different clusters. Motivated by a number of egregious examples of unfair bias in AI systems, many fundamental clustering problems have been revisited and re-formulated to incorporate fairness constraints. In this paper we initiate the study of the Minimum Bisection problem with fairness constraints. Here the input is a graph  $G$ , positive integers  $c$  and  $k$ , a function  $\chi : V(G) \rightarrow \{1, \dots, c\}$  that assigns a color  $\chi(v)$  to each vertex  $v$  in  $G$ , and  $c$  integers  $r_1, r_2, \dots, r_c$ . The goal is to partition the vertex set of  $G$  into two almost-equal sized parts  $A$  and  $B$  with at most  $k$  edges between them, such that for each color  $i \in \{1, \dots, c\}$ ,  $A$  has exactly  $r_i$  vertices of color  $i$ . Each color class corresponds to a group which we require the partition  $(A, B)$  to treat fairly, and the constraints that  $A$  has exactly  $r_i$  vertices of color  $i$  can be used to encode that no group is over- or under-represented in either of the two clusters.

We first show that introducing fairness constraints appears to make the Minimum Bisection problem qualitatively harder. Specifically we show that unless  $\text{FPT}=\text{W}[1]$  the problem admits no  $f(c)n^{\mathcal{O}(1)}$  time algorithm even when  $k = 0$ . On the other hand, our main technical contribution shows that is that this hardness result is simply a consequence of the very strict requirement that each color class  $i$  has *exactly*  $r_i$  vertices in  $A$ . In particular we give an  $f(k, c, \epsilon)n^{\mathcal{O}(1)}$  time algorithm that finds a balanced partition  $(A, B)$  with at most  $k$  edges between them, such that for each color  $i \in [c]$ , there are at most  $(1 \pm \epsilon)r_i$  vertices of color  $i$  in  $A$ .

Our approximation algorithm is best viewed as a proof of concept that the technique introduced by [Lampis, ICALP'18] for obtaining FPT-approximation algorithms for problems of bounded tree-width or clique-width can be efficiently exploited even on graphs of unbounded width. The key insight is that the technique of Lampis is applicable on tree decompositions with unbreakable bags (as introduced in [Cygan et al., SIAM Journal on Computing'14]). An important ingredient of our approximation scheme is a combinatorial result that may be of independent interest, namely that for every  $k$ , every graph  $G$  admits a tree decomposition with adhesions of size at most  $\mathcal{O}(k)$ , unbreakable bags, and logarithmic depth.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Parameterized complexity and exact algorithms



© Tanmay Inamdar, Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan; licensed under Creative Commons License CC-BY 4.0

31st Annual European Symposium on Algorithms (ESA 2023).

Editors: Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman; Article No. 63; pp. 63:1–63:17



Leibniz International Proceedings in Informatics

LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Keywords and phrases** FPT Approximation, Minimum Bisection, Unbreakable Tree Decomposition, Treewidth

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2023.63

**Related Version** *Full Version:* <https://arxiv.org/abs/2308.10657>



**Funding** *Tanmay Inamdar:* Supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 819416).

*Daniel Lokshтанov:* NSF award CCF-2008838.



*Saket Saurabh:* European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 819416), and Swarnajayanti Fellowship grant DST/SJF/MSA01/2017-18.

*Vaishali Surianarayanan:* NSF award CCF-2008838.

## 1 Introduction

Clustering is one of the most fundamental problems in computer science. In a clustering problem, we are typically interested in dividing the given collection of data points into a group of *clusters*, such that the set of data points belonging to each cluster are more “similar” to each other, as compared to the points belonging to other clusters. Depending on the specific setting and application, there are a number of ways to model this abstract task of clustering as a concrete mathematical problem. We refer the reader to surveys such as [30, 27, 3] for a detailed background and literature on the topic.

In one such model of clustering, the input is represented as a simple, undirected graph, and the existence of an edge between a pair of vertices denotes that the two vertices are related to, or similar to, each other. For example, this is how one models social networks as graphs [25] – the set of vertices corresponds to people, and an edge represents that the two people are friends with each other. In this setting, the classical MINIMUM BISECTION problem can be thought of as a clustering problem [31, 7] – we are interested in finding two size-balanced clusters of vertices, such that the number of edges going across the two clusters is minimized. More formally, in MINIMUM BISECTION problem, we are given a graph  $G = (V, E)$  on  $n$  vertices, and a non-negative integer  $k$ , and the goal is to determine whether there exists a balanced edge cut  $(A, B)$  of order  $k$ . Here, an *edge cut*  $(A, B)$  is a partition of  $V(G)$  into two non-empty subsets  $A$  and  $B$ , an edge cut is *balanced* if  $||A| - |B|| \leq 1$ , and the *order* of the cut  $(A, B)$  is the number of edges with one endpoint in  $A$  and the other in  $B$ . The NP-completeness of MINIMUM BISECTION has long been known [13], and it is extensively studied from the perspective of approximation and parameterized algorithms. MINIMUM BISECTION admits a logarithmic approximation in polynomial time [26], and it is hard to approximate within any constant factor, assuming the Unique Games Conjecture [19]. In the realm of Parameterized Algorithms, one can solve the problem exactly in time  $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ , i.e., it is Fixed-Parameter Tractable (FPT) parameterized by  $k$  [10, 9].

More recently, the notion of *fairness* has gained prominence in the literature of clustering algorithms – and algorithm design in general. This is motivated from the fact that, often the real-life data reflects unconscious biases, and unless the algorithm is explicitly required to counteract these biases, the output of the algorithm may have real-life consequences that are *unfair* (see, e.g., [15, 24, 11]). Researchers have proposed different models of fairness for the traditional center-based clustering problems, such as  $k$ -MEDIAN/MEANS/CENTER. These models of fairness can be broadly classified into two types – *individual fairness*, and *group fairness*. At a high level, individual fairness requires that the solution treats each of

the individuals (a point) in a fair way, e.g., every point has a cluster-center “nearby” [6]. On the other hand, in the group fairness setting, the set of points is typically divided into multiple colors, where each color represents, say a particular demographic (such as gender, ethnicity etc.). In this setting, the fairness constraints are represented in terms of the colors as a group. There are multiple notions of group fairness (see, e.g., [2, 6, 12, 22, 14, 18]), but to the specific interest to us is the *color-balanced clustering* model, studied in [28, 17, 1]. Roughly speaking, in this setting we want the “local proportions” of all colors in every cluster to be approximately equal to their “global proportions”. Inspired from this *color-balanced* notion of fairness, study the following *fair* version of MINIMUM BISECTION.

In this problem formulation the color classes  $i \in \{1, \dots, c\}$  are protected groups which are required to be treated fairly by the clustering algorithm. The imposed fairness constraint for group  $i$  is that, in the edge cut  $(A, B)$ , the set  $A$  contains precisely  $r_i$  vertices colored  $i$ .

#### FAIR BISECTION

**Input:** An instance  $(G, c, k, r^\circ, \chi)$ , where

- $G$  is an unweighted graph
- $c$  and  $k$  are positive integers
- $\chi : V(G) \rightarrow c$  is a coloring function on  $V(G)$  using at most  $c$  colors
- $r^\circ = (r_1, \dots, r_c)$  is a  $c$  length tuple of positive integers

**Question:** Does there exist an edge cut  $(A, B)$  of  $G$  of order at most  $k$  having exactly  $r_i$  vertices of color  $i$  in  $A$  for each  $i \in [c]$ .

We will say that an edge cut that satisfies the fairness constraints imposed by the tuple  $r^\circ$  is  $r^\circ$ -*fair*.

Thus, when  $r_i$  is set to be precisely half of the number  $c_i$  of vertices colored  $i$  an  $r^\circ$ -fair edge cut must evenly split each color class across the two sides  $A$  and  $B$ .

## Our Results

It is quite easy to see that the existing parameterized algorithms [9, 10] for MINIMUM BISECTION directly generalize to a  $n^{\mathcal{O}(c)}k^{\mathcal{O}(k)}$  time algorithm for FAIR BISECTION<sup>1</sup>. Therefore, the first natural question is whether it is possible to eliminate the dependence on  $c$  in the exponent of  $n$  in the running time. Our first result (Theorem 20) is that, assuming  $\text{FPT} \neq \text{W}[1]$ , an  $f(c)n^{\mathcal{O}(1)}$  time algorithm is not possible even when  $k = 0$ . In fact, this hardness result holds even in the special case where the vertices of each color are required to be evenly split across both partitions (in particular, when  $2r_i = c_i$  for every  $i$ ).

Our main technical contribution (Theorem 11) is to show that this hardness result is quite brittle. Indeed, the requirement that each color class  $i$  have *exactly*  $r_i$  vertices in  $A$  is probably much too strong in the color-balanced fairness setting. We are satisfied even if the number of vertices of each color class is sufficiently close to the desired target number. We will say that an edge cut  $(A, B)$  is  $(\epsilon, r^\circ)$ -*fair* if  $A$  contains no more than  $r_i(1 + \epsilon)$  of vertices colored  $i$  and  $B$  contains no more than  $(c_i - r_i)(1 + \epsilon)$  vertices colored  $i$ . We show (in Theorem 11) that there exists an algorithm that takes as input an instance  $(G, c, k, r^\circ, \chi)$ , together with an  $\epsilon > 0$ , runs in time  $f(\epsilon, k, c)n^{\mathcal{O}(1)}$ , and if  $G$  has a  $r^\circ$ -fair edge cut  $(\hat{A}, \hat{B})$  of order at most  $k$  then the algorithm produces a  $(\epsilon, r^\circ)$ -fair edge cut  $(A, B)$  of order at most  $k$ .

