Polylogarithmic Approximation Algorithms for Weighted-$\mathcal{F}$-Deletion Problems*

AKANKSHA AGRAWAL†, Ben-Gurion University of the Negev, Israel
DANIEL LOKSHTANOV, University of California Santa Barbara, United States
PRANABENDU MISRA, Max Planck Institute for Informatics, Germany
SAKET SAURABH, University of Bergen, Norway and The Institute of Mathematical Sciences, India
MEIRAV ZEHAVI, Ben-Gurion University of the Negev, Israel

For a family of graphs $\mathcal{F}$, the Weighted $\mathcal{F}$ Vertex Deletion problem, is defined as follows: given an n-vertex undirected graph G and a weight function $w : V(G) \to \mathbb{R}$, find a minimum weight subset $S \subseteq V(G)$ such that $G - S$ belongs to $\mathcal{F}$. We devise a recursive scheme to obtain $O(\log^{O(1)} n)$-approximation algorithms for such problems, building upon the classical technique of finding *balanced separators*. We obtain the first $O(\log^{O(1)} n)$-approximation algorithms for the following problems.

- Let $\mathscr{F}$ be a finite set of graphs containing a planar graph, and $\mathcal{F} = \mathscr{G}(\mathscr{F})$ be the maximal family of graphs such that every graph $H \in \mathscr{G}(\mathscr{F})$ excludes all graphs in $\mathscr{F}$ as minors. The vertex deletion problem corresponding to $\mathcal{F} = \mathscr{G}(\mathscr{F})$ is the Weighted Planar $\mathscr{F}$-Minor-Free Deletion (WP$\mathscr{F}$-MFD) problem. We give a randomized and a deterministic approximation algorithms for WP$\mathscr{F}$-MFD with ratios $O(\log^{1.5} n)$ and $O(\log^2 n)$, respectively. Prior to our work, a randomized constant factor approximation algorithm for the *unweighted* version was known [FOCS 2012]. After our work, a deterministic constant factor approximation algorithm for the *unweighted* version was also obtained [SODA 2019].
- We give an $O(\log^2 n)$-factor approximation algorithm for Weighted Chordal Vertex Deletion, the vertex deletion problem to the family of chordal graphs. On the way to this algorithm, we also obtain a constant factor approximation algorithm for Multicut on chordal graphs.
- We give an $O(\log^3 n)$-factor approximation algorithm for Weighted Distance Hereditary Vertex Deletion.

We believe that our recursive scheme can be applied to obtain $O(\log^{O(1)} n)$-approximation algorithms for many other problems as well.

*A preliminary version of this work has been published at the 21st International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2018).
†The corresponding author

Author's addresses: A. Agrawal, Department of Computer Science, Ben-Gurion University of the Negev, Beersheba, Israel; email: agrawal@post.bgu.ac.il; D. Lokshtanov, Department of Computer Science, University of California Santa Barbara, Santa Barbara, United States; email: daniello@ucsb.edu; P. Misra, Department of Algorithms and Complexity, Max Planck Institute for Informatics, Saarbrucken, Germany; email: pmisra@mpi-inf.mpg.de; S. Saurabh, Department of Informatics, University of Bergen, Bergen, Norway, and Institute of Mathematical Sciences, HBNI, Chennai, India; email: saket@imsc.res.in; M. Zehavi, Department of Computer Science, Ben-Gurion University of the Negev, Beersheba, Israel; email: meiravze@bgu.ac.il.
Authors' addresses: Akanksha Agrawal, agrawal@post.bgu.ac.il, Ben-Gurion University of the Negev, Beersheba, Israel; Daniel Lokshtanov, daniello@ucsb.edu, University of California Santa Barbara, Santa Barbara, United States; Pranabendu Misra, pmisra@mpi-inf.mpg.de, Max Planck Institute for Informatics, Saarbrucken, Germany; Saket Saurabh, saket@imsc.res.in, University of Bergen, Bergen, Norway, The Institute of Mathematical Sciences, HBNI, Chennai, India; Meirav Zehavi, meiravze@bgu.ac.il, Ben-Gurion University of the Negev, Beersheba, Israel.

1942-3454/2017/1-ART39 $15.00
https://doi.org/0000001.0000001

CCS Concepts: • **Theory of computation → Graph algorithms analysis**; **Approximation algorithms analysis**; **Algorithm design techniques**.

Additional Key Words and Phrases: approximation algorithm, balanced separators, chordal graphs, Planar $\mathcal{F}$ minor free graphs, distance hereditary graphs, $\mathcal{F}$-Vertex Deletion

ACM Reference Format:
Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. 2017. Polylogarithmic Approximation Algorithms for Weighted-$\mathcal{F}$-Deletion Problems. *ACM Trans. Comput. Theory* 9, 4, Article 39 (January 2017), 39 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Let $\mathcal{F}$ be a family of undirected graphs. Then a natural optimization problem is as follows.

Weighted $\mathcal{F}$ Vertex Deletion
Input: An undirected graph G and a weight function $w : V(G) \to \mathbb{R}$.
Question: Find a minimum weight subset $S \subseteq V(G)$ such that $G - S$ belongs to $\mathcal{F}$.

The Weighted $\mathcal{F}$ Vertex Deletion problem captures a wide class of node (or vertex) deletion problems that have been studied from the 1970s. For example, when $\mathcal{F}$ is the family of independent sets, forests, bipartite graphs, planar graphs, and chordal graphs, then the corresponding vertex deletion problem corresponds to Weighted Vertex Cover, Weighted Feedback Vertex Set, Weighted Vertex Bipartization (also called Weighted Odd Cycle Transversal), Weighted Planar Vertex Deletion and Weighted Chordal Vertex Deletion, respectively. By a classic theorem of Lewis and Yannakakis [35], the decision version of the Weighted $\mathcal{F}$ Vertex Deletion problem—deciding whether there exists a set S of weight at most k, such that removing S from G results in a graph with property Π—is NP-complete for every non-trivial hereditary property[1] Π.

Characterizing the graph properties, for which the corresponding vertex deletion problems can be approximated within a bounded factor in polynomial time, is a long standing open problem in approximation algorithms [50]. In spite of a long history of research, we are still far from a complete characterization. Constant factor approximation algorithms for Weighted Vertex Cover are known since 1970s [4, 38]. Lund and Yannakakis observed that the vertex deletion problem for any hereditary property with a "finite number of minimal forbidden induced subgraphs" can be approximated within a constant ratio [36]. They conjectured that for every nontrivial, hereditary property Π with an infinite forbidden set, the corresponding vertex deletion problem cannot be approximated within a constant ratio. However, it was later shown that Weighted Feedback Vertex Set, which does not have a finite forbidden set, admits a constant factor approximation [2, 5], thus disproving their conjecture. On the other hand a result by Yannakakis [49] shows that, for a wide range of graph properties Π, approximating the minimum number of vertices to delete in order to obtain a *connected* graph with the property Π within a factor $n^{1-\varepsilon}$ is NP-hard. We refer to [49] for the precise list of graph properties to which this result applies to, but it is worth mentioning the list includes the class of acyclic graphs and the class of outerplanar graphs.

In this paper, we explore the approximability of Weighted $\mathcal{F}$ Vertex Deletion for several different families $\mathcal{F}$ and design $O(\log^{O(1)} n)$-factor approximation algorithms for these problems. More precisely, our results are as follows.

[1]A graph property Π is simply a family of graphs closed under isomorphism, and it is called *non-trivial* if there exists an infinite number of graphs that are in Π, as well as an infinite number of graphs that are not in Π. A non-trivial graph property Π is called *hereditary* if $G \in \Pi$ implies that every induced subgraph of G is also in Π.

(1) Let $\mathscr{F}$ be a finite set of graphs that includes a planar graph. Let $\mathcal{F} = \mathscr{G}(\mathscr{F})$ be the family of graphs such that every graph $H \in \mathscr{G}(\mathscr{F})$ does not contain a graph from $\mathscr{F}$ as a minor. The vertex deletion problem corresponding to $\mathcal{F} = \mathscr{G}(\mathscr{F})$ is known as the Weighted Planar $\mathscr{F}$-Minor-Free Deletion (WP$\mathscr{F}$-MFD). The WP$\mathscr{F}$-MFD problem is a very generic problem and by selecting different sets of forbidden minors $\mathscr{F}$, one can obtain various fundamental problems such as Weighted Vertex Cover, Weighted Feedback Vertex Set or Weighted Treewidth η-Deletion. Our first result is a randomized $O(\log^{1.5} n)$-factor (deterministic $O(\log^2 n)$-factor) approximation algorithm for WP$\mathscr{F}$-MFD, for any finite $\mathscr{F}$ that contains a planar graph.
(2) We give an $O(\log^2 n)$-factor approximation algorithm for Weighted Chordal Vertex Deletion (WCVD), the vertex deletion problem corresponding to the family of chordal graphs. On the way to this algorithm, we also obtain a constant factor approximation algorithm for Weighted Multicut in chordal graphs.
(3) We give an $O(\log^3 n)$-factor approximation algorithm for Weighted Distance Hereditary Vertex Deletion (WDHVD). This is also known as the Weighted Rankwidth-1 Vertex Deletion (WR-1VD) problem. This is the vertex deletion problem corresponding to the family of distance hereditary graphs, or equivalently graphs of rankwidth 1.

All our algorithms follow the same recursive scheme: find "well structured balanced separators" in the graph by exploiting the properties of the family $\mathcal{F}$, and then use the structure of the balanced separator to obtain a approximate solution. In the following, we first describe the methodology by which we design all these approximation algorithms. Then, we give a brief overview, consisting of known results and our contributions, for each problem we study. Let us also mention that these problems inherit the hardness of approximation of Vertex Cover via simple reductions. In particular, they do not admit a PTAS (polynomial time approximation scheme) unless P = NP.

Our Methods. Multicommodity max-flow min-cut theorems are a classical technique in designing approximation algorithms, which was pioneered by Leighton and Rao in their seminal paper [34]. This approach can be viewed as using balanced vertex (or edge) separators[2] in a graph to obtain a divide-and-conquer approximation algorithm. In a typical application, the optimum solution S, forms a balanced separator of the graph. Thus, the idea is to find a minimum cost balanced separator W of the graph and add it to the solution, and then recursively solve the problem on each of the connected components. This leads to an $O(\log^{O(1)} n)$-factor approximation algorithm for the problem in question.

Our recursive scheme is a strengthening of this approach which exploits the structural properties of the family $\mathcal{F}$. Here the optimum solution S^* need not be a balanced separator of the graph. Indeed, a balanced separator of the graph could be much larger than S^*. Rather, S^* along with a possibly large but well-structured subset of vertices X, forms a balanced separator of the graph. We then exploit the presence of such a balanced separator in the graph to compute an approximate solution. Consider a family $\mathcal{F}$ for which Weighted $\mathcal{F}$ Vertex Deletion is amenable to our approach, and let G be an instance of this problem. Let S be the approximate solution that we will compute. Our approximation algorithm has the following steps:

(1) Find a well-structured set X, such that $G - X$ has a balanced separator W which is not too costly.
(2) Next, compute the balanced separator W of $G - X$ using the known factor $O(\sqrt{\log n})$-approximation algorithm (or deterministic $O(\log n)$-approximation algorithm) for Weighted Vertex Separators [12, 34]. Then add W into the solution set S and recursively solve the

[2]Roughly speaking, a *balanced vertex separator* is a set of vertices W, such that any connected component of $G - W$ contains at most $\frac{2}{3}$ of the vertices of G.

problem on each connected component of $G-(X\cup S)$. Let $S_1,\cdots,S_\ell$ be the solutions returned by the recursive calls. We add $S_1,\cdots,S_\ell$ to the solution S.

(3) Finally, we add X back into the graph and consider the instance $(G-S)\cup X$. Observe that, $V(G-S)$ can be partitioned into $V'\uplus X$, where $G[V']$ belongs to $\mathcal{F}$ and X is a well-structured set. We call such instances, the *special case* of Weighted $\mathcal{F}$ Vertex Deletion. We apply an approximation algorithm that exploits the structural properties of the special case to compute a solution.

Now consider the problem of finding the structured X. One way is to enumerate all the candidates for X and then pick the one where $G-X$ has a balanced vertex separator of least cost — this separator plays the role of W. However, the number of candidates for X in a graph could be too many to enumerate in polynomial time. For example, in the case of Weighted Chordal Vertex Deletion, the set X will be a clique in the graph, and the number of maximal cliques in a graph on n vertices could be as many as $3^{\frac{n}{3}}$ [37]. Hence, we cannot enumerate and test every candidate structure in polynomial time. However, we can exploit certain structural properties of family $\mathcal{F}$, to reduce the number of candidates for X in the graph. In our problems, we "tidy up" the graph by removing "short obstructions" that forbid the graph from belonging to the family $\mathcal{F}$. Then one can obtain an upper bound on the number of candidate structures. In the above example, recall that a graph G is chordal if and only if there are no induced cycles of length 4 or more. It is known that a graph G without any induced cycle of length 4 has at most $O(n^2)$ maximal cliques [11]. Observe that, we can greedily compute a set of vertices which intersects all induced cycles of length 4 in the graph. Therefore, at the cost of factor 4 in the approximation ratio, we can ensure that the graph has only polynomially many maximal cliques. Hence, one can enumerate all maximal cliques in the remaining graph [48] to test for X.