<sup>1</sup> A formal proof of this claim is a corollary of our Theorem 11.

## Our Methods

The hardness result of Theorem 20 is a fairly straightforward parameterized reduction from MULTI-DIMENSIONAL SUBSET SUM parameterized by the dimension<sup>2</sup>, whose main purpose is to put the parameterized approximation scheme of Theorem 11 in context. We only discuss here the methods in the proof of Theorem 11.

At a *very* high level the algorithm of Theorem 11 is the combination of two well-known techniques in parameterized algorithms: dynamic programming over tree decompositions with unbreakable bags (introduced by Cygan et al. [10]), and the geometric rounding technique of Lampis [20] for parameterized approximation schemes for problems on graphs of bounded tree-width or clique-width. The conceptual novelty in (and perhaps the most interesting technical aspect of) our work is to realize that Lampis’ technique can be applied even to dynamic programming algorithms over tree decompositions with unbounded width to yield approximation schemes for parameterized problems on general graphs. Executing on this vision requires a few non-trivial technical insights, which we will shortly highlight. However, to describe these technical insights in more detail we first give a brief description of the two techniques that we combine.

## Lampis’ Geometric Rounding Technique

We first discuss how the technique of Lampis [20] applies to tree decompositions of bounded width. A *tree decomposition* of a graph  $G$  is a pair  $(T, \beta)$  where  $T$  is a tree and  $\beta$  is a function that assigns to each vertex  $t \in V(T)$  a vertex set  $\beta(t) \subseteq V(G)$  (called a *bag*) in  $G$ . To be a tree decomposition the pair  $(T, \beta)$  must satisfy the tree-decomposition axioms: (i) for every  $v \in V(G)$  the set  $\{t \in V(T) : v \in \beta(t)\}$  induces a non-empty and connected subgraph of  $T$ , and (ii) for every edge  $uv \in E(G)$  there exists a  $t \in V(T)$  such that  $\{u, v\} \subseteq \beta(t)$ . The *width* (or tree-width) of a decomposition  $(T, \beta)$  is defined as  $\max_{t \in V(T)} |\beta(t)| - 1$ .

Roughly speaking, Lampis’ technique considers dynamic programming (DP) algorithms over a tree decomposition  $(T, \beta)$  of  $G$  of width  $k$ . In such an algorithm there is a DP-table for every node  $t$  of the decomposition tree, and suppose that the entries in these tables are indexed by vectors in  $\{1, 2, \dots, n\}^d$  (for some integer  $d$ ), where  $n$  is the number of vertices of  $G$ . To decrease the size of the DP tables and thereby also the running time of the algorithm, one “sparsifies” the DP table to only consider entries in  $S^d$ , where  $S = \{\lfloor (1 + \delta)^i \rfloor : i \geq 0\}$ . This makes the size of the DP table upper bounded by  $(\log_{1+\delta} n)^{\mathcal{O}(d)}$ , at the cost of introducing a multiplicative error of  $(1 + \delta)$  in every round of the DP algorithm (since now vectors in  $\{1, 2, \dots, n\}^d$  are “approximated” by their closest vector in  $S^d$ ). If the decomposition tree  $T$  has depth  $\mathcal{O}(\log n)$  the dynamic program only needs  $\mathcal{O}(\log n)$  rounds, and so the total error of the algorithm is a multiplicative factor of  $(1 + \delta)^{\mathcal{O}(\log n)}$ . Setting  $\delta = \epsilon / \log^2 n$  gives the desired trade-off between DP table size (and therefore running time) and accuracy. Luckily, every tree decomposition of width  $k$  can be turned into a tree decomposition of width at most  $3k + 2$  and depth  $\mathcal{O}(\log n)$  [4] and so this approach works on all graphs of tree-width  $k$ .

## Tree Decompositions with Unbreakable Bags

We now turn to the technique of Cygan et al. [10] for MINIMUM BISECTION, namely dynamic programming over tree decompositions with small adhesions and unbreakable bags. We again need to define a few technical terms. An *adhesion* of a tree decomposition  $(T, \beta)$  of a graph

<sup>2</sup> The hardness of MULTI-DIMENSIONAL SUBSET SUM parameterized by the dimension is folklore, but we were unable to find a reference, so for completeness we provide a proof.

$G$  is a set  $\beta(u) \cap \beta(v)$  for an edge  $uv \in E(T)$ . The *adhesion size* of a tree-decomposition is just the maximum size of an adhesion of the decomposition. A tree decomposition  $(T, \beta)$  is said to have  $(q, k)$ -*unbreakable bags* if for every bag  $\beta(t)$  of the decomposition and every edge-cut  $(A, B)$  of order at most  $k$  in  $G$  it holds that  $\min(|A \cap \beta(t)|, |B \cap \beta(t)|) \leq q$ .

The main engine behind the algorithm of Cygan et al. [10] (see also [9]) is a structural theorem that for every graph  $G$  and integer  $k$  there exists a tree decomposition  $(T, \beta)$  of  $G$  with adhesion size at most  $k$  and  $(k + 1, k)$ -unbreakable bags. This is coupled with an observation that even though this tree decomposition might have unbounded tree-width, we can still do dynamic programming over this tree decomposition, keeping a DP table for every *adhesion* of the tree decomposition, rather than for every bag. However, while tree-width based DP algorithms utilize a simple recurrence to calculate the DP table at a bag from the tables of its children, Cygan et al. [10] need to turn to a clever “randomized contraction” (see [8]) based algorithm to compute the DP table for an adhesion from the DP tables of its children.

### Combining Tree Decompositions with Unbreakable Bags and Geometric Rounding

As we mentioned earlier, the technique of Cygan et al. [10] for MINIMUM BISECTION generalizes in a relatively straightforward way, to give a  $f(k)n^{\mathcal{O}(c)}$  time algorithm for FAIR BISECTION. Here we do dynamic programming over the tree decomposition of  $G$  with adhesions of size  $k$  and  $(k + 1, k)$ -unbreakable bags. We have a DP table for every adhesion that is indexed by a vector in  $[n]^c$  (this vector describes partial solutions, where the  $i^{\text{th}}$  element of the vector is the number of vertices of color  $i$  that have so far been put on the  $A$  side in this partial solution).

We want to apply Lampis’ geometric rounding technique and “sparsify” the DP table to only consider entries in  $S^c$ , where  $S = \{\lfloor (1 + \delta)^i \rfloor : i \geq 0\}$ . There are a few technical obstacles to realizing this plan, that we overcome. The most important one of them is that the depth reduction theorem of Bodlaender and Hagerup [4] only applies to tree decompositions of bounded width, therefore it is not immediate how to obtain a tree decomposition with small adhesions, unbreakable bags and logarithmic depth. A closer inspection of the proof sketch of Bodlaender and Hagerup [4] reveals that a tree decomposition with adhesions of size  $k$  and  $(k + 1, k)$ -unbreakable bags can be turned into a tree decomposition with adhesions of size  $\mathcal{O}(k)$ , and logarithmic depth, such that each bag of the new decomposition is the union of a constant number of bags of the old one (the bags in this new decomposition do *not* need to themselves be unbreakable). Nevertheless we prove that some careful modifications to this tree decomposition are sufficient to obtain a tree decomposition with adhesions of size  $\mathcal{O}(k)$ , logarithmic depth, and  $(\mathcal{O}(k), k)$ -unbreakable bags (see Theorem 9). We believe that Theorem 9 will be a useful tool for future applications of Lampis’ geometric rounding technique to tree decompositions with unbreakable bags.

### Organization of the Paper

We begin by defining the basic notions on graphs and tree decompositions in Section 2. In Section 3, we prove Corollary 10 that shows how to obtain logarithmic-depth unbreakable tree decompositions. Then, in Section 4, we use such a tree decomposition to design our exact and approximate algorithms. In Section 5, we sketch the proof of our hardness result, which shows that FAIR BISECTION is  $W[1]$ -hard parameterized by  $c$  even when  $k = 0$ . Finally, in Section 6, we give concluding remarks and future directions. Proofs marked with  $*$  can be found in the full version of the paper.

## 2 Preliminaries

For an integer  $k$ , we denote the set  $\{1, 2, \dots, k\}$  by  $[k]$ . For a graph  $G$ , an *edge cut* is a pair  $A, B \subseteq V(G)$  such that  $A \cup B = V(G)$  and  $A \cap B = \emptyset$ . The order of an edge cut  $(A, B)$  is  $|E(A, B)|$ , that is, the number of edges with one endpoint in  $A$  and the other in  $B$ . For a subset  $X \subseteq V(G)$ , let  $G \setminus X$  denote the graph  $G[V(G) \setminus X]$ . For an edge cut  $(A, B)$ , and a subset  $X \subseteq V(G)$ , the cut *induced* on  $X$  by  $(A, B)$  is  $(A \cap X, B \cap X)$ .

► **Definition 1 (unbreakability).** A set  $X \subseteq V(G)$  is  $(q, s)$ -edge-unbreakable if every edge cut  $(A, B)$  of order at most  $s$  satisfies  $|A \cap X| \leq q$  or  $|B \cap X| \leq q$ .

For a rooted tree  $T$  and vertex  $t \in V(T)$ , we denote by  $T_t$  the subtree of  $T$  rooted at  $t$ . For a rooted tree  $T$  and a non-root vertex  $t \in V(T)$ , we denote the parent of  $t$  by  $\mathcal{P}(t)$ . The depth of a tree  $T_t$  is the maximum length of a  $t$  to leaf path in  $T_t$ . For a node  $t$ , we denote  $\text{ht}_T(t)$  to be the depth of the subtree  $T_t$  rooted at  $t$  in  $T$ .

Consider a tree decomposition  $(T, \beta)$  of a graph  $G$ . For every  $t \in V(T)$  a set  $\beta(t) \subseteq V(G)$ , is called a *bag*. We can extend the function  $\beta$  to subsets of  $V(T)$  in the natural way: for a subset  $X \subseteq V(T)$ ,  $\beta(X) := \bigcup_{x \in X} \beta(x)$ . Another important notion that we need is of tree decomposition where bags are “highly connected”, i.e., unbreakable. For a rooted tree  $T$  and vertex  $v \in V(T)$  we denote by  $T_v$  the subtree of  $T$  rooted at  $v$ . We refer to the vertices of  $T$  as nodes.

For  $s, t \in V(T)$  we say that  $s$  is a *descendant* of  $t$  or that  $t$  is an *ancestor* of  $s$  if  $t$  lies on the unique path from  $s$  to the root; note that a node is both an ancestor and a descendant of itself. By  $\text{child}(t)$ , we denote the set of children of  $t$  in  $T$ . For any  $X \subseteq V(T)$ , define  $G_X := G[\bigcup_{t \in X} \beta(V(T_t))]$ .

We define an *adhesion* of an edge  $e = (t, t_0) \in E(T)$  to be the set  $\sigma(e) := \beta(t) \cap \beta(t_0)$ , and an adhesion of  $t \in V(T)$  to be  $\sigma(t) := \sigma(t, \mathcal{P}(t))$ , or  $\sigma(t) = \emptyset$  if the parent of  $t$  does not exist, i.e., when  $t$  is the root of  $T$ . We define the following notation for convenience:

$$\gamma(t) := \bigcup_{s: \text{ descendant of } t} \beta(s)$$

$$\alpha(t) := \gamma(t) \setminus \sigma(t), \quad G_t := G[\gamma(t)] - E(G[\sigma(t)]).$$

We say that a rooted tree decomposition  $(T, \beta)$  of  $G$  is *compact* if for every node  $t \in V(T)$  for which  $\alpha(t) \neq \emptyset$  we have that  $G[\alpha(t)]$  is connected and  $N_G(\alpha(t)) = \sigma(t)$ .

## 3 Obtaining a Low Depth Unbreakable Tree Decomposition

In this section we show that there exists a tree decomposition that has low (i.e.,  $\mathcal{O}(\log n)$ ) depth, small-size (i.e.,  $\mathcal{O}(k)$ ) adhesions, and  $(\mathcal{O}(k), k)$ -unbreakable bags. To this end, we design a polynomial-time algorithm that, given a tree decomposition with small adhesions and unbreakable bags, produces a tree decomposition with the aforementioned properties. In the next section, we design a dynamic programming algorithm over such a low depth decomposition to obtain an FPT approximation for FAIR BISECTION.

In our algorithm, we use the notion of a *tree partition* of a graph, which, informally, captures the “tree-likeness” of a graph. Tree partitions were introduced by [29, 16], and are easy to define.

► **Definition 2 (Tree Partition).** A *tree partition* of a graph  $G$  is a pair  $(\mathcal{T}, \tau)$  where  $\mathcal{T}$  is a tree and  $\tau : V(G) \rightarrow V(\mathcal{T})$  is a function from  $V(G)$  to  $V(\mathcal{T})$  such that for each  $e = (u, v) \in E(G)$  either  $\tau(u) = \tau(v)$  or  $(\tau(u), \tau(v)) \in E(\mathcal{T})$ . A *rooted tree partition*  $(\mathcal{T}, \tau)$  with root  $r$  is the tree partition  $(\mathcal{T}, \tau)$  where the tree  $\mathcal{T}$  is a rooted tree with root  $r$ .

We remark that we use calligraphic font ( $\mathcal{T}$ ) to denote trees corresponding to Tree Partitions to easily distinguish them from graphs that are trees. Observe that for a tree  $T$ , the pair  $(\mathcal{T} = T, \tau)$  where  $\tau(v) = v$  for each  $v \in T$  is a trivial tree partition of  $T$ . For our result we only use tree partitions of trees. Given a tree decomposition  $(T, \beta)$  with small adhesions and unbreakable bags, our goal in this section is to use  $(T, \beta)$  to obtain a tree decomposition of bounded height without blowing up the adhesion size and unbreakability guarantees too much. For this we first find a tree partition  $(\mathcal{T}, \tau)$  of  $T$  that satisfies additional properties, such as logarithmic depth and for each  $t \in V(\mathcal{T})$ , it holds that  $|\tau^{-1}(t)| \leq 4$ . Using this tree partition, we obtain a tree decomposition  $(\mathcal{T}, \beta_1)$  whose underlying tree is  $\mathcal{T}$ , and each bag  $\beta_1(t)$ ,  $t \in V(\mathcal{T})$  is a union of at most 4 bags of  $(T, \beta)$ ;  $\beta_1(t) = \bigcup_{x \in \tau^{-1}(t)} \beta(x)$ . This tree decomposition already has bounded height, small adhesion size and each bag is a union of at most four unbreakable bags of  $(T, \beta)$ . From here with some extra work we obtain a tree decomposition with unbreakable bags as well. For this we use other properties of  $(\mathcal{T}, \tau)$  to modify  $(\mathcal{T}, \beta_1)$  to obtain our desired tree decomposition. As outlined above, for our result we need tree partitions of a tree satisfying some properties. We now define such tree partitions below and show how to find one in polynomial time.

► **Definition 3** (Nice Tree Partition). *A tree partition  $(\mathcal{T}, \tau)$  of a tree  $T$  is said to be a nice tree partition if it satisfies the following properties:*

1.  $\mathcal{T}$  has depth at most  $\lceil \log_2 |V(T)| \rceil$
2. for each  $t \in V(\mathcal{T})$ ,  $1 < |\tau^{-1}(t)| \leq 4$ .
3. for each  $t \in V(\mathcal{T})$ ,  $T[V_t]$  is a subtree of  $T$ , where  $V_t = \bigcup_{x \in V(\mathcal{T}_t)} \tau^{-1}(x)$ .

We now show how to find a nice tree partition of a tree in polynomial time. For this we use a recursive procedure. The core idea in each recursive step is to map a balanced separator  $b$  of the tree to the root of the tree partition. To ensure the connectivity properties of a tree partition, we have a set  $M$  of marked vertices in the tree that are always mapped to the root of the tree partition in addition to  $b$ . Then for each connected component in the forest obtained by removing  $M \cup \{b\}$  from the tree, we mark new vertices and recurse. We need some extra work to make sure that every node in the tree partition is mapped to by only a constant number of nodes in the tree. For this we ensure that in each recursive call we mark only a few ( $\leq 2$ ) new vertices.

► **Lemma 4.** *Given a tree  $T$  on  $n$  vertices with root  $r$ , one can in polynomial time compute a rooted nice tree partition  $(\mathcal{T}, \tau)$  of  $T$  with root  $r_{\mathcal{T}}$  such that  $\tau(r) = r_{\mathcal{T}}$ .*

**Proof.** We now design a procedure `FindBalancedTP` that takes as an argument a tree  $T'$ , and a non-empty set  $M \subseteq V(T')$  of size at most 2, and returns a rooted nice tree partition  $(\mathcal{T}', \tau')$  of  $T'$  with root  $r'$ , such that  $M \subseteq \tau'^{-1}(r')$ . We will invoke this procedure on the input tree  $T$  with  $M = \{r\}$ , where  $r$  is the root of  $T$  to obtain a rooted nice tree partition  $(\mathcal{T}, \tau)$  of  $T$ .

In the procedure `FindBalancedTP`( $T', M$ ) we carry out the following steps:

- We find a balanced bisector  $b$  of  $T'$  and initialize  $M' = M \cup \{b\}$ .
- If all vertices in  $M'$  do not lie on a path in  $T'$ , we add an extra vertex  $x$  to  $M'$ . Let  $x$  be the last common vertex on the path from  $m_1$  to  $m_2$  and the path from  $m_1$  to  $b$  in  $T$ . Modify  $M' = M' \cup \{x\}$ .
- For each tree  $H$  in the forest  $T' \setminus M'$ , we recursively call `FindBalancedTP`( $H, M_H$ ) where  $M_H = N_{T'}(M') \cap V(H)$  is the set of neighbors of vertices in  $M'$  in  $H$ . Let  $(\mathcal{H}, \tau_H)$  be the tree partition returned by this procedure call.

- We now construct a tree partition  $(\mathcal{T}', \tau')$  with root  $r'$ . We assign  $\tau'^{-1}(r') = M'$ . Then for each tree  $H$  in the forest  $T' \setminus M'$ , we make  $\mathcal{H}$  a subtree of  $\mathcal{T}'$  by attaching the root of  $\mathcal{H}$  as a child to  $r'$ . Further for each  $t \in V(\mathcal{H})$  we assign  $\tau'^{-1}(t) = \tau_H(t)$ .
- We return  $(\mathcal{T}', \tau')$ .