Next consider the task of solving an instance of the special case of the problem. We again apply a recursive scheme, but now with the advantage of a much more structured graph. By a careful modification of an LP solution to the instance, we eventually reduce it to instances of Weighted Multicut. In the above example, for Weighted Chordal Vertex Deletion we obtain instances of Weighted Multicut on a chordal graph. We note that a similar approach was also used by Jansen and Pilipczuck [28] to obtain an $O(\text{opt}^3\cdot\log^2\text{opt})$-approximation for Weighted Chordal Vertex Deletion, which may be considered as a starting point for our work. We follow the approach we described for all three problems that we study in this paper. We believe our recursive scheme can be applied to obtain $O(\log^{O(1)} n)$-approximation algorithms for Weighted $\mathcal{F}$ Vertex (Edge) Deletion corresponding to several other graph families $\mathcal{F}$.

Weighted Planar $\mathscr{F}$-Minor-Free Deletion. Let $\mathscr{F}$ be a finite set of graphs containing a planar graph. Formally, Weighted Planar $\mathscr{F}$-Minor-Free Deletion is defined as follows.

Weighted Planar $\mathscr{F}$-Minor-Free Deletion (WP$\mathscr{F}$-MFD)
Input: An undirected graph G and a weight function $w: V(G)\rightarrow\mathbb{R}$.
Question: Find a minimum weight subset $S\subseteq V(G)$ such that $G-S$ does not contain any graph in $\mathscr{F}$ as a minor.

The WP$\mathscr{F}$-MFD problem is a very generic problem that encompasses several known problems. To explain the versatility of the problem, we require a few definitions. A graph H is called a *minor* of a graph G if we can obtain H from G by a sequence of vertex deletions, edge deletions and edge contractions, and a family of graphs $\mathcal{F}$ is called *minor closed* if $G\in\mathcal{F}$ implies that every minor of G is also in $\mathcal{F}$. Given a graph family $\mathcal{F}$, by $\mathsf{ForbidMinor}(\mathcal{F})$ we denote the family of graphs such that $G\in\mathcal{F}$ if and only if G does not contain any graph in $\mathsf{ForbidMinor}(\mathcal{F})$ as a minor. By the celebrated Graph Minor Theorem of Robertson and Seymour, every minor closed family $\mathcal{F}$ is characterized by a finite family of forbidden minors [45]. That is, ForbidMinor($\mathcal{F}$) has finite size. Indeed, the size of ForbidMinor($\mathcal{F}$) depends on the family $\mathcal{F}$. Now for a finite collection of graphs $\mathscr{F}$, as above, we may define the Weighted $\mathscr{F}$-Minor-Free Deletion problem. And observe that, even though the definition of Weighted $\mathscr{F}$-Minor-Free Deletion we only consider finite sized $\mathscr{F}$, this problem actually encompasses deletion to every minor closed family of graphs. Let $\mathscr{G}$ be the set of all finite undirected graphs, and let $\mathscr{L}$ be the family of all finite subsets of $\mathscr{G}$. Thus, every element $\mathscr{F} \in \mathscr{L}$ is a finite set of graphs, and throughout the paper we assume that $\mathscr{F}$ is explicitly given. In this paper, we show that when $\mathscr{F} \in \mathscr{L}$ contains at least one planar graph, then it is possible to obtain an $O(\log^{O(1)} n)$-factor approximation algorithm for WP$\mathscr{F}$-MFD.

The case where $\mathscr{F}$ contains a planar graph, while being considerably more restricted than the general case, already encompasses a number of the well-studied instances of WP$\mathscr{F}$-MFD. For example, when $\mathscr{F} = \{K_2\}$, a complete graph on two vertices, this is the Weighted Vertex Cover problem. When $\mathcal{F} = \{C_3\}$, a cycle on three vertices, this is the Weighted Feedback Vertex Set problem. Another fundamental problem, which is also a special case of WP$\mathscr{F}$-MFD, is Weighted Treewidth-η Vertex Deletion or Weighted η-Transversal. Here the task is to delete a minimum weight vertex subset to obtain a graph of treewidth at most η. Since any graph of treewidth η excludes a $(\eta + 1) \times (\eta + 1)$ grid as a minor, we have that the set $\mathscr{F}$ of forbidden minors of treewidth η graphs contains a planar graph. Treewidth-η Vertex Deletion plays an important role in generic efficient polynomial time approximation schemes based on Bidimensionality theory [16, 17]. Among other examples of Planar $\mathscr{F}$-Minor-Free Deletion problems that can be found in the literature on approximation and parameterized algorithms, are the cases of $\mathscr{F}$ being $\{K_{2,3}, K_4\}$, $\{K_4\}$, $\{\theta_c\}$, and $\{K_3, T_2\}$,[3] which correspond to removing vertices to obtain an outerplanar graph, a series-parallel graph, a diamond graph, and a graph of pathwidth 1, respectively.

Apart from the case of Weighted Vertex Cover [4, 38] and Weighted Feedback Vertex Set [2, 5], there was not much progress on approximability/non-approximability of WP$\mathscr{F}$-MFD until the work of Fiorini, Joret, and Pietropaoli [13], which gave a constant factor approximation algorithm for the case of WP$\mathscr{F}$-MFD where $\mathscr{F}$ is a diamond graph, i.e., a graph with two vertices and three parallel edges. In 2011, Fomin et al. [14] considered Planar $\mathscr{F}$-Minor-Free Deletion (i.e. the unweighted version of WP$\mathscr{F}$-MFD) in full generality and designed a randomized (deterministic) $O(\log^{1.5} n)$-factor ($O(\log^2 n)$-factor) approximation algorithm for it. Later, Fomin et al. [15] gave a randomized constant factor approximation algorithm for Planar $\mathscr{F}$-Minor-Free Deletion. Very recently, Gupta et al. [23] have given $O(\log \ell)$ approximation algorithm for (unweighted) Planar $\mathscr{F}$-Minor-Free Deletion, where ℓ is the maximum number of vertices in any planar graph in $\mathscr{F}$. Our algorithm for WP$\mathscr{F}$-MFD extends these results to the weighted setting, at the cost of increasing the approximation factor to $\log^{O(1)} n$.

Theorem 1.1. *For every set $\mathscr{F} \in \mathscr{L}$, where $\mathscr{F}$ contains at least one planar graph,* WP$\mathscr{F}$-MFD *admits a randomized (deterministic) $O(\log^{1.5} n)$-factor ($O(\log^2 n)$-factor) approximation algorithm running in time n^c, where $c = c(\mathscr{F})$ is a constant depending on $\mathscr{F}$.*

We mention some recent related works. Bansal et al. [3] have studied the edge deletion version of the Treewidth-η Vertex Deletion problem, under the name Bounded Treewidth Interdiction Problem, and gave a bicriteria approximation algorithm. In particular, for a graph G and an integer $\eta > 0$, they gave a polynomial time algorithm that finds a subset of edges F' of G such that $|F'| \leq O((\log n \log\log n) \cdot \text{opt})$ and the treewidth of $G - F'$ is $O(\eta \log \eta)$.[4] In our setting where η is

[3] T_2 is the star-composition of (three) edges, i.e. a vertex attached to three (vertex disjoint) edges (see Theorem 2.5 of [47] for more details).

[4] The size of F' and the running time of the algorithm are independent of η.

a fixed constant, this immediately implies a factor $O(\log n \log\log n)$ approximation algorithm for the edge deletion version of WP$\mathscr{F}$-MFD.[5] However, it is not immediately clear if their approach can be extended to WP$\mathscr{F}$-MFD.[6]

Weighted Chordal Vertex Deletion. Formally, the Weighted Chordal Vertex Deletion problem is defined as follows.

Weighted Chordal Vertex Deletion (WCVD)
Input: An undirected graph G and a weight function $w : V(G) \to \mathbb{R}$.
Question: Find a minimum weight subset $S \subseteq V(G)$ such that $G - S$ is a chordal graph.

The class of chordal graphs is a natural class of graphs that has been extensively studied from the viewpoints of Graph Theory and Algorithm Design. Many important problems that are NP-hard on general graphs, such as Independent Set, and Graph Coloring are solvable in polynomial time once restricted to the class of chordal graphs [22]. Recall that a graph is chordal if and only if it does not have any induced cycle of length 4 or more. Thus, Chordal Vertex Deletion (CVD) can be viewed as a natural variant of the classic Feedback Vertex Set (FVS). Indeed, while the objective of FVS is to eliminate all cycles, the CVD problem only asks us to eliminate induced cycles of length 4 or more. Despite the apparent similarity between the objectives of these two problems, the design of approximation algorithms for WCVD is very challenging. In particular, chordal graphs can be dense—indeed, a clique is a chordal graph. As we cannot rely on the sparsity of output, our approach must deviate from those employed by approximation algorithms from FVS. That being said, chordal graphs still retain some properties that resemble those of trees, and these properties are utilized by our algorithm.

Prior to our work, only two non-trivial approximation algorithms for CVD were known. The first one, by Jansen and Pilipczuk [27], is a deterministic $O(\text{opt}^2 \log \text{opt} \log n)$-factor approximation algorithm, and the second one, by Agrawal et al. [1], is a deterministic $O(\text{opt} \log^2 n)$-factor approximation algorithm. The second result implies that CVD admits an $O(\sqrt{n} \log n)$-factor approximation algorithm.[7] After our work, Kim and Kwon [31] obtained an $O(\text{opt} \log \text{opt})$-approximation for CVD. In this paper we obtain the first $O(\log^{O(1)} n)$-approximation algorithm for WCVD.

Theorem 1.2. *WCVD admits a deterministic $O(\log^2 n)$-factor approximation algorithm.*

While this approximation algorithm follows our general scheme, it also requires us to incorporate several new ideas. In particular, to implement the third step of the scheme, we need to design a different $O(\log n)$-factor approximation algorithm for the special case of WCVD where the vertex-set of the input graph G can be partitioned into two sets, X and $V(G) \setminus X$, such that $G[X]$ is a clique and $G[V(G) \setminus X]$ is a chordal graph. This approximation algorithm is again based on recursion, but it is more involved. At each recursive call, it carefully manipulates a fractional solution of a special form. Moreover, to ensure that its current problem instance is divided into two subinstances that are independent and simpler than their origin, we introduce multicut constraints. In addition to these constraints, we keep track of the complexity of the subinstances, which is measured via the cardinality of the maximum independent set in the graph. Our multicut constraints result in an

[5]One can run their algorithm first and remove the solution output by their algorithm to obtain a graph of treewidth at most $O(\eta \log \eta)$. Then one can find an optimal solution using standard dynamic programming running in time $f(c) \cdot n$, where f is a computable function, $\eta = c = c(\mathscr{F})$ is some constant depending on $\mathscr{F}$.

[6]We thank Nikhil Bansal and Seeun William Umboh for several discussions and for pointing us that their algorithm does not work for WP$\mathscr{F}$-MFD.

[7]If $\text{opt} \geq \sqrt{n}/\log n$, we output a greedy solution to the input graph, and otherwise we have that $\text{opt} \log^2 n \leq \sqrt{n} \log n$, hence we call the $O(\text{opt} \log^2 n)$-factor approximation algorithm.

instance of Weighted Multicut, which we ensure is on a chordal graph. Formally, Weighted Multicut is defined as follows.

Weighted Multicut
Input: An undirected graph G, a weight function $w : V(G) \longrightarrow \mathbb{R}$ and a set $T = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs of (not necessarily distinct) vertices of G.
Question: Find a minimum weight subset $S \subseteq V(G)$ such that for any pair $(s_i, t_i) \in \mathcal{T}$, $G - S$ does not have any path between s_i and t_i.

We note that Weighted Multicut [7] on general graphs is NP-hard to approximate within every constant factor, assuming the Unique Games Conjecture of Khot [30]. Also, the problem is NP-hard on trees [19], and hence it is also NP-hard on chordal graphs. For Weighted Multicut on chordal graphs, no constant-factor approximation algorithm was previously known. We design the first such algorithm, which our main algorithm employs as a black box.

Theorem 1.3. Weighted Multicut *admits a constant-factor approximation algorithm on chordal graphs.*

This algorithm is inspired by the work of Garg, Vazirani and Yannakakis on Weighted Multicut on trees [19]. Here, we carefully exploit the well-known characterization of the class of chordal graphs as the class of graphs that admit clique forests. We believe that this result is of independent interest. The algorithm by Garg, Vazirani and Yannakakis [19] is a classic primal-dual algorithm. A more recent algorithm, by Golovin, Nagarajan and Singh [21], uses total unimodularity to obtain a different algorithm for Multicut on trees.

Weighted Distance Hereditary Vertex Deletion. We start by formally defining the Weighted Distance Hereditary Vertex Deletion problem.

Weighted Distance Hereditary Vertex Deletion (WDHVD)
Input: An undirected graph G and a weight function $w : V(G) \longrightarrow \mathbb{R}$.
Question: Find a minimum weight subset $S \subseteq V(G)$ such that $G - S$ is a distance hereditary graph.

A graph G is a *distance hereditary graph* (also called a completely separable graph [24]) if the distances between vertices in every connected induced subgraph of G are the same as in the graph G. Distance hereditary graphs were named and first studied by Hworka [26]. However, an equivalent family of graphs was earlier studied by Olaru and Sachs [46] and shown to be perfect. It was later discovered that these graphs are precisely the graphs of rankwidth 1 [39].

Rankwidth is a graph parameter introduced by Oum and Seymour [42] to approximate yet another graph parameter called Cliquewidth. The notion of cliquewidth was defined by Courcelle and Olariu [9] as a measure of how "clique-like" the input graph is. This is similar to the notion of treewidth, which measures how "tree-like" the input graph is. One of the main motivations was that several NP-complete problems become tractable on the family of cliques (complete graphs), the assumption was that these algorithmic properties extend to "clique-like" graphs [8]. However, computing cliquewidth and the corresponding cliquewidth decomposition seems to be computationally intractable. This then motivated the notion of rankwidth, which is a graph parameter that approximates cliquewidth well while also being algorithmically tractable [40, 42]. For more information on cliquewidth and rankwidth, we refer to the surveys by Hlinený et al. [25] and Oum [41].