We now prove that for any tree  $T'$  with root  $r'$ , and any non-empty subset  $M \subseteq V(T')$  with  $0 < |M| \leq 2$ , the procedure **FindBalancedTP** $(T', M)$  returns a rooted nice tree partition  $(\mathcal{T}', \tau')$  of  $T'$  with root  $r'$  such that  $M \subseteq \tau'^{-1}(r')$ . The proof is by induction.

*Base Case*  $|V(T')| = 1$  or  $V(T') = M'$ : In this case  $V(\mathcal{T}') = \{r'\}$  and  $\tau(r') = M'$ . Observe that  $0 < |M'| \leq 4$ . This is because the procedure is called with a non-empty set  $M$  of size at most two. Then, the procedure initializes  $M' = M$ , and adds at most two other vertices ( $b$  and  $x$ ) in  $V(T')$  to  $M'$ . Thus  $(\mathcal{T}', \tau')$  is a nice tree partition with root  $r'$  and  $M \subseteq \tau'^{-1}(r')$ .

Now we prove the *inductive case* where  $V(T')$  has size  $i$ ,  $i > 1$  and  $V(T') \neq M'$ . For this we assume the inductive hypothesis that the procedure returns a tree partition with the desired properties for all trees  $H$  having less than  $i$  vertices and non-empty sets  $M' \subseteq V(H)$  of size at most two.

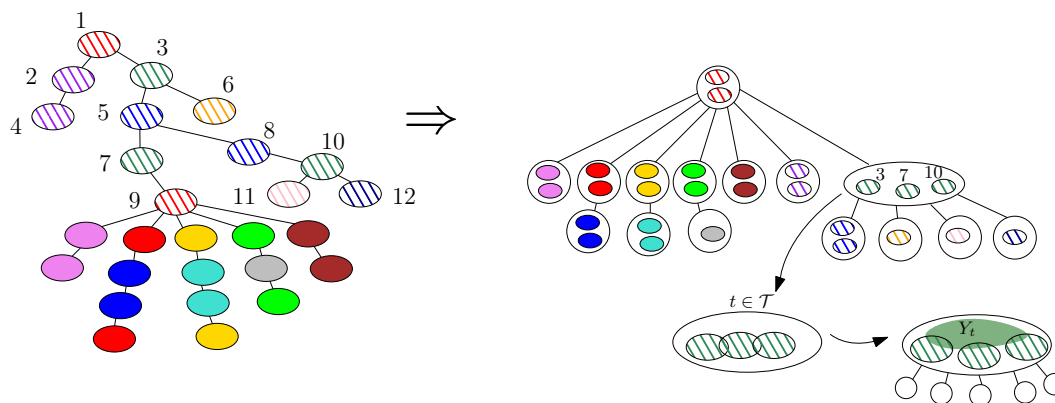
Let  $H$  be a tree in the forest  $T' \setminus M'$ . We now show that  $|V(H)| \leq \lceil |V(T')|/2 \rceil$  and  $1 < |M_H| \leq 2$ . By construction  $M'$  contains the vertex  $b$ , a balanced bisector of  $T'$ . Thus  $V(H)$  has size at most  $\lceil |V(T')|/2 \rceil$ .  $|M_H| > 1$  since  $H$  contains at least one child of  $M'$  since it is a tree in the forest  $T' \setminus M'$ . To show  $|M_H| \leq 2$ , we first show there is a vertex  $s$  in  $M'$  such that in the forest  $T' \setminus \{s\}$  every vertex  $s' \in M' \setminus \{s\}$  is contained in a different tree. If all vertices in  $M \cup \{b\}$  do not lie on a path in  $T'$ , then  $s$  is just the vertex  $x$  we added to  $M'$  in the second step of the procedure. If the vertices of  $M \cup \{b\}$  lie on a path  $P$  in  $T'$  then  $M' = M \cup \{b\}$ . In this case if  $|M'| \leq 2$ , then  $s$  is any vertex in  $M'$ . On the other hand if  $|M'| = 3$ , then  $s$  is the second vertex from  $M'$  in the path  $P$ . Due to the property of  $s$ ,  $H$  may contain a child of  $s$  and a child of one other  $s' \in M' \setminus \{s\}$ . Thus  $|M_H| \leq 2$ .

Since  $H$  is a tree with  $|V(H)| \leq \lceil |V(T')|/2 \rceil$  and  $1 < |M_H| \leq 2$ , by induction the tree partition  $(\mathcal{H}, \tau_H)$  returned by the call to the procedure **FindBalancedTP** $(H, M_H)$  is a nice tree partition with root  $r_H$  and  $M_H \subseteq \tau_H^{-1}(r_H)$ .

We now show that  $(\mathcal{T}', \tau')$  is a rooted tree partition of  $T'$  with root  $r'$ . First we show that each vertex  $v \in \mathcal{T}'$  is mapped to exactly one vertex  $t \in \mathcal{T}'$  by  $\tau'$ . If  $v \in M'$ , then  $\tau(v)$  is mapped to  $r'$ . If  $v \in H$ ,  $H \in T' \setminus M'$ , then since  $(\mathcal{H}, \tau_H)$  is a rooted tree partition of  $H$ ,  $\tau(v) = \tau_H(v)$  by construction. Next we show that each edge  $(x, y) \in T'$  satisfies either  $\tau'(x) = \tau'(y)$  or  $(\tau'(x), \tau'(y)) \in E(\mathcal{T}')$ , by considering three cases. (i) If  $x, y \in M'$ , then this is trivially true. (ii) If  $x, y \notin M'$  then  $x, y$  must belong to some tree  $H \in T' \setminus M'$  and thus by induction  $(\tau'(x) = \tau_H(x), \tau'(y) = \tau_H(y)) \in E(\mathcal{T}')$ . (iii) If  $x \in M'$  and  $y \notin M'$ , by construction,  $\tau(x) = r'$  and  $y \in M_H$  for some  $H \in T' \setminus M'$ . Since  $M_H \subseteq \tau_H^{-1}(r_H)$  and  $r_H$  is a child of  $r'$  in  $\mathcal{T}'$ ,  $(\tau'(x) = r', \tau'(y) = r_H) \in E(\mathcal{T}')$ .

$M \subseteq \tau'^{-1}(r')$  just by construction. We now prove properties (1) – (4) in Definition 3 to show that  $(\mathcal{T}', \tau')$  is a nice tree partition. Recall that for each  $H \in T' \setminus M'$ ,  $H$  is a subtree of  $T'$  with  $r_H$  being a child of  $r'$  in  $\mathcal{T}'$ . Then, since  $|V(H)| \leq \frac{\lceil |V(T')| \rceil}{2}$ , by inductive hypothesis,  $\mathcal{H}$  has depth at most  $\lceil \log_2(|V(T')|/2) \rceil = \lceil \log_2(|V(T')|) \rceil - 1$ . Therefore,  $\mathcal{T}'$  has depth  $\lceil \log_2 |V(T')| \rceil$ , since the addition of the root  $r'$  increases the depth by 1. For each  $t' \in V(\mathcal{T}')$ ,  $1 < |\tau'^{-1}(t')| \leq 4$  since  $1 < |\tau'^{-1}(r')| \leq 4$  and for each  $H \in T' \setminus M'$  and for each  $t \in V(\mathcal{H})$ ,  $1 < |\tau_H^{-1}(t)| \leq 4$ . For each  $t' \in V(\mathcal{T}')$ ,  $T'[V_{t'}]$  is a subtree of  $T'$ , where  $V_{t'} = \bigcup_{x \in V(\mathcal{T}'_{t'})} \tau'^{-1}(x)$  because for each  $H \in T' \setminus M'$  and  $t \in V(\mathcal{H})$ ,  $H[V_t]$  is a subtree of  $H$ , where  $V_t = \bigcup_{x \in V(\mathcal{H}_t)} \tau_H^{-1}(x)$  and  $H$  is subtree of  $T'$ . This completes the proof. ◀





■ **Figure 1** Left: Tree decomposition  $(T, \beta)$  with bags colored for easy understanding. Right: tree decomposition  $(T, \beta_1)$  that is constructed using a tree partition  $(T, \tau)$ . The bags in  $(T, \beta)$  are mapped according to  $\tau$  to  $(T, \beta_1)$ . The bags in  $\tau^{-1}(t)$ ,  $t \in \mathcal{T}$  can overlap as demonstrated by bags 3, 7, 10 but their overlap is small and contained in  $Y_t$ .

Let  $(T, \beta)$  be a rooted tree decomposition of a graph  $G$  with root  $r$ ,  $(q, k)$ -unbreakable bags and adhesions of size at most  $k$ . Further let  $(\mathcal{T}, \tau)$  be a rooted nice tree partition of  $T$  with root  $r_{\mathcal{T}}$  as provided by Lemma 4. We now show that we can obtain a natural rooted tree decomposition  $(\mathcal{T}, \beta_1)$  of  $G$  where the tree in the decomposition is  $\mathcal{T}$ . Here  $\beta_1 : V(\mathcal{T}) \rightarrow 2^{V(G)}$  and  $\beta_1(t) = \bigcup_{x \in \tau^{-1}(t)} \beta(x)$ . See Figure 1 for an example of  $(\mathcal{T}, \beta_1)$ .

From now on we fix  $G, (T, \beta), (\mathcal{T}, \tau)$ , and  $\beta_1$  for the rest of the section. We remark that to prove  $(\mathcal{T}, \beta_1)$  is a tree decomposition we will not need the *nice* properties of  $(\mathcal{T}, \tau)$  nor the properties of the bags and adhesions in  $(T, \beta)$ . We will later use them to deduce some helpful structural properties of  $(\mathcal{T}, \beta_1)$ . Figure 1 is the accompanying figure for the proof of the following lemma.