As algorithms for Treewidth-η Vertex Deletion are applied as subroutines to solve many graph problems, we believe that algorithms for Weighted Rankwidth-η Vertex Deletion (WR-ηVD) will be useful in this respect. In particular, Treewidth-η Vertex Deletion has been considered in designing efficient approximation, kernelization and fixed parameter tractable algorithms for WP$\mathscr{F}$-MFD and its unweighted counterpart Planar $\mathscr{F}$-Minor-Free Deletion [3, 14, 16–18]. Along similar lines, we believe that WR-ηVD and its unweighted counterpart will be useful in designing efficient approximation, kernelization and fixed parameter tractable algorithms for Weighted $\mathcal{F}$ Vertex Deletion where $\mathcal{F}$ is characterized by a finite family of forbidden *vertex minors* [39].

Recently, Kim and Kwon [32] designed an $O(\text{opt}^2 \log n)$-factor approximation algorithm for Distance Hereditary Vertex Deletion (DHVD). This result implies that DHVD admits an $O(n^{2/3} \log n)$-factor approximation algorithm. In this paper, we take first step towards obtaining a good approximation algorithm for WR-ηVD by designing a $O(\log^{O(1)} n)$-factor approximation algorithm for WDHVD.

Theorem 1.4. *WDHVD or WR-1VD admits an $O(\log^3 n)$-factor approximation algorithm.*

We note that several steps of our approximation algorithm for WR-1VD can be generalized for an approximation algorithm for WR-ηVD and thus we believe that our approach should yield an $O(\log^{O(1)} n)$-factor approximation algorithm for WR-ηVD. We leave that as an interesting open problem for the future.

2 PRELIMINARIES

For a positive integer k, we use $[k]$ as a shorthand for $\{1, 2, \dots, k\}$. Given a function $f : A \rightarrow B$ and a subset $A' \subseteq A$, we let $f|_{A'}$ denote the function f restricted to the domain A'.

Graphs. Given a graph G, we let $V(G)$ and $E(G)$ denote its vertex-set and edge-set, respectively. In this paper, we only consider undirected graphs. We let $n = |V(G)|$ denote the number of vertices in the graph G, where G will be clear from context. The *open neighborhood*, or simply the *neighborhood*, of a vertex $v \in V(G)$ is defined as $N_G(v) = \{w \mid \{v, w\} \in E(G)\}$. The *closed neighborhood* of v is defined as $N_G[v] = N_G(v) \cup \{v\}$. The *degree* of v is defined as $d_G(v) = |N_G(v)|$. We can extend the definition of the neighborhood of a vertex to a set of vertices as follows. Given a subset $U \subseteq V(G)$, $N_G(U) = \bigcup_{u \in U} N_G(u)$ and $N_G[U] = \bigcup_{u \in U} N_G[u]$. The *induced subgraph* $G[U]$ is the graph with vertex-set U and edge-set $\{\{u, u'\} \mid u, u' \in U, \text{ and } \{u, u'\} \in E(G)\}$. Moreover, we define $G - U$ as the induced subgraph $G[V(G) \setminus U]$. We omit subscripts when the graph G is clear from context. For graphs G and H, by $G \cap H$, we denote the graph with vertex set as $V(G) \cap V(H)$ and edge set as $E(G) \cap E(H)$. An *independent set* in G is a set of vertices such that there is no edge in G between any pair of vertices in this set. The *independence number* of G, denoted by $\alpha(G)$, is defined as the cardinality of the largest independent set in G. A *clique* in G is a set of vertices such that there is an edge in G between every pair of vertices in this set.

A *path* $P = (x_1, x_2, \dots, x_\ell)$ in G is a subgraph of G where $V(P) = \{x_1, x_2, \dots, x_\ell\} \subseteq V(G)$ is a set of distinct vertices and $E(P) = \{\{x_1, x_2\}, \{x_2, x_3\}, \dots, \{x_{\ell-1}, x_\ell\}\} \subseteq E(G)$, where $\ell \in [n]$. The vertices x_1 and x_ℓ are called the *endpoints* of the path P and the remaining vertices in $V(P)$ are called the *internal vertices* of P. We also say that P is a path between x_1 and x_ℓ or connects x_1 and x_ℓ. A *cycle* $C = (x_1, x_2, \dots, x_\ell)$ in G is a subgraph of G where $V(C) = \{x_1, x_2, \dots, x_\ell\} \subseteq V(G)$ and $E(C) = \{\{x_1, x_2\}, \{x_2, x_3\}, \dots, \{x_{\ell-1}, x_\ell\}, \{x_\ell, x_1\}\} \subseteq E(G)$, i.e., it is a path with an additional edge between x_1 and x_ℓ. The graph G is *connected* if there is a path between every pair of vertices in G, otherwise G is *disconnected.* A connected graph without any cycles is a *tree*, and a collection of vertex disjoint trees is a *forest.* A maximal connected subgraph of G is called a *connected component* of G. Given a function $f : V(G) \rightarrow \mathbb{R}$ and a subset $U \subseteq V(G)$, we denote $f(U) = \sum_{v \in U} f(v)$. Moreover, we say that a subset $U \subseteq V(G)$ is a *balanced separator for G* if for each connected component C in $G - U$, it holds that $|V(C)| \leq \frac{2}{3}|V(G)|$. We refer the reader to [10] for details on standard graph theoretic notations and terminologies that are not explicitly defined here.

Forest Decompositions. A *forest decomposition* of a graph G is a pair (F, β) where F is forest, and $\beta : V(F) \rightarrow 2^{V(G)}$ is a function that satisfies the following:

(i) $\bigcup_{v \in V(F)} \beta(v) = V(G)$;
(ii) for every edge $\{v, u\} \in E(G)$, there is a node $w \in V(F)$ such that $v, u \in \beta(w)$;
(iii) for every $v \in V(G)$, the collection of nodes $T_v = \{u \in V(F) \mid v \in \beta(u)\}$ is a subtree of F.

For $v \in V(F)$, we call $\beta(v)$ the *bag* of v, and for the sake of clarity of presentation, we sometimes use v and $\beta(v)$ interchangeably. We refer to the vertices in $V(F)$ as *nodes*. The *width* of a forest decomposition is the maximum size of a bag minus one. A *tree decomposition* is a forest decomposition where F is a tree. For a graph G, by $\text{tw}(G)$ we denote the minimum over all possible *tree decompositions* of G, the maximum size of a bag minus one in that *tree decomposition*.

Minors. Given a graph G and an edge $\{u, v\} \in E(G)$, the graph G/e denotes the graph obtained from G by contracting the edge $\{u, v\}$, that is, the vertices u, v are deleted from G and a new vertex $uv^\star$ is added to G which is adjacent to the all the neighbors of u, v previously in G (except for u, v). A graph H that is obtained by a sequence of edge contractions in G is said to be a contraction of G. A graph H is a *minor* of a G if H is the contraction of some subgraph of G. We say that a graph G is F-minor free when F is not a minor of G. Given a family $\mathcal{F}$ of graphs, we say that a graph G is $\mathcal{F}$-minor free, if for all $F \in \mathcal{F}$, F is not a minor of G. It is well known that if H is a minor of G, then $\text{tw}(H) \leq \text{tw}(G)$. It is well known that a graph is *planar* if it is $\{K_5, K_{3,3}\}$-minor free. Here, K_5 is a clique on 5 vertices and $K_{3,3}$ is a complete bipartite graph with both sides of bipartition having 3 vertices.

Chordal Graphs. Let G be a graph. For a cycle C on at least four vertices, we say that $\{u, v\} \in E(G)$ is a *chord* of C if $u, v \in V(C)$ but $\{u, v\} \notin E(C)$. A cycle C is *chordless* if it contains at least four vertices and has no chords. The graph G is a *chordal graph* if it has no chordless cycle as an induced subgraph. A *clique forest* of G is a forest decomposition of G where every bag is a maximal clique. The following lemma shows that the class of chordal graphs is exactly the class of graphs which have a clique forest.

Lemma 2.1 ([22]). *A graph G is a chordal graph if and only if G has a clique forest. Moreover, a clique forest of a chordal graph can be constructed in polynomial time.*

Given a subset $U \subseteq V(G)$, we say that U *intersects* a chordless cycle C in G if $U \cap V(C) \neq \emptyset$. Observe that if U *intersects* every chordless cycle of G, then $G - U$ is a chordal graph.

3 APPROXIMATION ALGORITHM FOR WP$\mathscr{F}$-MFD

In this section we prove Theorem 1.1. We can assume that the weight $w(v)$ of each vertex $v \in V(G)$ is positive, else we can insert v into any solution. Below we state a result from [43], which will be useful in our algorithm.

Proposition 3.1 ([43]). *Let $\mathscr{F}$ be a finite set of graphs such that $\mathscr{F}$ contains a planar graph. Then, any graph G that excludes any graph from $\mathscr{F}$ as a minor satisfies* $\text{tw}(G) \leq c = c(\mathscr{F})$.

We let $c = c(\mathscr{F})$ to be the constant returned by Proposition 3.1. The approximation algorithm for WP$\mathscr{F}$-MFD comprises of two components. The first component handles the special case where the vertex set of input graph G can be partitioned into two sets M and X such that $|M| \leq c + 1$ and $H = G[X]$ is an $\mathscr{F}$-minor free graph. We note that there can be edges between vertices in M and vertices in H. We show that for these special instances, in polynomial time we can compute the size of the optimum solution and a set realizing it.

The second component is a recursive algorithm that solves general instances of the problem. Here, we gradually disintegrate the general instance until it becomes an instance of the special type where we can resolve it in polynomial time. More precisely, for each guess of $c+1$ sized subgraph M of G, we find a small separator S (using an approximation algorithm) that together with M breaks the input graph into two graphs significantly smaller than their origin. It first removes $M \cup S$, and solves each of the two resulting subinstances by calling itself recursively; then, it inserts M back into the graph, and uses the solutions it obtained from the recursive calls to construct an instance of the special case which is then solved by the first component.

3.1 Constant sized graph + $\mathscr{F}$-minor free graph

We first handle the special case where the input graph G consists of a graph M of size at most $c+1$ and an $\mathscr{F}$-minor free graph H. We refer to this algorithm as Special-WP. More precisely, along with the input graph G and the weight function w, we are also given a graph M with at most $c+1$ vertices and an $\mathscr{F}$-minor free graph H such that $V(G) = V(M) \cup V(H)$, where the vertex-sets $V(M)$ and $V(H)$ are disjoint. Note that the edge-set $E(G)$ may contain edges between vertices in M and vertices in H. We will show that such instances may be solved optimally in polynomial time. We start with the following easy observation.

Observation 1. *Let G be a graph with $V(G) = X \uplus Y$, such that $|X| \leq c+1$ and $G[Y]$ is an $\mathscr{F}$-minor free graph. Then, the treewidth of G is at most $2c+1$.*

Lemma 3.2. *Let G be a graph of treewidth t with a non-negative weight function w on the vertices, and let $\mathscr{F}$ be a finite family of graphs. Then, one can compute a minimum weight vertex set S such that $G - S$ is $\mathscr{F}$-minor free, in time $f(q,t) \cdot n$, where n is the number of vertices in G and q is a constant that depends only on $\mathscr{F}$.*

Proof. This follows from the fact that finding such a set S is expressible as an MSO-optimization formula ϕ whose length, q, depends only on the family $\mathscr{F}$ [15]. Then, by Theorem 7 of [6], we can compute an optimal sized set S in time $f(q,t) \cdot n$. □

Now, we apply the above lemma to the graph G and the family $\mathscr{F}$, and obtain a minimum weight set S such that $G - S$ is $\mathscr{F}$-minor free.

3.2 General Graphs

We proceed to handle general instances by developing a $d \cdot \log^2 n$-factor approximation algorithm for WP$\mathscr{F}$-MFD, Gen-WP-APPROX, thus proving the correctness of Theorem 1.1. The exact value of the constant d will be determined later.

Recursion. We define each call to our algorithm Gen-WP-APPROX to be of the form (G', w'), where (G', w') is an instance of WP$\mathscr{F}$-MFD such that G' is an induced subgraph of G and $w' = w|_{V(G')}$. We denote $n' = |V(G')|$.

Goal. For each recursive call Gen-WP-APPROX(G', w'), we aim to prove the following. Recall that opt denotes the cost of the current instance, i.e. (G', w').

Lemma 3.3. Gen-WP-APPROX *returns a solution that is at least* opt *and at most* $\frac{d}{2} \cdot \log^2 n' \cdot$ opt. *Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At each recursive call, the size of the graph G' becomes smaller. Thus, when we prove that Lemma 3.3 is true for the current call, we assume that the approximation factor is bounded by $\frac{d}{2} \cdot \log^2 \widehat{n} \cdot \text{opt}$ for any call where the size $\widehat{n}$ of the vertex-set of its graph is strictly smaller than n'.

Termination. In time bounded by $O(\widehat{c}(\mathscr{F}) \cdot n^{O(1)})$, where $\widehat{c} = \widehat{c}(\mathscr{F})$ is a constant, we can test whether G' has a minor $F \in \mathscr{F}$ [44]. If G' has at most $10 \cdot (3c + 1)$ vertices, then we can output the size of an optimum solution and an optimum solution in time bounded by $O(2^{O(c)} \cdot \widehat{c} \cdot n^{O(1)})$. Hereafter we assume that G' has more than $10 \cdot (3c+1)$ vertices. For each $M \subseteq V(G)$ of size at most $c + 1$, we can check if $G - M$ has a minor $F \in \mathscr{F}$. If $G - M$ is $\mathscr{F}$-minor free then we are in a special instance, where $G - M$ is $\mathscr{F}$ minor free and M is a constant sized graph. We optimally resolve this instance in polynomial time using the algorithm Special-WP. Since we output an optimal sized solution in the base cases, we thus ensure that at the base case of our induction Lemma 3.3 holds.