► **Lemma 5** (\*). *The pair  $(\mathcal{T}, \beta_1)$  is a tree decomposition of  $G$ .*

Observe that since  $(\mathcal{T}, \tau)$  is a nice tree partition of  $T$ , the tree decomposition  $(\mathcal{T}, \beta_1)$  has depth at most  $\lceil \log_2 |V(G)| \rceil$  and each of its bags is a union of at most four bags of  $(T, \beta)$ . We now prove a few other useful properties of  $(\mathcal{T}, \beta_1)$  that will help us design our desired tree decomposition.

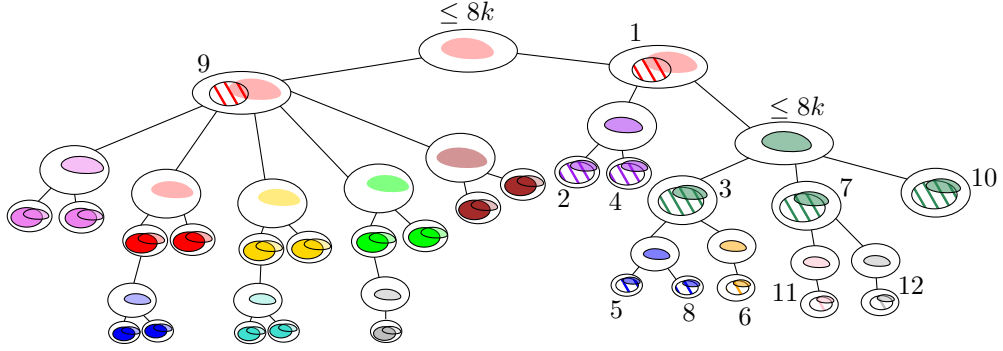
► **Lemma 6.** *There exists a function  $\gamma : V(\mathcal{T}) \rightarrow V(T)$ , and a set  $Y_t \subseteq \beta_1(t)$  for each node  $t \in V(\mathcal{T})$ , that satisfy, for each node  $t \in V(\mathcal{T})$ , the following properties:*

1.  $|Y_t| \leq 8k$
  2. If  $t$  is not the root  $r_{\mathcal{T}}$  then  $\beta_1(\mathcal{P}(t)) \cap \beta_1(t) \subseteq Y_t$
  3. for each distinct  $x, y \in \tau^{-1}(t)$ ,  $\beta(x) \cap \beta(y) \subseteq Y_t$
  4. for each child  $t_c$  of  $t$  in  $\mathcal{T}$ , it holds that  $t = \tau(\gamma(t_c))$  and  $\beta_1(t_c) \cap \beta_1(t) \subseteq Y_t \cup \beta(\gamma(t_c))$
- Furthermore,  $\gamma$  and the sets  $Y_t$  for each  $t \in V(\mathcal{T})$  can be computed in polynomial time.

**Proof.** For  $e = (x, x') \in E(T)$ , let  $\sigma(e) = \beta(x) \cap \beta(x')$  be the adhesion of edge  $e$  in  $(T, \beta)$ .

For  $r_{\mathcal{T}}$ , let  $T_{r_{\mathcal{T}}} = \bigcup_{x \in \tau^{-1}(r_{\mathcal{T}})} \sigma((x, \mathcal{P}(x)))$  and  $\gamma(r_{\mathcal{T}}) = r$ . For  $t \in V(\mathcal{T})$ ,  $t \neq r_{\mathcal{T}}$ , let  $E_t = \{e : e = (x, y) \in E(T), x \in \tau^{-1}(\mathcal{P}(t)), y \in \tau^{-1}(t)\}$ . Then let  $Y_t = \bigcup_{x \in \tau^{-1}(r_{\mathcal{T}})} \sigma((x, \mathcal{P}(x))) \cup \bigcup_{e \in E_t} \sigma(e)$ .

For each  $x$  in  $\tau^{-1}(\mathcal{P}(t))$ , there exists at most one  $y \in \tau^{-1}(t)$  such that  $(x, y) \in E(T)$ . This is because  $T[V_t]$  is connected, where  $t = \bigcup_{x \in V(\mathcal{T}_t)} \tau^{-1}(x)$ . If  $x$  has an edge to two vertices in  $\tau(t)$ , then there would be a cycle in  $T$ . Thus,  $|E_t| \leq 4$ . Further since  $\mathcal{T}$  is from a nice tree partition,  $|\tau^{-1}(t)| \leq 4$ . Therefore  $|Y_t| \leq 8k$  for each  $t \in V(\mathcal{T})$ .



■ **Figure 2** This shows the final tree decomposition  $(T^*, \beta^*)$  constructed using  $(T, \beta)$  and  $(\mathcal{T}, \beta_1)$  as shown in Fig 1.  $(T^*, \beta^*)$  is our desired tree decomposition with low depth, unbreakable bags and small adhesions. Recall that,  $V(T^*) = V(T) \cup V(\mathcal{T})$  and  $E(T^*) = \{(\tau(x), x) : x \in T\} \cup \{(\gamma(t), t) : t \in \mathcal{T} \setminus \{r_{\mathcal{T}}\}\}$ .

For  $t \neq r_{\mathcal{T}}, t \in V(\mathcal{T})$ . Since  $(T, \beta)$  is a tree decomposition and  $E_t$  are the only set of edges in  $\mathcal{T}$  between vertices in  $\tau^{-1}(t)$  and  $\tau^{-1}(\mathcal{P}(t))$ . Further since  $(\mathcal{T}, \beta_1)$  is a tree decomposition with  $\beta_1(t) = \bigcup_{x \in \tau^{-1}(t)} \beta(x)$ ,  $\beta_1(\mathcal{P}(t)) \cap \beta_1(t) \subseteq Y_t$ .

Recall that  $(T, \beta)$  is a tree decomposition. Therefore, for any two nodes  $x, y \in V(T)$ , consider a vertex  $v \in \beta(x) \cap \beta(y)$ . Note that  $v$  must belong to every bag of the node appearing on the unique  $x$  to  $y$  path in  $T$ . Let  $z \in V(T)$  be the least common ancestor of  $x$  and  $y$  – note that  $z$  may be equal to  $x$  or  $y$  or neither. Suppose  $z \notin \{x, y\}$ . Then,  $v$  must appear in  $\beta(x) \cap \beta(\mathcal{P}(x)) = \sigma(x, \mathcal{P}(x))$ , as well as in  $\sigma(y, \mathcal{P}(y))$ . Otherwise, if  $z = x$  (w.l.o.g.), then  $v \in \sigma(y, \mathcal{P}(y))$ . Since  $Y_t \supseteq \sigma(x, \mathcal{P}(x)) \cup \sigma(y, \mathcal{P}(y))$ , we get the third property.

Let  $t_c \in V(\mathcal{T})$ ,  $t_c \neq r_{\mathcal{T}}$ . Further let  $t = \mathcal{P}(t_c)$  in  $\mathcal{T}$ . We now show that for all but at most one vertex  $x \in \tau^{-1}(t)$ ,  $\beta_1(t_c) \cap \beta(x) \subseteq Y_t$ . If for all  $x \in \tau^{-1}(t)$ ,  $\beta_1(t_c) \cap \beta(x) \subseteq Y_t$  then we assign  $\gamma(t_c) = y$  for some  $y \in \tau^{-1}(t)$ . In this case, property 4 directly holds. Otherwise there is one vertex  $x \in \tau^{-1}(t)$  such that  $\beta_1(t_c) \cap \beta(x)$  is not a subset of  $Y_t$ , then we assign  $\gamma(t_c) = x$ . Here too, property 4 holds.

Now we show that for all but at most one vertex  $x \in \tau^{-1}(t)$ ,  $\beta_1(t_c) \cap \beta(x) \subseteq Y_t$ . Since  $\mathcal{T}$  is a nice tree partition,  $T[V_{t_c}]$  is a subtree of  $T$ , where  $V_{t_c} = \bigcup_{x \in V(\mathcal{T}_{t_c})} \tau^{-1}(x)$ . Next  $V_{t_c}$  does not contain  $r_T$  because  $t_c \neq r_{\mathcal{T}}$  and  $\tau(r_t) = r_{\mathcal{T}}$ . Further  $V_{t_c}$  is tree in the forest  $T \setminus \tau^{-1}(t)$ . Every vertex in  $\tau^{-1}(t)$  has at most one neighbor in  $V_{t_c}$  otherwise it will form a cycle. Since  $T$  is a rooted tree there is at most one node  $x \in \tau^{-1}(t)$  whose neighbor in  $V_{t_c}$  is not an ancestor (or a parent) of  $x$ . Thus for all others  $\beta(x) \cap \beta_1(t_c) \subseteq \sigma(x, \mathcal{P}(x)) \subseteq Y_T$ . ◀

Let  $\gamma : V(\mathcal{T}) \rightarrow V(T)$  and  $Y_t \subseteq \beta_1(t)$  for each node  $t \in \mathcal{T}$  be function and sets given by Lemma 6. We now define a pair  $(T^*, \beta^*)$  based on  $T, \mathcal{T}, \gamma$  and  $Y_t$  that we will prove to be a tree decomposition of  $G$  having all our desired properties including unbreakable bags.