Recursive Call. For the analysis of a recursive call, let S^* denote a hypothetical set that realizes the optimal solution opt of the current instance (G', w'). Let (F, β) be a forest decomposition of $G' - S^*$ of width at most c, whose existence is guaranteed by Proposition 3.1. Using standard arguments on forests we have the following observation.

Observation 2. *There exists a node $v \in V(F)$ such that $\beta(v)$ is a balanced separator for $G' - S^*$.*

From Observation 2 we know that there exists a node $v \in V(F)$ such that $\beta(v)$ is a balanced separator for $G' - S^*$. This together with the fact that $G' - S^*$ has treewidth at most c results in the following observation.

Observation 3. *There exist a subset $M \subseteq V(G')$ of size at most $c + 1$ and a subset $S \subseteq V(G') \setminus M$ of weight at most* opt *such that $M \cup S$ is a balanced separator for G'.*

This gives us a polynomial time algorithm as stated in the following lemma.

Lemma 3.4. *There is a deterministic (randomized) algorithm which in polynomial-time finds $M \subseteq V(G')$ of size at most $c+1$ and a subset $S \subseteq V(G') \setminus M$ of weight at most $q \cdot \log n' \cdot \text{opt}$ ($q^* \cdot \sqrt{\log n'} \cdot \text{opt}$) for some fixed constant q (q^*) such that $M \cup S$ is a balanced separator for G'.*

Proof. Note that we can enumerate every $M \subseteq V(G')$ of size at most $c+1$ in time $O(n^c)$. For each such M, we can either run the randomized $q^* \cdot \sqrt{\log n'}$-factor approximation algorithm by Feige et al. [12] or the deterministic $q \cdot \log n'$-factor approximation algorithm by Leighton and Rao [34] to find a balanced separator S_M of $G' - M$. Here, q and q^* are fixed constants. By Observation 3, there is a set S in $\{S_M : M \subseteq V(G') \text{ and } M \leq c+1\}$ such that $w(S) \leq q \cdot \log n' \cdot \text{opt}$ ($w(S) \leq q^* \cdot \sqrt{\log n'} \cdot \text{opt}$). Thus, the desired output is a pair (M, S) where M is one of the vertex subsets of size at most $c + 1$ such that $S_M = S$. □

We call the algorithm in Lemma 3.4 to obtain a pair (M, S). Since $M \cup S$ is a balanced separator for G', we can partition the set of connected components of $G' - (M \cup S)$ into two sets, $\mathscr{A}_1$ and $\mathscr{A}_2$, such that for $V_1 = \bigcup_{A \in \mathscr{A}_1} V(A)$ and $V_2 = \bigcup_{A \in \mathscr{A}_2} V(A)$ it holds that $n_1, n_2 \leq \frac{2}{3} n'$ where $n_1 = |V_1|$ and $n_2 = |V_2|$. We remark that we use different algorithms for finding a balanced separator in Lemma 3.4 based on whether we are looking for a randomized algorithm or a deterministic algorithm.

Next, we define two inputs of (the general case of) WP$\mathscr{F}$-MFD: $I_1 = (G'[V_1], w'|_{V_1})$ and $I_2 = (G'[V_2], w'|_{V_2})$. Let opt_1 and opt_2 denote the optimal solutions to I_1 and I_2, respectively. Observe that since $V_1 \cap V_2 = \emptyset$, it holds that $\text{opt}_1 + \text{opt}_2 \leq \text{opt}$. We solve each of the subinstances by recursively calling algorithm Gen-WP-APPROX. By the inductive hypothesis, we thus obtain two sets, S_1 and S_2, such that $G'[V_1] - S_1$ and $G'[V_2] - S_2$ are $\mathscr{F}$-minor free graphs, and $w'(S_1) \leq \frac{d}{2} \cdot \log^2 n_1 \cdot \text{opt}_1$ and $w'(S_2) \leq \frac{d}{2} \cdot \log^2 n_2 \cdot \text{opt}_2$.

We now define an input of the special case of WP$\mathscr{F}$-MFD: $J = (G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)], w'|_{(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)})$. Observe that $G'[V_1 \setminus S_1]$ and $G'[V_2 \setminus S_2]$ are $\mathscr{F}$-minor free graphs and there are no edges between vertices in V_1 and vertices in V_2 in $G' - M$, and M is of constant size. Therefore, we resolve this instance by calling algorithm Special-WP. We thus obtain a set, $\widehat{S}$, such that $G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a $\mathscr{F}$-minor free graph, and $w'(\widehat{S}) \leq \text{opt}$ (since $|(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)| \leq n'$ and the optimal solution of each of the special subinstances is at most opt).

Observe that any obstruction in $G' - S$ is either completely contained in $G'[V_1]$, or completely contained in $G'[V_2]$, or it contains at least one vertex from M. This observation, along with the fact that $G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a $\mathscr{F}$-minor free graph, implies that $G' - T$ is a $\mathscr{F}$-minor free graph where $T = S \cup S_1 \cup S_2 \cup \widehat{S}$. Thus, it is now sufficient to show that $w'(T) \leq \frac{d}{2} \cdot (\log n')^2 \cdot \text{opt}$.

By the discussion above, we have that

$$\begin{aligned} w'(T) &\leq w'(S) + w'(S_1) + w'(S_2) + w'(\widehat{S}) \\ &\leq q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot ((\log n_1)^2 \cdot \text{opt}_1 + (\log n_2)^2 \cdot \text{opt}_2) + \text{opt} \end{aligned}$$

Recall that $n_1, n_2 \leq \frac{2}{3}n'$ and $\text{opt}_1 + \text{opt}_2 \leq \text{opt}$. Thus, we have that

$$\begin{aligned} w'(T) &< q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot (\log \tfrac{2}{3}n')^2 \cdot \text{opt} + \text{opt} \\ &< \tfrac{d}{2} \cdot (\log n')^2 \cdot \text{opt} + \log n' \cdot \text{opt} \cdot (q + 1 + \tfrac{d}{2} \cdot (\log \tfrac{3}{2})^2 - \tfrac{d}{2} \cdot 2 \cdot \log \tfrac{3}{2}). \end{aligned}$$

Overall, we conclude that to ensure that $w'(T) \leq \frac{d}{2} \cdot \log^2 n' \cdot \text{opt}$, it is sufficient to ensure that $q + 1 + \frac{d}{2} \cdot (\log \frac{3}{2})^2 - \frac{d}{2} \cdot 2 \cdot \log \frac{3}{2} \leq 0$, which can be done by fixing $d = \dfrac{2}{2 \log \frac{3}{2} - (\log \frac{3}{2})^2} \cdot (q + 1)$.

If we use the $O(\sqrt{\log n})$-factor approximation algorithm by Feige et al. [12] for finding a balance separator in Lemma 3.4, then we can do the analysis similar to the deterministic case and obtain a randomized factor-$O(\log^{1.5} n)$ approximation algorithm for WP$\mathscr{F}$-MFD.

We now analyse the running time of the algorithm. For a constant $\widehat{c} = \widehat{c}(\mathscr{F})$, in time $O(\widehat{c}(\mathscr{F}) \cdot n^{O(1)})$, we can test whether or not G' has a minor $F \in \mathscr{F}$ [44]. If G' has no minor from $\mathscr{F}$, we can correctly return $\emptyset$ as the solution. If G' has at most $10 \cdot (3c + 1)$ vertices, then we can output the size of an optimum solution and an optimum solution in time bounded by $O(2^{O(c)} \cdot \widehat{c} \cdot n^{O(1)})$. Otherwise, the algorithm computes a balanced separator $M \cup S$, where $|M| \leq c + 1$. The above step can be executed in time bounded by $n^{O(c)}$ (see Lemma 3.4). The algorithm then creates two subinstance of the problem, $G'[V_1]$ and $G'[V_2]$, where $|V_1|, |V_2| \leq 2n/3 + c + 1 \leq 21n/30$, and resolves them recursively. Furthermore, the algorithm creates a special instance of the problems, and resolves it using the algorithm Special-WP. Thus for a constant $\widetilde{c} = \widetilde{c}(\mathscr{F})$, the running time of the algorithm can be bounded by $\mathbb{T}(n) \leq 2\mathbb{T}(2n/3) + \widetilde{c} \cdot n^{O(c)}$. Hence we can conclude that the algorithm runs in time bounded by $\widetilde{c} \cdot n^{O(c)}$.

4 WEIGHTED CHORDAL VERTEX DELETION ON GENERAL GRAPHS

In this section we prove Theorem 1.2. Clearly, we can assume that the weight $w(v)$ of each vertex $v \in V(G)$ is positive, else we can insert v into any solution.

Roughly speaking, our approximation algorithm consists of two components. The first component handles the special case where the input graph G consists of a clique C and a chordal graph H. Here, we also assume that the input graph has no "short" chordless cycle. This component is comprised of a recursive algorithm that is based on the method of divide and conquer. The algorithm keeps track of a fractional solution $\mathbf{x}$ of a special form that it carefully manipulated at each recursive call, and which is used to analyze the approximation ratio. In particular, we ensure that $\mathbf{x}$ does not assign high values, and that it assigns 0 to vertices of the clique C as well as vertices of some other cliques.

To divide a problem instance into two instances, we find a maximal clique M of the chordal graph H that breaks H into two "simpler" chordal graphs. The clique C remains intact at each recursive call, and the maximal clique M is also a part of both of the resulting instances. Thus, to ensure that we have simplified the problem, we measure the complexity of instances by examining the maximum size of an independent set of their graphs. Since the input graph has no "short" chordless cycle, the maximum depth of the recursion tree is bounded by $O(\log n)$. Moreover, to guarantee that we obtain instances that are independent, we incorporate multicut constraints while ensuring that we have sufficient "budget" to satisfy them. We ensure that these multicut constraints are associated with chordal graphs, which allows us to utilize the algorithm we design in Section 5.

The second component is a recursive algorithm that solves general instances of the problem. Initially, it easily handles "short" chordless cycles. Then, it gradually disintegrates a general instance until it becomes an instance of the special form that can be solved using the first component. More precisely, given a problem instance, the algorithm divides it by finding a maximal clique M (using an exhaustive search which relies on the guarantee that G has no "short" chordless cycle) and a small separator S (using an approximation algorithm) that together break the input graph into two graphs significantly smaller than the original graph. It first removes $M \cup S$ and solves each of the two resulting subinstances by calling itself recursively; then, it inserts M back into the graph, and uses the solutions it obtained from the recursive calls to construct an instance of the special case solved by the first component.

4.1 Clique+Chordal Graphs

In this subsection we handle the special case where the input graph G consists of a clique C and a chordal graph H. More precisely, along with the input graph G and the weight function w, we are also given a clique C and a chordal graph H such that $V(G) = V(C) \cup V(H)$, where the vertex-sets $V(C)$ and $V(H)$ are disjoint. Here, we also assume that G has no chordless cycle on at most 48 vertices. Note that the edge-set $E(G)$ may contain edges between vertices in C and vertices in H. We call this special case the *Clique+Chordal special case*. Our objective is to prove the following result.

Lemma 4.1. *The Clique+Chordal special case of WCVD admits an $O(\log n)$-factor approximation algorithm.*

We assume that $n \geq 64$,[1] else the input instance can be solve by brute-force. Let c be a fixed constant (to be determined). In the rest of this subsection, we design a $c \cdot \log n$-factor approximation algorithm for the Clique+Chordal special case of WCVD.

Recursion. Our approximation algorithm is a recursive algorithm, which we call CVD-APPROX, and define each call to be of the form $(G', w', C, H', \mathbf{x})$. Here, G' is an induced subgraph of G such that $V(C) \subseteq V(G')$, and H' is an induced subgraph of H, where $H' = G' - C$. The argument $\mathbf{x}$ is discussed below. We remark that we continue to use n to refer to the size of the vertex-set of the input graph G rather than the current graph G'.

Arguments. While the execution of our algorithm progresses, we keep track of two arguments: the size of a maximum independent set of the current graph G', denoted by $\alpha(G')$, and a fractional solution $\mathbf{x}$. Due to the special structure of G', the computation of $\alpha(G')$ is simple:

Observation 4. *The measure $\alpha(G')$ can be computed in polynomial time.*

[1] This assumption simplifies some of the calculations ahead.

Proof. Any maximum independent set of G' consists of at most one vertex from C and an independent set of H'. It is well known that the computation of the size of a maximum independent set of a chordal graph can be performed in polynomial time [22]. Thus, we can compute $\alpha(H')$ in polynomial time. Next, we iterate over every vertex $v \in V(C)$, and we compute $\alpha_v = \alpha(\widehat{H}) + 1$ for the graph $\widehat{H} = H' \setminus N_{G'}(v)$ in polynomial time (since $\widehat{H}$ is a chordal graph). Overall, we return $\max\{\alpha(H'), \max_{v \in V(C)}\{\alpha_v\}\}$. □

The necessity of tracking $\alpha(G')$ stems from the fact that our recursive algorithm is based on the method of divide-and-conquer, and to ensure that when we divide the current instance into two instances we obtain two "simpler" instances, we need to argue that some aspect of these instances has indeed been simplified. Although this aspect cannot be the size of the instance (since the two instances can share many common vertices), we show that it can be the size of a maximum independent set.

A fractional solution $\mathbf{x}$ is a function $\mathbf{x} : V(G') \rightarrow [0, \infty)$ such that for every chordless cycle Q of G' it holds that $\mathbf{x}(V(Q)) \geq 1$. An optimal fractional solution minimizes the weight $w'(\mathbf{x}) = \sum_{v \in V(G')} w'(v) \cdot \mathbf{x}(v)$. Clearly, the solution to the instance (G', w') of WCVD is at least as large as the weight of an optimal fractional solution. Although we initially compute an optimal fractional solution $\mathbf{x}$ (at the initialization phase that is described below), during the execution of our algorithm, we manipulate this solution so it may no longer be optimal. Prior to any call to CVD-APPROX with the exception of the first call, we ensure that $\mathbf{x}$ satisfies the following invariants:

- **Low-Value Invariant**: For any $v \in V(G')$, it holds that $\mathbf{x}(v) < (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta} \cdot \frac{1}{c \cdot \log n}$. Here, δ is the depth of the current recursive call in the recursion tree.[2]
- **Zero-Clique Invariant**: For any $v \in V(C)$, it holds that $\mathbf{x}(v) = 0$.

Goal. The depth of the recursion tree will be bounded by $q \cdot \log n$ for some fixed constant q. The correctness of this claim is proved when we explain how to perform a recursive call. For each recursive call CVD-APPROX $(G', w', C, H', \mathbf{x})$ with the exception of the first call, we aim to prove the following.

Lemma 4.2. *For any $\delta \in \{1, 2, \ldots, q \cdot \log n\}$, each recursive call to* CVD-APPROX *of depth $\delta \geq 2$ returns a solution that is at least* opt *and at most $(\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta - 1} \cdot c \cdot \log(\alpha(G')) \cdot w'(\mathbf{x})$. Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At the initialization phase, we see that in order to prove Lemma 4.1, it is sufficient to prove Lemma 4.2.

Initialization. Initially, the graphs G' and H' are simply set to be the input graphs G and H, and the weight function w' is simply set to be input weight function w. Moreover, we compute an optimal fractional solution $\mathbf{x} = \mathbf{x}_{\text{init}}$ by using the ellipsoid method [29].[3] Recall that the following claim holds.

Observation 5. *The solution of the instance (G', w') of WCVD is lower bounded by $w'(\mathbf{x}_{\text{init}})$.*

[2]The depth of the first call is defined to be 1.

[3]The ellipsoid method for solving Linear Programming runs (in polynomial time) independent of the number of constraints (where constraints are implicitly given), if there is a separation oracle (for more details, see, [20]). It is a folklore result that Chordal Vertex Deletion admits a separation oracle. Roughly speaking, given a graph G and an assignment to its vertices, the objective of the separation oracle for Chordal Vertex Deletion is to find a chordless cycle (if it exists) of length at least 4, for which the sum of the values assigned to the vertices of the cycle is less than 1.

Thus, to prove Lemma 4.1, it is sufficient to return a solution that is at least opt and at most $c \cdot \log n \cdot w'(\mathbf{x})$. We would like to proceed by calling our algorithm recursively. For this purpose, we first need to ensure that $\mathbf{x}$ satisfies the low-value and zero-clique invariants, to which end we use the following notation. We let $h(\mathbf{x}) = \{v \in V(G') : \mathbf{x}(v) \geq 1/(c \cdot \log n)\}$ denote the set of vertices to which $\mathbf{x}$ assigns high values. Moreover, given a clique M in G', we let $(\mathbf{x} \setminus M) : V(G') \to [0, \infty)$ denote the function that assigns 0 to any vertex in M and $(1 + 3 \cdot \max_{u \in V(G')} \{\mathbf{x}(u)\})\mathbf{x}(v)$ to any other vertex $v \in V(G')$. Now, to adjust $\mathbf{x}$ to be of the desired form both at this phase and at later recursive calls, we rely on the two following lemmata.

Lemma 4.3. *Define $\widehat{G} = G' - h(\mathbf{x})$, $\widehat{w} = w'|_{V(\widehat{G})}$ and $\widehat{\mathbf{x}} = \mathbf{x}|_{V(\widehat{G})}$. Then, $c \cdot \log n \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c \cdot \log n \cdot w'(\mathbf{x})$.*

Proof. By the definition of $h(\mathbf{x})$, it holds that $w'(\widehat{\mathbf{x}}) \leq w'(\mathbf{x}) - \frac{1}{c \cdot \log n} \cdot w'(h(\mathbf{x}))$. Thus, $c \cdot \log n \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c \cdot \log n \cdot w'(\mathbf{x})$. □

Thus, it is safe to update G' to $G' - h(\mathbf{x})$, w' to $w'|_{V(\widehat{G})}$, H' to $H' - h(\mathbf{x})$ and $\mathbf{x}$ to $\mathbf{x}|_{V(\widehat{G})}$, where we ensure that once we obtain a solution to the new instance, we add $w'(h(\mathbf{x}))$ to this solution and $h(\mathbf{x})$ to the set realizing it.

Lemma 4.4. *Given a clique M in G', the function $(\mathbf{x} \setminus M)$ is a valid fractional solution such that $w'(\mathbf{x} \setminus M) \leq (1 + 3 \cdot \max_{v \in V(G')} \{\mathbf{x}(v)\}) w'(\mathbf{x})$.*

Proof. To prove that $(\mathbf{x} \setminus M)$ is a valid fractional solution, let Q be some chordless cycle in G'. We need to show that $(\mathbf{x} \setminus M)(V(Q)) \geq 1$. Since M is a clique, Q can contain at most two vertices from M. Thus, since $\mathbf{x}$ is a valid fractional solution, it holds that $\mathbf{x}(V(Q) \setminus V(M)) \geq 1 - 2 \cdot \max_{u \in V(G')} \{\mathbf{x}(u)\}$. By the definition of $(\mathbf{x} \setminus M)$, this fact implies that $(\mathbf{x} \setminus M)(V(Q)) = (\mathbf{x} \setminus M)(V(Q) \setminus V(M)) \geq (1 + 3 \cdot \max_{u \in V(G')} \{\mathbf{x}(u)\})(1 - 2 \cdot \max_{u \in V(G')} \{\mathbf{x}(u)\})$. As we have removed vertices in $h(\mathbf{x})$ from G', we have $\max_{u \in V(G')} \{\mathbf{x}(u)\} \leq 1/(c \cdot \log n)$. Thus we can obtain that $(\mathbf{x} \setminus M)(V(Q)) \geq \min\{(1 + \frac{3}{c \cdot \log n})(1 - \frac{2}{c \cdot \log n}), 1\} = \min\{1 + 1/(c \cdot \log n) - 6/((c \cdot \log n)^2), 1\} \geq 1$, where the last inequality relies on the assumption $n \geq 64$.

For the proof of the second part of the claim, note that $w'(\mathbf{x} \setminus M) = (1 + 3 \cdot \max_{v \in V(G')} \{\mathbf{x}(v)\}) w'(\mathbf{x}|_{V(G') \setminus V(M)}) \leq (1 + 3 \cdot \max_{v \in V(G')} \{\mathbf{x}(v)\}) w'(\mathbf{x})$. □

Next, it is possible to call CVD-APPROX recursively with the fractional solution $(\mathbf{x} \setminus C)$. In the context of the low-value invariant, observe that indeed, for any $v \in V(G')$, it now holds that $(\mathbf{x} \setminus C)(v) = (1 + 3 \cdot \max_{u \in V(G')} \{\mathbf{x}(u)\})\mathbf{x}(v) < (1 + \frac{3}{c \cdot \log n}) \cdot \frac{1}{c \cdot \log n} < (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta} \cdot \frac{1}{c \cdot \log n}$ for $\delta = 1$. Similarly, by Lemma 4.4, $w'(\mathbf{x} \setminus C) \leq (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta} \cdot w'(\mathbf{x})$ for $\delta = 1$. It is also clear that $\alpha(G') \leq n$. Thus, if Lemma 4.2 is true, we return a solution that is at least opt and at most $c \cdot \log n \cdot w(\mathbf{x})$ as desired. In other words, to prove Lemma 4.1, it is sufficient that we next focus only on the proof of Lemma 4.2. The proof of this lemma is done by induction. When we consider some recursive call, we assume that the solutions returned by the additional recursive calls that it performs, which are associated with graphs $\widetilde{G}$ such that $\alpha(\widetilde{G}) \leq \frac{3}{4}\alpha(G')$, comply with the demands of the lemma.

Termination. Once G' becomes a chordal graph, we return 0 as our solution and $\emptyset$ as the set that realizes it. Clearly, we thus satisfy the demands of Lemma 4.2. In fact, we thus also ensure that the execution of our algorithm terminates once $\alpha(G') < 24$:

Lemma 4.5. *If $\alpha(G') < 24$, then G' is a chordal graph.*

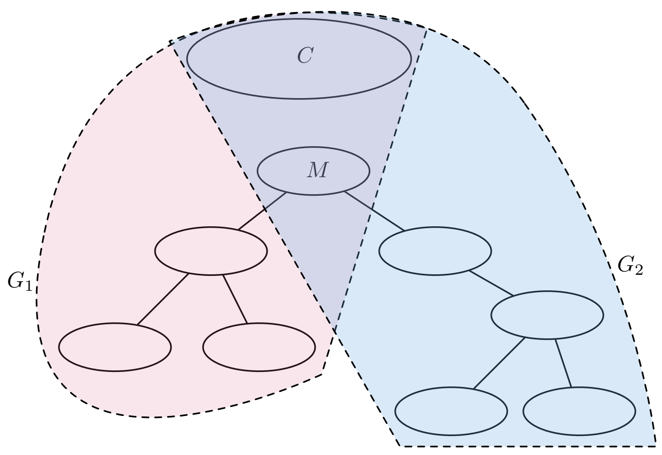

Fig. 1. Subinstances created by a recursive call

Proof. Suppose, by way of contradiction, that G' is not a chordal graph. Then, it contains a chordless cycle Q. Since G' is an induced subgraph of G, where G is assumed to exclude any chordless cycle on at most 48 vertices, we have that $|V(Q)| > 48$. Note that if we traverse Q in some direction, and insert every second vertex on Q into a set, excluding the last vertex in case $|V(Q)|$ is odd, we obtain an independent set. Thus, we have that $\alpha(G) \geq 24$, which is a contradiction. □

Thus, since we will ensure that each recursive call is associated with a graph whose independence number is at most 3/4 the independence number of the current graph, we have the following observation.

Observation 6. *The maximum depth of the recursion tree is bounded by $q \cdot \log n$ for some fixed constant q.*

Recursive Call. Since H' is a chordal graph, it admits a clique forest (Lemma 2.1). In particular, it contains only $O(n)$ maximal cliques, and one can find the set of these maximal cliques in polynomial time [22]. By standard arguments on trees, we deduce that H' has a maximal clique M such that after we remove M from G' we obtain two (not necessarily connected) graphs, $\widehat{H}_1$ and $\widehat{H}_2$, such that $\alpha(\widehat{H}_1), \alpha(\widehat{H}_2) \leq \frac{2}{3}\alpha(H')$, and that the clique M can be found in polynomial time. Let $G_1 = G'[V(\widehat{H}_1) \cup V(M) \cup V(C)]$, $H_1 = H'[V(\widehat{H}_1) \cup V(M)]$, $G_2 = G'[V(\widehat{H}_2) \cup V(M) \cup V(C)]$ and $H_2 = H'[V(\widehat{H}_2) \cup V(M)]$, and observe that $\alpha(G_1), \alpha(G_2) \leq \frac{2}{3}\alpha(G') + 2 \leq \frac{3}{4}\alpha(G')$. Here, the last inequality holds because $\alpha(G') \geq 24$, else by Lemma 4.5, the execution should have already terminated.

We proceed by replacing $\mathbf{x}$ by $(\mathbf{x} \setminus M)$. For the sake of clarity, we denote $\mathbf{x}^* = (\mathbf{x} \setminus M)$. We set $c \geq \max\{9q, 16\}$. In the next observation we prove the low-invariant property for $(\mathbf{x} \setminus M)$ (for the next depth of the recursion).

Observation 7. *For each $v \in V(G')$, we have $\mathbf{x}^*(v) < (\frac{c \cdot \log n + 9}{c \cdot \log n})^{\delta+1} \cdot \frac{1}{c \cdot \log n}$. Furthermore, $w'(\mathbf{x}^*) \leq (\frac{c \cdot \log n + 9}{c \cdot \log n}) \cdot w'(x)$.*

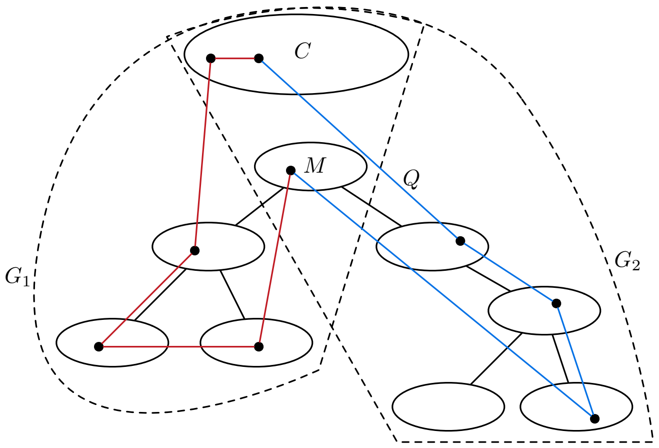

Fig. 2. An illustration of a bad cycle

Proof. By the low-value invariant, for each $v \in V(G')$, we have $\mathbf{x}(v) < (\frac{c\cdot\log n+9}{c\cdot\log n})^\delta \cdot \frac{1}{c\cdot\log n}$. Recall that $\mathbf{x}^*(v) = (1+3\cdot\max_{v\in V(G')}\{\mathbf{x}(v)\})\cdot\mathbf{x}(v)$. Thus, $\mathbf{x}^*(v) < (1+3\cdot(\frac{c\cdot\log n+9}{c\cdot\log n})^\delta\cdot\frac{1}{c\cdot\log n})\cdot\mathbf{x}(v)$. Note that by Observation 6 and the fact that $c \geq \max\{9q,16\}$, we have that $(\frac{c\cdot\log n+9}{c\cdot\log n})^\delta \leq e < 3$, and therefore $1+3\cdot(\frac{c\cdot\log n+9}{c\cdot\log n})^\delta\cdot\frac{1}{c\cdot\log n} \leq \frac{c\cdot\log n+9}{c\cdot\log n}$. Hence we can conclude that $\mathbf{x}^*(v) < (\frac{c\cdot\log n+9}{c\cdot\log n})^{\delta+1}\cdot\frac{1}{c\cdot\log n}$.