▶ **Definition 7** ( $(T^*, \beta^*)$ ). Let  $T^*$  be a graph with  $V(T^*) = V(T) \cup V(\mathcal{T})$  and  $E(T^*) = \{(\tau(x), x) : x \in T\} \cup \{(\gamma(t), t) : t \in \mathcal{T} \setminus \{r_{\mathcal{T}}\}\}$ . Also let  $\beta^* : V(T^*) \rightarrow 2^{V(G)}$  be a function with  $\beta^*(t) = Y_t$ , for  $t \in \mathcal{T}$  and  $\beta^*(x) = Y_{\tau(x)} \cup \beta(x)$  for  $x \in T$ .

In the following, we show that  $(T^*, \beta^*)$  is a tree decomposition of  $G$  (see Figure 2).

▶ **Lemma 8** (\*).  $(T^*, \beta^*)$  is a rooted tree decomposition of  $G$ . Further  $(T^*, \beta^*)$  satisfies the following properties:

1. every adhesion of  $(T^*, \beta^*)$  is of size at most  $8k$
2. every bag of  $(T^*, \beta^*)$  is  $(q + 8k, k)$ -unbreakable in  $G$ .
3.  $T^*$  has depth at most  $2\lceil \log_2 |V(G)| \rceil$

We are now ready to prove the main theorem of this section.

► **Theorem 9.** *There exists a polynomial-time algorithm that takes input an  $n$ -vertex graph  $G$  and positive integers  $k$  and  $q$ , and a rooted tree decomposition  $(T, \beta)$  of  $G$  satisfying the following properties:*

1. every adhesion of  $(T, \beta)$  is of size at most  $k$
  2. every bag of  $(T, \beta)$  is  $(q, k)$ -unbreakable in  $G$
- and finds a compact tree decomposition  $(T', \beta')$  of  $G$  satisfying the following properties:
1. every adhesion of  $(T', \beta')$  is of size at most  $8k$
  2. every bag of  $(T', \beta')$  is  $(q + 8k, k)$ -unbreakable in  $G$ .
  3.  $T'$  has depth at most  $2\lceil \log_2 n \rceil$ .

**Proof.** Let  $(T, \beta)$  be the input tree decomposition of  $G$ . We first compute a nice tree partition  $(\mathcal{T}, \tau)$  of  $T$  using Lemma 4. Then we obtain the tree decomposition  $(\mathcal{T}, \beta_1)$  of  $G$  where  $\beta_1 : V(G) \rightarrow V(\mathcal{T})$  and  $\beta_1(t) = \bigcup_{x \in \tau^{-1}(t)} \beta(x)$  – it is a tree decomposition by Lemma 5.

Let  $\beta^* : V(T^*) \rightarrow 2^{V(G)}$  be a function with  $\beta^*(t) = Y_t$ , for  $t \in \mathcal{T}$  and  $\beta^*(x) = Y_{\tau(x)} \cup \beta(x)$  for  $x \in T$ . We compute the tree decomposition  $(T^*, \beta^*)$  with  $V(T^*) = V(T) \cup V(\mathcal{T})$  and  $E(T^*) = \{(\tau(x), x) : x \in T\} \cup \{(\gamma(t), t) : t \in \mathcal{T} \setminus \{r_{\mathcal{T}}\}\}$ . By Lemma 8 it satisfies all our required properties except compactness. We can in polynomial time obtain a compact tree decomposition  $(T', \beta')$  whose each bag is a subset of some bag of  $(T^*, \beta^*)$  and whose height is the same as  $\mathcal{T}^*$  [5]. Thus the tree decomposition  $(T', \beta')$  will satisfy all our required properties. ◀

The following corollary directly follows from Theorem 9, and a known result ([9]) that outputs a tree decomposition of a graph satisfying the premise of Corollary 10 in time  $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ .

► **Corollary 10.** *Given an  $n$ -vertex graph  $G$  and an integer  $k$ , one can in time  $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$  compute a rooted compact tree decomposition  $(T, \beta)$  of  $G$  such that:*

1. Every adhesion of  $(T, \beta)$  is of size at most  $8k$
2. Every bag of  $(T, \beta)$  is  $(9k, k)$ -unbreakable in  $G$
3.  $T$  has depth at most  $2\lceil \log_2 n \rceil$

## 4 Exact and Approximation algorithms

Let  $(G, c, k, r^\circ, \chi)$  be an instance of FAIR BISECTION and let  $n = |V(G)|$ . We start by invoking the algorithm of Theorem 10 with  $G$  and  $k$  to obtain a rooted compact tree decomposition  $(T, \beta)$  of  $G$  with root  $r$ , having  $(9k, k)$ -edge-unbreakable bags and adhesions of size at most  $8k$ . This takes time  $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ . Recall that an edge cut  $(A, B)$  is  $(\epsilon, r^\circ)$ -fair if  $A$  and  $B$  contain no more than  $r_i(1 + \epsilon)$  and  $(c_i - r_i)(1 + \epsilon)$  vertices respectively.

► **Theorem 11.** *Given an instance  $(G, c, k, r^\circ, \chi)$  of FAIR BISECTION and  $\epsilon > 0$  there exists an algorithm that in time  $2^{\mathcal{O}(k \log k)} \cdot \left(\frac{c}{\epsilon}\right)^{\mathcal{O}(c)} \cdot n^{\mathcal{O}(1)}$  finds an  $(\epsilon, r^\circ)$ -fair edge cut of  $G$  if one exists, else returns no.*

Given a subset  $S \subseteq V(G)$ , we use  $\chi^\circ(S)$  to denote the  $c$  length tuple where the  $i^{\text{th}}$  entry is the number of vertices  $v$  in  $S$  having color  $i$ , i.e.  $\chi(v) = i$ . We remark that we use  $^\circ$  to denote tuples of integers of length  $c$ . Further we use operators such as  $+$ ,  $-$ , scalar multiplication, and  $\lceil \cdot \rceil$  on tuples, which perform the respective operations on each entry in the tuple(s).

## 63:12 Parameterized Complexity of Fair Bisection

For a node  $t \in V(T)$  recall that  $\gamma(t) = \bigcup_{s: \text{descendant of } t} \beta(s)$ ,  $\alpha(t) = \gamma(t) \setminus \sigma(t)$ ,  $G_t = G[\gamma(t)] - E[G[\sigma(t)]]$ . We perform bottom-up dynamic programming on  $(T, \beta)$ . For each node  $t \in V(T)$ , we first define a Boolean function  $f_t : \{0, \dots, k\} \times 2^{\sigma(t)} \times \{0, \dots, n\}^c \times \{0, \dots, n\}^c \rightarrow \{\text{True}, \text{False}\}$ . For each integer  $w \in \{0, \dots, k\}$ , subset  $A_t \subseteq \sigma(t)$ , and  $c$  length tuples  $a^\circ$  and  $b^\circ$  with  $a^\circ, b^\circ \in \{0, \dots, n\}^c$ , we define the following.

► **Definition 12.**  $f_t(w, A_t, a^\circ, b^\circ) = \text{True}$  if there exists an edge cut  $(A, B)$  of  $G_t$  that satisfies the following properties: (1)  $(A, B)$  has order at most  $w$ , (2)  $A \cap \sigma(t) = A_t$ , (3)  $\chi^\circ(A \cap \alpha(t)) = a^\circ$ , and (4)  $\chi^\circ(B \cap \alpha(t)) = b^\circ$ . If such a cut does not exist,  $f_t(w, A_t, a^\circ, b^\circ) = \text{False}$ . Further if  $f_t(w, A_t, a^\circ, b^\circ) = \text{True}$ , we say an edge cut  $(A, B)$  of  $G_t$  that satisfies properties 1 – 4 **realizes**  $f_t(w, A_t, a^\circ, b^\circ)$ .

From the definition of  $f_t$  one can make the following observation.

► **Observation 13.**  $(G, c, k, r^\circ, \chi)$  is a yes-instance to FAIR BISECTION if and only if for  $f_r(k, \emptyset, r^\circ, c - r^\circ) = \text{True}$ , where  $r$  is the root of  $T$ .

In order to reduce the size of the domain of  $f$  (and hence the running time), we work with the *reduced* domain  $D = \{(1 + \delta)^i : i \geq 0\}$ . This will approximate the number of vertices of each color at either side of the cut to the nearest power of  $1 + \delta$ , where  $\delta > 0$  is a parameter whose value will be fixed later.

Let  $\mathcal{C}_t$  be the set of all possible edge-cuts  $(A, B)$  of  $G_t$ . To compute  $f_t$  we have a *table*  $M_t : \{0, \dots, k\} \times 2^{\sigma(t)} \times D^c \times D^c \rightarrow \{\mathcal{C}_t \cup \perp\}$  that satisfies properties  $M_t \rightarrow f_t$  and  $f_t \rightarrow M_t$  (defined below in Definition 14 and 16).  $M_t$  will help us to approximately obtain  $f_t$ . Let  $z \geq 0$  be a sufficiently large constant; for example,  $z = 10$  suffices. We have Definitions 14 and 16 that will be crucial towards proving the correctness of the approximation algorithm.