From Lemma 4.4 we have $w'(\mathbf{x}^*) \leq (1+3\cdot\max_{v\in V(G')}\{\mathbf{x}(v)\})w'(\mathbf{x})$. Note that $(1+3\cdot\max_{v\in V(G'))}\{\mathbf{x}(v)\})$ $\leq (1+3\cdot(\frac{c\cdot\log n+9}{c\cdot\log n})^\delta\cdot\frac{1}{c\cdot\log n}) \leq \frac{c\cdot\log n+9}{c\cdot\log n}$. Thus, $w'(\mathbf{x}^*) \leq (\frac{c\cdot\log n+9}{c\cdot\log n})\cdot w'(x)$. □

From the above observation, to prove Lemma 4.2, it is sufficient to return a solution that is at least opt and at most $(\frac{c\cdot\log n}{c\cdot\log n+9})^\delta\cdot\log\alpha(G')\cdot w(\mathbf{x}^*)$.

Next, we define two subinstances, $I_1 = (G_1, w|_{V(G_1)}, C, H_1, \mathbf{x}^*|_{V(G_1)})$ and $I_2 = (G_2, w|_{V(G_2)}, C, H_2, \mathbf{x}^*|_{V(G_2)})$ (see Figure 1). We solve these subinstances by recursive calls to CVD-APPROX (by the above discussion, these calls are valid — we satisfy the low-value and zero-clique invariants). Thus, we obtain two solutions, s_1 to I_1 and s_2 to I_2, and two sets that realize these solutions, S_1 and S_2. Let $\mathbf{x}_1^* = \mathbf{x}^*|_{V(G_1)})$ and $\mathbf{x}_2^* = \mathbf{x}^*|_{V(G_2)})$. By the inductive hypothesis, we have the following observations.

Observation 8. *$S_1 \cup S_2$ intersects any chordless cycle in G' that lies entirely in either G_1 or G_2.*

Observation 9. *Given $i \in \{1,2\}$, $s_i \leq (\frac{c\cdot\log n}{c\cdot\log n+9})^\delta\cdot c\cdot\log(\alpha(G_i))\cdot w(\mathbf{x}_i^*)$.*

Moreover, since $\mathbf{x}^*(V(C)\cup V(M)) = 0$, we also have the following observation.

Observation 10. $w(\mathbf{x}_1^*) + w(\mathbf{x}_2^*) = w(\mathbf{x}^*)$.

We say that a cycle of G' is *bad* if it is a chordless cycle that belongs entirely to neither G_1 nor G_2 (see Figure 2). Next, we show how to intersect bad cycles.

Bad Cycles. For any pair (v, u) of vertices, where $v \in V(C)$ and $u \in V(M)$, we let $\mathcal{P}_1(v, u)$ denote the set of any (simple) path P_1 between v and u whose internal vertices belong only to G_1 and $\{v, u\} \notin E(G')$. Symmetrically, we let $\mathcal{P}_2(v, u)$ denote the set of any path P_2 between v and u whose internal vertices belong only to G_2 and $\{v, u\} \notin E(G')$. We note here that when $\{v, u\} \in E(G')$ then $\mathcal{P}_1(v, u) = \mathcal{P}_2(v, u) = \emptyset$.

We first examine the relation between bad cycles and pairs (v, u) of vertices $v \in V(C)$ and $u \in V(M)$.

Lemma 4.6. *For any bad cycle Q there are pairs $(v, u), (v', u')$ of vertices (possibly $v = v'$ or $u = u'$), where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, $\{v, v'\} \cap N_{G'}(u') = \emptyset$, and there are paths $P_1 \in \mathcal{P}_1(v, u)$ and $P_2 \in \mathcal{P}_2(v', u')$ such that $V(P_1) \subseteq V(Q)$ and $V(P_2) \subseteq V(Q)$.*

Proof. Let Q be some bad cycle. By the definition of a bad cycle, Q must contain at least one vertex a from $H_1 \setminus V(M)$ and at least one vertex b from $H_2 \setminus V(M)$. Since C and M are cliques, Q can contain at most two vertices from C and at most two vertices from M, and if it contains two vertices from C (resp. M), then these two vertices are neighbors. As the set $V(C) \cup V(M)$ contains all vertices common to G_1 and G_2, Q must contain at least one vertex $v \in V(C)$ and at least one vertex $u \in V(M)$ with $\{v, u\} \notin E(G')$ and a path between v and u whose all internal vertices belong only to G_1 (and thus are outside $V(M) \cup V(C)$). Furthermore, as Q contains a vertex from $H_2 \setminus V(M)$, there must exists vertices $v' \in V(C)$ (possibly $v' = v$) and at least one vertex $u \in V(M)$ (possibly $u' = u$) with $\{v', u'\} \notin E(G')$ and a path between v' and u' whose all internal vertices belong only to G_2. From the above we can conclude that the subpath of Q between v and u that contains a belongs to $\mathcal{P}_1(v, u)$, while the subpath of Q between v' and u' that contains b belongs to $\mathcal{P}_2(v', u')$. This concludes the proof. □

In light of Lemma 4.6, to intersect bad cycles, we now examine how the fractional solution $\mathbf{x}^*$ handles pairs (v, u) of vertices $v \in V(C)$ and $u \in V(M)$.

Lemma 4.7. *Consider (not necessarily distinct) pairs $(v, u), (v', u')$ of vertices,*[4] *where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, and $\{v, v'\} \cap N_{G'}(u') = \emptyset$. If for some $i \in \{1, 2\}$, there is $P \in \mathcal{P}_i(v, u)$, with $\mathbf{x}^*(V(P)) < 1/2$, then for $j \in \{1, 2\} \setminus \{i\}$, we have $\mathbf{x}^*(V(P')) \geq 1/2$, for every $P' \in \mathcal{P}_j(v', u')$.*

Proof. Towards a contradiction, suppose that the lemma is incorrect. Consider pairs (v, u) and (v', u') of vertices, where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, and $\{v, v'\} \cap N_{G'}(u') = \emptyset$, for which the condition of the lemma is not satisfied. Suppose that there is a path $P_1 \in \mathcal{P}_1(v, u)$ such that $\mathbf{x}^*(V(P_1)) < 1/2$, and a path $P_2 \in \mathcal{P}_2(v', u')$ such that $\mathbf{x}^*(V(P_2)) < 1/2$. (All other cases can be handled symmetrically.) Since $\mathbf{x}^*$ is a valid fractional solution, we deduce that $G'[V(P_1) \cup V(P_2)]$ does not contain any chordless cycle. By the construction of $\mathcal{P}_1(v, u)$, P_1 contain a vertex $a \in V(G_1) \setminus (V(M) \cup V(C)) = V(H_1) \setminus V(M)$. Similarly, by the construction of $\mathcal{P}_2(v', u')$, P_2 contain a vertex $a' \in V(G_2) \setminus (V(M) \cup V(C)) = V(H_2) \setminus V(M)$. Note that the vertex sets of $H_1 - V(M)$ and $H_2 - V(M)$ are disjoint. Also, no vertex in $H_1 - V(M)$ is adjacent (in G') to a vertex in $H_2 - V(M)$. We will now show that $G'[V(P_1) \cup V(P_2)]$ contains a chordless cycle, contradicting that $\mathbf{x}^*$ is a valid fractional solution. If $N_{G'}(\{u, v\}) \cap (V(P_2) \setminus \{v', u'\}) = \emptyset$ and $N_{G'}(\{u', v'\}) \cap (V(P_1) \setminus \{v, u\}) = \emptyset$, then $G'[V(P_1) \cup V(P_2)]$ contains a chordless cycle. Now consider the case when at least one of $N_{G'}(\{u, v\}) \cap (V(P_2) \setminus \{v', u'\}) \neq \emptyset$ or $N_{G'}(\{u', v'\}) \cap (V(P_1) \setminus \{v, u\}) \neq \emptyset$ holds, say $N_{G'}(\{u, v\}) \cap (V(P_2) \setminus \{v', u'\}) \neq \emptyset$. (The other case is symmetric.) If there is a vertex $b \in V(P_2) \setminus \{v', u'\}$, such that $\{b, v\}, \{b, u\} \in E(G')$, then notice that $G'[V(P_1) \cup \{b\}]$ contains a chordless cycles. Thus we assume that for every $b \in V(P_2) \setminus \{v', u'\}$ at least one of $\{b, v\}$ or $\{b, u\}$

[4]Possibly, $v = v'$ or $u = u'$.

is not an edge in G'. Let b_v be the vertex that is closest to u' in P_2, such that $\{v, b_v\} \in E(G')$. Note that b_v exists, as $v, v' \in V(C)$ and C is a clique in G'. Moreover, $b_v \neq u'$ as $v \notin N_{G'}(u')$. Let b_u be the vertex closest to b_v in the subpath of P_2 between b_v and u', such that $\{u, b_u\} \in E(G')$. Again, b_u exists, as $u, u' \in V(M)$ and M is a clique in G', and $b_u \neq v'$, as $v' \notin N_{G'}(u)$. By our assumption, we have $N_{G'}(\{u, v\}) \cap (V(P_2) \setminus \{v', u'\}) \neq \emptyset$, thus at least one of b_v, b_u belongs to $V(G_2) \setminus (V(M) \cup V(C))$. Let P'_2 be the subpath in P_2 between the vertices b_v and b_u. As no vertex in $H_1 \setminus V(M)$ is adjacent to a vertex $H_2 \setminus V(M)$, $a \in V(P_1) \cap (V(H_1) \setminus V(M))$, and $a' \in V(P_2) \cap (V(H_2) \setminus V(M))$, we can conclude that $G'[V(P_1) \cup V(P'_2)]$ contains a chordless cycle. This concludes the proof. □

Given $i \in \{1, 2\}$, let $2\mathbf{x}^*_i$ denote the fractional solution that assigns to each vertex the value assigned by $\mathbf{x}^*_i$ times 2. Note that we have $\widehat{H}_1 = G_1 \setminus (V(C) \cup V(M))$ and $\widehat{H}_2 = G_2 \setminus (V(C) \cup V(M))$, where $\widehat{H}_1$ and $\widehat{H}_2$ are chordal graphs. For every (not necessarily distinct) pairs $(v, u), (v', u')$ of vertices (possibly, $v = v'$ or $u = u'$), where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, and $\{v, v'\} \cap N_{G'}(u') = \emptyset$, we perform the following operation. We initialize $\mathcal{T}_1(v, u, v', u') = \mathcal{T}_2(v, u, v', u') = \emptyset$. Insert each pair in $\{(a, b) : a \in N_{G_1}(v) \cap V(\widehat{H}_1), b \in N_{G_1}(u) \cap V(\widehat{H}_1), \widehat{H}_1$ has a path between a and $b\}$ into $\mathcal{T}_1(v, u, v', u')$. Next, we insert each pair in $\{(a', b') : a' \in N_{G_2}(v') \cap V(\widehat{H}_2), b' \in N_{G_2}(u') \cap V(\widehat{H}_2)$, $\widehat{H}_2$ has a path between a' and $b'\}$ into $\mathcal{T}_2(v, u, v', u')$. We remark that the vertices in a pair in $\mathcal{T}_1(v, u, v', u')$ are not necessarily distinct. The following observation follows from Lemma 4.7.

Observation 11. *Consider (not necessarily distinct) pairs $(v, u), (v', u')$ of vertices (possibly, $v = v'$ or $u = u'$), where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, and $\{v, v'\} \cap N_{G'}(u') = \emptyset$. There exists an index $i = i(v, u, v', u') \in \{1, 2\}$, such that for every $(a, b) \in \mathcal{T}_i(v, u, v', u')$ and for every path P between a and b in $\widehat{H}_i$, we have $2\mathbf{x}^*_i(V(P)) \geq 1$.*

Proof. Note that for $(a, b) \in \mathcal{T}_1(v, u, v', u')$ and a path P'_1 between a and b in $\widehat{H}_1$, the path P_1, where $V(P_1) = V(P'_1) \cup \{v, u\}$ and $E(P_1) = E(P'_1) \cup \{v, a\}, \{u, b\}$, belongs to the set $\mathcal{P}_1(v, u)$. Similarly, for $(a', b') \in \mathcal{T}_2(v, u, v', u')$ and a path P'_2 between a' and b' in $\widehat{H}_2$, the path P_2, where $V(P_2) = V(P'_2) \cup \{v', u'\}$ and $E(P_2) = E(P'_2) \cup \{v', a'\}, \{u', b'\}$, belongs to the set $\mathcal{P}_2(v', u')$. Thus using Lemma 4.7 and the fact that $\mathbf{x}^*(V(C) \cup V(M)) = 0$, we can obtain that there is $i = i(v, u, v', u') \in \{1, 2\}$, such that for every $(a, b) \in \mathcal{T}_i(v, u, v', u')$ and for every path P between a and b in $\widehat{H}_i$, we have $2\mathbf{x}^*_i(V(P)) \geq 1$. □

The following lemma translates Observation 11 into an algorithm.

Lemma 4.8. *For every (not necessarily distinct) pairs $(v, u), (v', u')$ of vertices (possibly, $v = v'$ or $u = u'$), where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, and $\{v, v'\} \cap N_{G'}(u') = \emptyset$, one can output (in polynomial time) an index $i = i(v, u, v', u') \in \{1, 2\}$ such that for every $(a, b) \in \mathcal{T}_i(v, u)$ and every path P in $\widehat{H}_i$ between a and b, we have $2\mathbf{x}^*_i(V(P)) \geq 1$.*

Proof. Consider pairs $(v, u), (v', u')$, of vertices (possibly, $v = v'$ or $u = u'$), where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, and $\{v, v'\} \cap N_{G'}(u') = \emptyset$. If there is $i \in \{1, 2\}$ such that $\mathcal{T}_i(v, u, v', u') = \emptyset$, then we have trivially obtained the required index which is $i(v, u, v', u') = i$. Otherwise, we proceed as follows. For any index $i \in \{1, 2\}$, we perform the following procedure. For each pair $(a, b) \in \mathcal{T}_i(v, u)$, we use Dijkstra's algorithm to compute the minimum weight of a path between a and b in the graph $\widehat{H}_i$ where the weights are given by $2\mathbf{x}^*_i$. In case for every pair (a, b) the minimum weight is at least 1, we have found the desired index $i(v, u, v', u')$. Moreover, by Observation 11, for at least one index $i \in \{1, 2\}$, the minimum over the minimum weights associated with the pairs (a, b) should be at least 1 (if this value is at least 1 for both indices, we arbitrarily decide to fix $i(v, u, v', u') = 1$). □

At this point, we need to rely on approximate solutions to Weighted Multicut in chordal graphs (in this context, we will employ the algorithm given by Theorem 1.3 in Section 5). Here, a fractional solution $\mathbf{y}$ is a function $\mathbf{y} : V(G') \longrightarrow [0, \infty)$ such that for every pair $(s_i, t_i) \in \mathcal{T}$ and any path P between s_i and t_i, it holds that $\mathbf{y}(V(P)) \geq 1$.[5] An optimal fractional solution minimizes the weight $w(\mathbf{y}) = \sum_{v \in V(G')} w(v) \cdot \mathbf{y}(v)$. Let fopt denote the weight of an optimal fractional solution.

By first employing the algorithm given by Lemma 4.8, we next construct two instances of Weighted Multicut. The first instance is $J_1 = (\widehat{H}_1, w_1, \mathcal{T}_1)$ and the second instance is $J_2 = (\widehat{H}_2, w_2, \mathcal{T}_2)$, where the sets $\mathcal{T}_1$ and $\mathcal{T}_2$ are defined as follows. We initialize $\mathcal{T}_1 = \emptyset$. For every (not necessarily distinct) pairs $(v, u), (v', u')$ of vertices (possibly, $v = v'$ or $u = u'$), where $v, v' \in V(C)$, $u, u' \in V(M)$, $\{v, v'\} \cap N_{G'}(u) = \emptyset$, $\{v, v'\} \cap N_{G'}(u') = \emptyset$, and $i(v, u, v', u') = 1$,[6] we insert each pair in $\mathcal{T}_1(v, u, v', u')$ into $\mathcal{T}_1$. The definition of $\mathcal{T}_2$ is symmetric to the one of $\mathcal{T}_1$.

By Observation 11 and since for all $v \in V(C) \cup V(M)$ it holds that $\mathbf{x}_1^*(v) = \mathbf{x}_2^*(v) = 0$, we deduce that $2\mathbf{x}_1^*$ and $2\mathbf{x}_2^*$ are valid solutions to J_1 and J_2, respectively. Thus, by calling the algorithm given by Lemma 5.1 with each instance, we obtain a solution r_1 to the first instance, along with a set R_1 that realizes it, such that $r_1 \leq 2d \cdot w(\mathbf{x}_1^*)$, and we also obtain a solution r_2 to the second instance, along with a set R_2 that realizes it, such that $r_2 \leq 2d \cdot w(\mathbf{x}_2^*)$, for some fixed constant d.[7]

By Observation 8 and Lemma 4.6, we obtained a set $S^* = S_1 \cup S_2 \cup R_1 \cup R_2$ for which we have the following observation.

Observation 12. *S^* intersects any chordless cycle in G', and it holds that* $w(S^*) \leq s_1 + s_2 + r_1 + r_2$.

Recall that to prove Lemma 4.2 we need to show that $s_1 + s_2 + r_1 + r_2 \leq (\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta - 1} \cdot c \cdot \log(\alpha(G')) \cdot w'(\mathbf{x})$ and we have $\delta \geq 2$. Furthermore, we have $(\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta} \cdot c \cdot \log(\alpha(G')) \cdot w'(\mathbf{x}) \leq (\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta - 1} \cdot c \cdot \log(\alpha(G')) \cdot w'(\mathbf{x})$. This together with Lemma 4.4 implies that it is enough to show $s_1 + s_2 + r_1 + r_2 \leq (\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta} \cdot c \cdot \log(\alpha(G')) \cdot w(\mathbf{x}^*)$. Recall that for any $i \in \{1, 2\}$, $r_i \leq 2d \cdot w(\mathbf{x}_i^*)$. Thus, by Observation 9 and since for any $i \in \{1, 2\}$, $\alpha(G_i) \leq \frac{3}{4}\alpha(G')$, we have that

$$w(S^*) \leq (\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta} \cdot c \cdot \log(\frac{3}{4}\alpha(G')) \cdot (w(\mathbf{x}_1^*) + w(\mathbf{x}_2^*)) + 2d \cdot (w(\mathbf{x}_1^*) + w(\mathbf{x}_2^*)).$$

By Observation 10, we further deduce that

$$w^*(S^*) \leq \left((\frac{c \cdot \log n}{c \cdot \log n + 9})^{\delta} \cdot c \cdot \log(\frac{3}{4}\alpha(G')) + 2d \right) \cdot w(\mathbf{x}^*).$$

Now, it only remains to show that $(\frac{c \log n}{c \log n + 9})^{\delta} \cdot c \cdot \log(\frac{3}{4}\alpha(G')) + 2d \leq (\frac{c \log n}{c \log n + 9})^{\delta} \cdot c \cdot \log \alpha(G')$, which is equivalent to $2d \leq (\frac{c \log n}{c \log n + 9})^{\delta} \cdot c \cdot \log(\frac{4}{3})$. Recall that $\delta \leq q \cdot \log n$ (Observation 6). Thus, it is sufficient that we show that $2d \leq (\frac{c \log n}{c \log n + 9})^{q \cdot \log n} \cdot c \cdot \log(\frac{4}{3})$. However, the term $(\frac{c \log n}{c \log n + 9})^{q \cdot \log n}$ is lower bounded by $1/e^{9q}$. In other words, it is sufficient that we fix $c \geq 2 \cdot e^{9q} \cdot d \cdot 1/\log(\frac{4}{3})$.

4.2 General Graphs

In this subsection we handle general instances by developing a $d \cdot \log^2 n$-factor approximation algorithm for WCVD, Gen-CVD-APPROX, thus proving the correctness of Theorem 1.2. The exact value of the constant $d \geq \max\{96, 2c\}$ is determined later.[8] This algorithm is based on recursion,

[5]Note that if $s_i = t_i$, then $\mathbf{y}(s_i) \geq 1$.

[6]$i(v, u, v', u')$ is the index returned by Lemma 4.8.

[7]Here, d is the approximation factor for Weighted Multicut. For $i \in \{1, 2\}$, we have $r_i \leq 2d \cdot w(\mathbf{x}_i^*)$, as $2\mathbf{x}_i^*$ is a bound on the solution size for the Weighted Multicut instance J_i.

[8]Recall that c is the constant we fixed to ensure that the approximation ratio of CVD-APPROX is bounded by $c \cdot \log n$.

and during its execution, we often encounter instances that are of the form of the Clique+Chordal special case of WCVD, which will be dealt with using the algorithm CVD-APPROX of Section 4.1.

Recursion. We define each call to our algorithm Gen-CVD-APPROX to be of the form (G', w'), where (G', w') is an instance of WCVD such that G' is an induced subgraph of G, and we denote $n' = |V(G')|$. We ensure that after the initialization phase, the graph G' never contains chordless cycles on at most 48 vertices. We call this invariant the *C_{48}-free invariant.* In particular, this guarantee ensures that the graph G' always contains only a small number of maximal cliques:

Lemma 4.9 ([11, 48]). *The number of maximal cliques of a graph G' that has no chordless cycles on four vertices is bounded by $O(n'^2)$, and they can be enumerated in polynomial time using a polynomial delay algorithm.*

Goal. For each recursive call Gen-CVD-APPROX(G', w'), we aim to prove the following.

Lemma 4.10. Gen-CVD-APPROX *returns a solution that is at least* opt *and at most* $\frac{d}{2} \cdot \log^2 n' \cdot \text{opt}$. *Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At each recursive call, the size of the graph G' becomes smaller. Thus, when we prove that Lemma 4.10 is true for the current call, we assume that the approximation factor is bounded by $\frac{d}{2} \cdot \log^2 \widehat{n} \cdot \text{opt}$ for any call where the size $\widehat{n}$ of the vertex-set of its graph is strictly smaller than n'.

Initialization. Initially, we set $(G', w') = (G, w)$. However, we need to ensure that the C_{48}-free invariant is satisfied. For this purpose, we update G' as follows. First, we let C_{48} denote the set of all chordless cycles on at most 48 vertices of G'. Clearly, C_{48} can be computed in polynomial time and it holds that $|C_{48}| \leq n^{48}$. Now, we construct an instance of Weighted 48-Hitting Set, where the universe is $V(G')$, the family of 48-sets is C_{48}, and the weight function is w'. Since each chordless cycle must be intersected, it is clear that the optimal solution to our Weighted 48-Hitting Set instance is at most opt. By using the standard c'-approximation algorithm for Weighted c'-Hitting Set [33], which is suitable for any fixed constant c', we obtain a set $S \subseteq V(G')$ that intersects all cycles in C_{48} and whose weight is at most $48 \cdot \text{opt}$. Having the set S, we remove its vertices from G'. Now, the C_{48}-free invariant is satisfied, which implies that we can recursively call our algorithm. To the outputted solution, we add $w(S)$ and S. If Lemma 4.10 is true, we obtain a solution that is at most $\frac{d}{2} \cdot \log^2 n \cdot \text{opt} + 48 \cdot \text{opt} \leq d \cdot \log^2 n \cdot \text{opt}$, which allows us to conclude the correctness of Theorem 1.2. We remark that during the execution of our algorithm, we only update G' by removing vertices from it, and thus it will always be safe to assume that the C_{48}-free invariant is satisfied.

Termination. Observe that due to Lemma 4.9, we can test in polynomial time whether G' consists of a clique and a chordal graph: we examine each maximal clique of G', and check whether after its removal we obtain a chordal graph. Once G' becomes such a graph that consists of a chordal graph and a clique, we solve the instance (G', w') by calling algorithm CVD-APPROX. Since $c \cdot \log n' \leq \frac{d}{2} \cdot \log^2 n'$, we thus ensure that at the base case of our induction, Lemma 4.10 holds.

Recursive Call. For the analysis of a recursive call, let S^* denote a hypothetical set that realizes the optimal solution opt of the current instance (G', w'). Moreover, let (F, β) be a clique forest of $G' - S^*$, whose existence is guaranteed by Lemma 2.1. Using standard arguments on forests, we have the following observation.

Observation 13. *There exist a maximal clique M of G' and a subset $S \subseteq V(G') \setminus M$ of weight at most* opt *such that $M \cup S$ is a balanced separator for G'.*

The following lemma translates this observation into an algorithm.

Lemma 4.11. *There is a polynomial-time algorithm that finds a maximal clique M of G' and a subset $S \subseteq V(G') \setminus M$ of weight at most $q \cdot \log n' \cdot \text{opt}$ for some fixed constant q such that $M \cup S$ is a balanced separator for G'.*

Proof. We examine every maximal clique of G'. By Lemma 4.9, we need only consider $O(n'^2)$ maximal cliques, and these cliques can be enumerated in polynomial time. For each such clique M, we run the $q \cdot \log n'$-factor approximation algorithm by Leighton and Rao [34] to find a balanced separator S_M of $G' - M$. Here, q is some fixed constant. We let S denote some set of minimum weight among the sets in $\{S_M : M \text{ is a maximal clique of } G'\}$. By Observation 13, $w(S) \leq q \cdot \log n' \cdot \text{opt}$. Thus, the desired output is a pair (M, S) where M is one of the examined maximal cliques such that $S_M = S$. □

We call the algorithm in Lemma 4.11 to obtain a pair (M, S). Since $M \cup S$ is a balanced separator for G', we can partition the set of connected components of $G' - (M \cup S)$ into two sets, $\mathcal{A}_1$ and $\mathcal{A}_2$, such that for $V_1 = \bigcup_{A \in \mathcal{A}_1} V(A)$ and $V_2 = \bigcup_{A \in \mathcal{A}_2} V(A)$ it holds that $n_1, n_2 \leq \frac{2}{3}n'$ where $n_1 = |V_1|$ and $n_2 = |V_2|$. We remark that we used the $O(\log n)$-factor approximation algorithm by Leighton and Rao [34] in Lemma 4.11 to find the balanced separator instead of the $O(\sqrt{\log n})$-factor approximation algorithm by Feige et al. [12], as the algorithm by Feige et al. is randomized.