► **Definition 14** (Property  $M_t \rightarrow f_t$ ). If  $M_t(w, A_t, a^\circ, b^\circ) \neq \perp$  then  $\exists x^\circ \in \{0, \dots, n\}$  and  $y^\circ \in \{0, \dots, n\}$  such that:

- $f_t(w, A_t, x^\circ, y^\circ) = \text{True}$  and  $M_t(w, A_t, a^\circ, b^\circ)$  is an edge-cut that realizes  $f_t(w, A_t, x^\circ, y^\circ)$
- $a^\circ \leq x^\circ \leq (1 + \delta)^{z \cdot \text{ht}(t) \log^2 n} \cdot a^\circ$
- $b^\circ \leq y^\circ \leq (1 + \delta)^{z \cdot \text{ht}(t) \log^2 n} \cdot b^\circ$

► **Definition 15** (Global-feasible edge cut). An edge cut  $(A, B)$  is *global-feasible* if there exists an edge cut  $(A', B')$  of  $G$  having order at most  $k$  which induces the cut  $(A, B)$  on  $A \cup B$ .

► **Definition 16** (Property  $f_t \rightarrow M_t$ ). If  $f_t(w, A_t, x^\circ, y^\circ) = \text{True}$  and there is a *global-feasible* edge cut  $(A, B)$  of  $G_t$  that realizes it then  $\exists a^\circ \in D^c$  and  $b^\circ \in D^c$  such that:

- $M_t(w, A_t, a^\circ, b^\circ) \neq \perp$
- $a^\circ \leq x^\circ \leq (1 + \delta)^{z \cdot \text{ht}(t) \log^2 n} \cdot a^\circ$
- $b^\circ \leq y^\circ \leq (1 + \delta)^{z \cdot \text{ht}(t) \log^2 n} \cdot b^\circ$

► **Definition 17** (Good  $M_t$ ).  $M_t$  is *good* if it satisfies properties  $M_t \rightarrow f_t$  and  $f_t \rightarrow M_t$ .

► **Lemma 18.** For each  $\epsilon > 0$  and  $\delta = \frac{\epsilon}{2z \log^3 n}$ , if  $f_r(k, \phi, r^\circ, c^\circ - r^\circ) = \text{True}$  and  $M_r$  is good, then  $\exists a^\circ, b^\circ \in D^c$  such that  $M_r(k, \phi, a^\circ, b^\circ)$  is a  $(\epsilon, r^\circ)$ -fair edge cut of  $G$ .

**Proof.** We first note that, if  $\delta := \frac{\epsilon}{2z \log^3 n}$ , then  $(1 + \delta)^{z \cdot \log^3 n} \leq 1 + \epsilon$ . This is because  $\ln(1 + \epsilon) \geq \frac{\epsilon}{1 + \epsilon} \geq \frac{\epsilon}{2}$ , since  $\epsilon \in (0, 1)$ , which implies that  $(1 + \delta)^{z \cdot \log^3 n} \leq \exp\left(\frac{\epsilon}{2z \log^3 n} \cdot z \log^3 n\right) \leq \exp\left(\frac{\ln(1 + \epsilon)}{z \log^3 n} \cdot z \log^3 n\right) = 1 + \epsilon$ . Furthermore,  $\log_{1 + \delta} n = (\log n / \epsilon)^{O(1)}$ .

Let  $f_r(k, \phi, r^\circ, c^\circ - r^\circ) = \text{True}$  and  $M_r$  satisfy properties  $M_r \rightarrow f_r$  and  $f_r \rightarrow M_r$ . Since  $ht(T) = \log n$ , by the previous claim  $(1 + \delta)^{z \cdot ht(t) \log^2 n} \leq 1 + \epsilon$ . So by property  $f_r \rightarrow M_r$ ,  $\exists a^\circ, b^\circ \in D^c$  such that: (1)  $M_r(k, \phi, a^\circ, b^\circ) \neq \perp$ , (2)  $a^\circ \leq r^\circ \leq (1 + \epsilon) \cdot a^\circ$ , and  $b^\circ \leq c^\circ - r^\circ \leq (1 + \epsilon) \cdot b^\circ$ .

Further by property  $M_r \rightarrow f_r$ , since  $M_r(k, \phi, a^\circ, b^\circ) \neq \perp$ ,  $\exists x^\circ, y^\circ \in \{0, \dots, n\}$  such that, (1)  $f_r(k, \phi, x^\circ, y^\circ) = \text{True}$  and  $M_r(k, \phi, a^\circ, b^\circ)$  is an edge-cut that realizes  $f_r(k, \phi, x^\circ, y^\circ)$ , (2)  $a^\circ \leq x^\circ \leq (1 + \epsilon) \cdot a^\circ \leq (1 + \epsilon) \cdot r^\circ$ , and (3)  $b^\circ \leq y^\circ \leq (1 + \epsilon) \cdot b^\circ \leq (1 + \epsilon) \cdot (r^\circ - c^\circ)$ . Thus,  $M_r(k, \phi, a^\circ, b^\circ)$  is a  $(\epsilon, r^\circ)$ -fair edge-cut of  $G$ . This completes our proof.  $\blacktriangleleft$

Lemma 18 shows us that computing a good table  $M$  efficiently is sufficient for obtaining our final approximation. We now state as a theorem that we can compute a good  $M_t$  assuming a good  $M_{t'}$  has been computed for each  $t' \in \text{child}(t)$ .

► **Lemma 19** (\*). *There exists an algorithm that takes as input  $t \in V(T)$ ,  $\delta > 0$ ,  $(T, \beta)$ , and a good  $M_{t'}$  for each  $t' \in \text{child}(t)$  and computes a good  $M_t$  in time  $2^{\mathcal{O}(k \log k)} (\log_{1+\delta} n)^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$ .*

Assuming Lemma 19 and Lemma 18, the correctness of our algorithm follows. Setting  $\delta$  as in Lemma 18, we obtain our desired runtime thus proving Theorem 11.

**Proof of Theorem 11.** Let  $\delta := \frac{\epsilon}{2z \log^3 n}$ . In our algorithm we compute  $M$  by computing good  $M_t$  using Lemma 19 for each  $t \in V(T)$ , bottom up, starting from leaves of  $T$  to root of  $T$ . We finally go over each  $a^\circ, b^\circ \in D^c$  and output a cut  $M_r(k, \phi, a^\circ, b^\circ)$  that is a  $(\epsilon, r^\circ)$ -fair edge cut of  $G$  if one exists.

The correctness follows directly from the definition of  $f$  and Lemma 18. The time taken by our algorithm is equal the size of domain of  $M$  times the time taken to compute each entry in  $M$ . The size of the domain of  $M$  is at most  $2^{\mathcal{O}(k)} (\log_{1+\delta} n)^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$  because  $|D| \leq \log_{1+\delta} n$  and all adhesions in  $(T, \beta)$  have size at most  $8k$ . The time taken to compute each entry in  $M$  is  $2^{\mathcal{O}(k \log k)} (\log_{1+\delta} n)^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$  by Lemma 19.

Using  $\log_{1+\delta} n = (\log n / \epsilon)^{\mathcal{O}(1)}$  and a standard case analysis on whether  $c \leq \frac{\log n}{\log \log n}$ , it follows that the total time taken is  $2^{\mathcal{O}(k \log k)} (\log_{1+\delta} n)^{\mathcal{O}(c)} n^{\mathcal{O}(1)} \leq 2^{\mathcal{O}(k \log k)} \left(\frac{\epsilon}{\epsilon}\right)^{\mathcal{O}(c)} n^{\mathcal{O}(1)}$ .  $\blacktriangleleft$

### Computing $M_t$ : A sketch of proof of Lemma 19

In particular, we design an algorithm that takes as input a graph  $G$ , the tree decomposition  $(T, \beta)$  and a node  $t$  of  $T$ , together with dynamic programming tables  $M_{t'}$  for every child  $t'$  of  $t$ , and outputs the appropriate dynamic programming table  $M_t$  (which is good) for  $t$ . This algorithm is an adaptation of a similar step performed by Cygan et al. [10] in their algorithm for the MINIMUM BISECTION problem. The algorithm of Cygan et al. [10] proceeds by a random coloring step, followed by a “knapsack”-like dynamic programming algorithm. Our algorithm proceeds in a similar manner, but faces the following key difficulty: in order to keep time and space bounded by  $f(k, c, \epsilon) n^{\mathcal{O}(1)}$  we can only store approximate values in the knapsack dynamic programming table (the table satisfies soundness and completeness properties similar to Definition 17). Therefore, after computing each entry of the table (from previous entries) we need to perform a rounding step that introduces a  $(1 + (\frac{\epsilon}{\log n})^{\mathcal{O}(1)})$  multiplicative factor in the error bound. The standard way of solving KNAPSACK involves considering each item in the input one by one, however this would lead to the rounding error possibly accumulating and getting out of hand. We overcome this by organizing the dynamic program in a complete binary tree. That is, split the items in two equal sized groups, compute dynamic programming tables for the two groups recursively, and combine

the dynamic programming tables to the two halves to a dynamic programming for all the items. This ensures that the total error is upper bounded by a multiplicative factor of  $(1 + (\frac{\epsilon}{\log n})^{\mathcal{O}(1)})^{\mathcal{O}(\log^3 n)} = 1 + \mathcal{O}(\epsilon)$ .

## 5 Hardness

Here, we sketch the proof of our result that establishes  $W[1]$ -hardness of FAIR BISECTION. To this end, we first consider the following problem, called MULTI-DIMENSIONAL SUBSET SUM. In this problem, we are given an instance  $(\mathcal{V}, T)$ , where  $\mathcal{V} = \{V_1, \dots, V_n\}$ , such that each  $V_i \in \mathcal{V}$  is a  $d$ -dimensional vector, i.e.,  $V_i \in \mathbb{Z}_{\geq 0}^d$ ; and  $T \in \mathbb{Z}_{\geq 0}^d$  is the  $d$ -dimensional target vector. The task is to determine whether there exists a subset  $U \subseteq \mathcal{V}$  such that  $\sum_{V_i \in U} V_i = T$ ?

Although it is folklore that MULTI-DIMENSIONAL SUBSET SUM is  $W[1]$ -hard parameterized by the dimension  $d$ , we are unable to find a reference for this result. Thus, we give a reduction from BINARY CONSTRAINED SATISFACTION PROBLEM to MDSS; the  $W[1]$ -hardness of the former problem was established in [23, 21]. In fact, our reduction shows that MDSS is  $W[1]$ -hard even when the integer entries in each vector are bounded by a polynomial in  $n$ . As the first step of our reduction, given an instance  $(\mathcal{V}, T)$  of MDSS, we reduce it to MULTI-DIMENSIONAL PARTITION (MDP), where the target vector  $T$  is exactly half of the sum of entries along each dimension. Now, for each vector  $V_i \in \mathcal{V}$ , we create a subset  $U_i$  of vertices, which contains exactly  $V_i(j)$  many vertices of color  $1 \leq j \leq d$ . Then, we arbitrarily choose a vertex in  $U_i$  and connect the rest of the vertices in  $U_i$  to it, making a connected component a star. Proceeding this way for each vector  $V_i$ , we obtain a graph  $G$  that is a disjoint union of  $n$  stars on  $U_i$ 's, with each  $U_i$  containing at most polynomially many vertices of each color. It is straightforward to see the equivalence between the instance  $(\mathcal{V}, T)$  of MDP, and the resulting instance of FAIR BISECTION, with the cut-size  $k$  being zero. Thus, we conclude with the following theorem, whose formal proof can be found in the full version.

► **Theorem 20.** FAIR BISECTION is  $W[1]$ -hard parameterized by the number of colors  $c$ , even when  $k$ , the cut-size is zero.

## 6 Conclusion

In this paper, we initiated the study of FAIR BISECTION from the perspective of parameterized algorithms. We showed that the problem is  $W[1]$ -hard parameterized by the number of colors  $c$ , even when  $k = 0$ ; thus, we cannot hope to generalize the FPT algorithm to FAIR BISECTION with a running time of the form  $f(k, c) \cdot n^{\mathcal{O}(1)}$ . On the other hand, the known  $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$  algorithm for MINIMUM BISECTION ([9, 10]) extends to FAIR BISECTION in a straightforward manner with running time  $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(c)}$ . Our main result is that FAIR BISECTION admits an FPT-approximation algorithm that finds an  $(\epsilon, r)$ -fair bisection in time  $2^{\mathcal{O}(k \log k)} \cdot (\frac{c}{\epsilon})^{\mathcal{O}(c)} \cdot n^{\mathcal{O}(1)}$ . In fact, by setting  $\epsilon = 1/(2n)$ , we can obtain the previously mentioned exact algorithm as a corollary.

We note that our approximation algorithm also works in the setting where a vertex can belong to multiple color classes. Also, our technique can be extended to FAIR  $q$ -SECTION problem, where we want to partition the vertex set into  $q$  parts such that (i) at most  $k$  edges with endpoints in different parts, and (ii) each part has proportional representation from each color – here, the algorithm will have an XP dependence on  $q$ .

Our main conceptual contribution is the observation that it is possible to design parameterized approximation algorithms by applying the technique of Lampis [20] to design DP over tree decompositions with unbreakable bags. Towards this goal we designed an algorithm that

given a graph  $G$  and integer  $k$  computes a  $(9k, k)$ -unbreakable tree decomposition of  $G$  with logarithmic depth and adhesions of size at most  $8k$  in time  $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ . We expect that this will be a useful tool for obtaining parameterized approximation algorithms for other problems by using Lampis [20]-style dynamic programming over tree decompositions with unbreakable bags.

---

## References

- 1 Sayan Bandyapadhyay, Fedor V. Fomin, and Kirill Simonov. On coresets for fair clustering in metric and euclidean spaces and their applications. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 23:1–23:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.23.
- 2 Sayan Bandyapadhyay, Tanmay Inamdar, Shreyas Pai, and Kasturi R. Varadarajan. A constant approximation for colorful k-center. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 12:1–12:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.12.
- 3 Johannes Blömer, Christiane Lammersen, Melanie Schmidt, and Christian Sohler. Theoretical analysis of the k-means algorithm—a survey. In *Algorithm Engineering*, pages 81–116. Springer, 2016.
- 4 Hans L. Bodlaender and Torben Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. *SIAM J. Comput.*, 27(6):1725–1746, 1998. doi:10.1137/S0097539795289859.
- 5 Mikolaj Bojanczyk and Michal Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 407–416. ACM, 2016. doi:10.1145/2933575.2934508.
- 6 Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. Proportionally fair clustering. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1032–1041. PMLR, 2019. URL: <http://proceedings.mlr.press/v97/chen19d.html>.
- 7 Yixin Chen, Ya Zhang, and Xiang Ji. Size regularized cut for data clustering. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 211–218, 2005. URL: <https://proceedings.neurips.cc/paper/2005/hash/379a7ba015d8bf1c70b8add2c287c6fa-Abstract.html>.
- 8 Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016.
- 9 Marek Cygan, Pawel Komosa, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, Saket Saurabh, and Magnus Wahlström. Randomized contractions meet lean decompositions. *ACM Trans. Algorithms*, 17(1):6:1–6:30, 2021. doi:10.1145/3426738.
- 10 Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Minimum bisection is fixed-parameter tractable. *SIAM J. Comput.*, 48(2):417–450, 2019. doi:10.1137/140988553.
- 11 Jefferey Dastin. Amazon scraps secret ai recruiting tool that showed bias against women. *Reuters*, 2018. <https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G>.

- 12 Sorelle A. Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. The (im)possibility of fairness: different value systems require different mechanisms for fair decision making. *Commun. ACM*, 64(4):136–143, 2021. doi:10.1145/3433949.
- 13 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 14 Mehrdad Ghadiri, Samira Samadi, and Santosh S. Vempala. Socially fair k-means clustering. In Madeleine Clare Elish, William Isaac, and Richard S. Zemel, editors, *FACCT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pages 438–448. ACM, 2021. doi:10.1145/3442188.3445906.
- 15 Patrick J Grother, Patrick J Grother, Mei Ngan, and K Hanaoka. *Face recognition vendor test (FRVT)*. US Department of Commerce, National Institute of Standards and Technology, 2014.
- 16 Rudolf Halin. Tree-partitions of infinite graphs. *Discret. Math.*, 97(1-3):203–217, 1991. doi:10.1016/0012-365X(91)90436-6.
- 17 Lingxiao Huang, Shaofeng H.-C. Jiang, and Nisheeth K. Vishnoi. Coresets for clustering with fairness constraints. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7587–7598, 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/810dfbbebb17302018ae903e9cb7a483-Abstract.html>.
- 18 Xinrui Jia, Kshiteej Sheth, and Ola Svensson. Fair colorful k-center clustering. In Daniel Bienstock and Giacomo Zambelli, editors, *Integer Programming and Combinatorial Optimization – 21st International Conference, IPCO 2020, London, UK, June 8-10, 2020, Proceedings*, volume 12125 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 2020. doi:10.1007/978-3-030-45771-6\_17.
- 19 Subhash Khot and Nisheeth K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into  $\ell_1$ . *J. ACM*, 62(1):8:1–8:39, 2015. doi:10.1145/2629614.
- 20 Michael Lampis. Parameterized approximation schemes using graph widths. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 775–786. Springer, 2014. doi:10.1007/978-3-662-43948-7\_64.
- 21 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2181–2200. SIAM, 2020. doi:10.1137/1.9781611975994.134.
- 22 Yury Makarychev and Ali Vakilian. Approximation algorithms for socially fair clustering. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 3246–3264. PMLR, 2021. URL: <http://proceedings.mlr.press/v134/makarychev21a.html>.
- 23 Dániel Marx. Can you beat treewidth? *Theory Comput.*, 6(1):85–112, 2010. doi:10.4086/toc.2010.v006a005.
- 24 Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019.
- 25 Evelien Otte and Ronald Rousseau. Social network analysis: a powerful strategy, also for the information sciences. *Journal of information Science*, 28(6):441–453, 2002.
- 26 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 255–264. ACM, 2008. doi:10.1145/1374376.1374415.



- 27 Lior Rokach. A survey of clustering algorithms. In *Data mining and knowledge discovery handbook*, pages 269–298. Springer, 2009.
- 28 Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means. In Evripidis Bampis and Nicole Megow, editors, *Approximation and Online Algorithms – 17th International Workshop, WAOA 2019, Munich, Germany, September 12-13, 2019, Revised Selected Papers*, volume 11926 of *Lecture Notes in Computer Science*, pages 232–251. Springer, 2019. doi:10.1007/978-3-030-39479-0\_16.
- 29 Detlef Seese. Tree-partite graphs and the complexity of algorithms. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985*, volume 199 of *Lecture Notes in Computer Science*, pages 412–421. Springer, 1985. doi:10.1007/BFb0028825.
- 30 Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.
- 31 Jin-Tai Yan and Pei-Yung Hsiao. A fuzzy clustering algorithm for graph bisection. *Inf. Process. Lett.*, 52(5):259–263, 1994. doi:10.1016/0020-0190(94)00148-0.