Next, we define two inputs of (the general case of) WCVD: $I_1 = (G'[V_1], w'|_{V_1})$ and $I_2 = (G'[V_2], w'|_{V_2})$. Let opt_1 and opt_2 denote the optimal solutions to I_1 and I_2, respectively. Observe that since $V_1 \cap V_2 = \emptyset$, it holds that $\text{opt}_1 + \text{opt}_2 \leq \text{opt}$. We solve each of the subinstances by recursively calling algorithm Gen-CVD-APPROX. By the inductive hypothesis, we thus obtain two sets, S_1 and S_2, such that $G'[V_1] - S_1$ and $G'[V_2] - S_2$ are chordal graphs, and $w'(S_1) \leq \frac{d}{2} \cdot \log^2 n_1 \cdot \text{opt}_1$ and $w'(S_2) \leq \frac{d}{2} \cdot \log^2 n_2 \cdot \text{opt}_2$.

We proceed by defining an input of the Clique+Chordal special case of WCVD: $J = (G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)], w'|_{(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)})$. Observe that since $G'[V_1] - S_1$ and $G'[V_2] - S_2$ are chordal graphs and M is a clique, this is indeed an instance of the Clique+Chordal special case of WCVD. We solve this instance by calling algorithm CVD-APPROX. We thus obtain a set, $\widehat{S}$, such that $G'[(V_1 \cup V_2 \cup M) - (S_1 \cup S_2 \cup \widehat{S})]$ is a chordal graphs, and $w'(\widehat{S}) \leq c \cdot \log n' \cdot \text{opt}$ (since $|(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2)| \leq n'$ and the optimal solution of each of the subinstances is at most opt).

Observe that since M is a clique and there is no edge in $E(G')$ between a vertex in V_1 and a vertex in V_2, any chordless cycle of $G' - (S \cup S_1 \cup S_2)$ entirely belongs to either $G'[(V_1 \cup M) \setminus S_1]$ or $G'[(V_2 \cup M) \setminus S_2]$. This observation, along with the fact that $G'[(V_1 \cup V_2 \cup M) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a chordal graphs, implies that $G' - T$ is a chordal graphs where $T = S \cup S_1 \cup S_2 \cup \widehat{S}$. Thus, it is now sufficient to show that $w'(T) \leq \frac{d}{2} \cdot \log^2 n' \cdot \text{opt}$.

By the discussion above, we have that

$$\begin{aligned} w'(T) &\leq w'(S) + w'(S_1) + w'(S_2) + w'(\widehat{S}) \\ &\leq q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot (\log^2 n_1 \cdot \text{opt}_1 + \log^2 n_2 \cdot \text{opt}_2) + c \cdot \log n' \cdot \text{opt}. \end{aligned}$$

Recall that $n_1, n_2 \leq \frac{2}{3}n'$ and $\text{opt}_1 + \text{opt}_2 \leq \text{opt}$. Thus, we have that

$$\begin{aligned} w'(T) &\leq q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot (\log^2 \tfrac{2}{3}n') \cdot \text{opt} + c \cdot \log n' \cdot \text{opt} \\ &\leq \tfrac{d}{2} \cdot \log^2 n' \cdot \text{opt} + (q + c - \tfrac{d}{2} \log \tfrac{3}{2}) \cdot \log n' \cdot \text{opt}. \end{aligned}$$

Overall, we conclude that to ensure that $w'(T) \leq \frac{d}{2} \cdot \log^2 n' \cdot \text{opt}$, it is sufficient to ensure that $q + c - \frac{d}{2} \log \frac{3}{2} \leq 0$, which can be done by fixing $d = \frac{2}{\log \frac{3}{2}} \cdot (q + c)$.

5 WEIGHTED MULTICUT IN CHORDAL GRAPHS

In this section we prove Theorem 1.3. This algorithm is inspired by the algorithm of Garg.et.al [19] for Multicut in trees. The key idea is to define distances for vertices in $V(G)$ from a fixed node $r_G \in V(G)$, using a feasible fractional LP solution $\mathbf{x}$. These distances are used to construct a collection of vertex subsets such that each of them is a feasible multicut in G, and we output the one with minimum weight. We argue that the minimum weight subset yields a constant factor approximation. The feasibility of these vertex subsets is a consequence of the distances; if a subset is not feasible then the distance between some pair $(s, t) \in \mathcal{T}$ is too small, and hence $\mathbf{x}$ is not a feasible solution to the LP. We now present our arguments formally.

Let us denote $c = 8$. Recall that for Weighted Multicut, a fractional solution $\mathbf{x}$ is a function $\mathbf{x} : V(G) \rightarrow [0, \infty)$ such that for every pair $(s, t) \in \mathcal{T}$ and any path P between s and t, it holds that $\mathbf{x}(V(P)) \geq 1$. An optimal fractional solution minimizes the weight $w(\mathbf{x}) = \sum_{v \in V(G)} w(v) \cdot \mathbf{x}(v)$. Let fopt denote the weight of an optimal fractional solution. Theorem 1.3 follows from the next result, whose proof is the focus of this section.

Lemma 5.1. *Given an instance of* Weighted Multicut *in chordal graphs, one can find (in polynomial time) a solution that is at least* opt *and at most* $4c \cdot$ fopt, *along with a set that realizes it.*

Preprocessing. By using the ellipsoid method, we may next assume that we have optimal fractional solution $\mathbf{x}$ at hand. We say that $\mathbf{x}$ is *nice* if for all $v \in V(G)$, there exists $i \in \{0\} \cup \mathbb{N}$ such that $\mathbf{x}(v) = \frac{i}{n}$. Let $h(\mathbf{x}) = \{v \in V(G) : \mathbf{x}(v) \geq 1/c\}$ denote the set of vertices to which $\mathbf{x}$ assigns high values.

Lemma 5.2. *Define a function* $\widehat{\mathbf{x}} : V(G) \rightarrow [0, \infty)$ *as follows. For all* $v \in V(G)$, *if* $\mathbf{x}(v) < 1/2n$ *then* $\widehat{\mathbf{x}}(v) = 0$, *and otherwise* $\widehat{\mathbf{x}}(v)$ *is the smallest value of the form* i/n, *for some* $i \in \mathbb{N}$, *that is at least* $2\mathbf{x}(v)$. *Then,* $\widehat{\mathbf{x}}$ *is a nice fractional solution such that* $w(\widehat{\mathbf{x}}) \leq 4w(\mathbf{x})$.

Proof. To show that $\widehat{\mathbf{x}}$ is a fractional solution, consider some path P between s and t such that $(s, t) \in \mathcal{T}$. Let $\ell'(\mathbf{x}) = \{v \in V(G) : \mathbf{x}(v) < 1/2n\}$. We have that $\widehat{\mathbf{x}}(V(P)) = \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \widehat{\mathbf{x}}(v) \geq 2 \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$. Thus, to show that $\widehat{\mathbf{x}}(V(P)) \geq 1$, it is enough to show $\frac{1}{2} \leq \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$. Since $\mathbf{x}$ is a fractional solution, it holds that $\mathbf{x}(V(P)) = \sum_{v \in V(P) \cap \ell'(\mathbf{x})} \mathbf{x}(v) + \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v) \geq 1$. Thus, $1 \leq \frac{1}{2n}|V(P) \cap \ell'(\mathbf{x})| + \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$. Since $|V(P) \cap \ell'(\mathbf{x})| \leq n$, we conclude that $\frac{1}{2} \leq \sum_{v \in V(P) \setminus \ell'(\mathbf{x})} \mathbf{x}(v)$.

The second part of the claim follows from the observation that for all $v \in V(G)$, $\widehat{\mathbf{x}}(v) \leq 4\mathbf{x}(v)$. □

Accordingly, we update $\mathbf{x}$ to $\widehat{\mathbf{x}}$. Our preprocessing step also relies on the following standard lemma.

Lemma 5.3. *Define* $\widehat{G} = G - h(\mathbf{x})$, $\widehat{w} = w|_{V(\widehat{G})}$ *and* $\widehat{\mathbf{x}} = \mathbf{x}|_{V(\widehat{G})}$. *Then,* $c \cdot w(\widehat{\mathbf{x}}) + w(h(\mathbf{x})) \leq c \cdot w(\mathbf{x})$.

Proof. By the definition of $h(\mathbf{x})$, it holds that $w(\widehat{\mathbf{x}}) \leq w(\mathbf{x}) - \frac{1}{c} w(h(\mathbf{x}))$. Thus, $c \cdot w(\widehat{\mathbf{x}}) + w(h(\mathbf{x})) \leq c \cdot (w(\mathbf{x}) - \frac{1}{c} \cdot w(h(\mathbf{x}))) + w(h(\mathbf{x})) = c \cdot w(\mathbf{x})$. □

We thus further update G to $\widehat{G}$, w to $\widehat{w}$ and $\mathbf{x}$ to $\widehat{\mathbf{x}}$, where we ensure that once we obtain a solution to the new instance, we add $w(h(\mathbf{x}))$ to this solution and $h(\mathbf{x})$ to the set realizing it. Overall, we may next focus only on the proof of the following lemma.

Lemma 5.4. *Let* (G, w) *be an instance of* Weighted Multicut *in chordal graphs, and* $\mathbf{x}$ *be a nice fractional solution such that* $h(\mathbf{x}) = \emptyset$. *Then, one can find (in polynomial time) a solution that is at least* opt *and at most* $c \cdot w(\mathbf{x})$, *along with a set that realizes it.*

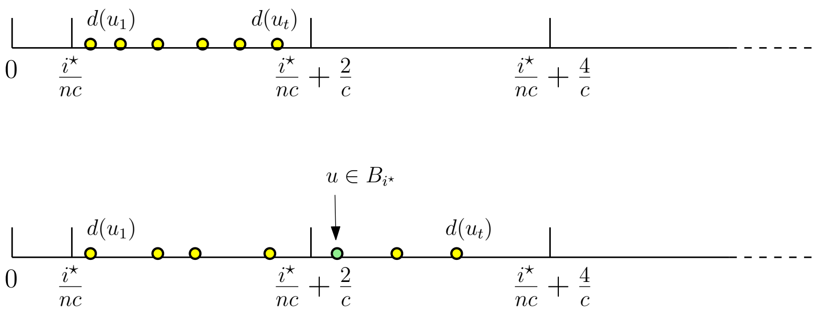
$d(u_1)$
$d(u_t)$
0
$\frac{i^\star}{nc}$
$\frac{i^\star}{nc} + \frac{2}{c}$
$\frac{i^\star}{nc} + \frac{4}{c}$
$u \in B_{i^\star}$

Fig. 3. Illustration of the proof of feasibility. The membership of a vertex v into $B_{i^\star}$ is determined by a partition of the real line into intervals of length $\frac{2}{c}$, and the value of $d(v)$. If some path P from $u_1 = r_G$ to $u_t = v$ has no vertices in $B_{i^\star}$ then the $x(P) < \frac{4}{c}$; otherwise $V(P) \cap B_{i^\star} \neq \emptyset$

The Algorithm. Since G is a chordal graph, we can first construct in polynomial time a clique forest (F, β) of G (Lemma 2.1). Without loss of generality, we may assume that F is a tree, else G is not a connected graph and we can handle each of its connected components separately. Now, we arbitrarily root F at some node r_F, and we arbitrarily choose a vertex $r_G \in \beta(r_F)$. We then use Dijkstra's algorithm to compute (in polynomial time) for each vertex $v \in V(G)$, the value $d(v) = \min_{P \in \mathcal{P}(v)} \mathbf{x}(V(P))$, where $\mathcal{P}(v)$ is the set of paths in G between r_G and v.

We define $n+1$ bins: for all $i \in \{0, 1, \ldots, n\}$, the bin B_i contains every vertex $v \in V(G)$ for which there exists $j \in \{0\} \cup \mathbb{N}$ such that $d(v) - \mathbf{x}(v) < (\frac{i}{n} + 2j)\frac{1}{c} \leq d(v)$ (i.e., $0 \leq d(v) - (\frac{i}{n} + 2j)\frac{1}{c} < \mathbf{x}(v)$). Let B_{i^*}, $i^* \in \{0, 1, \ldots, n\}$, be a bin that minimizes $w(B_{i^*})$. The output consists of $w(B_{i^*})$ and B_{i^*}.

Approximation Factor. Given $r \in [0, 1]$, let $\widehat{B}_r$ be the set that contains every vertex $v \in V(G)$ for which there exists $j \in \{0\} \cup \mathbb{N}$ such that $0 \leq d(v) - (r + 2j)\frac{1}{c} < \mathbf{x}(v)$. We start with the following claim.

Lemma 5.5. *There exists $r^* \in [0, 1]$ such that $w(\widehat{B}_{r^*}) \leq c \cdot w(\mathbf{x})$.*

Proof. For any $d \geq 0$, observe that there exists exactly one $j \in \{0\} \cup \mathbb{N}$ for which there exists $r \in [0, 1]$ such that $0 \leq d - (r + 2j)\frac{1}{c} < \frac{1}{c}$, and denote it by $j(d)$. Suppose that we choose $r \in [0, 1]$ uniformly at random. Consider some vertex $v \in V(G)$. Then, since $h(\mathbf{x}) = \emptyset$, the probability that there exists $j \in \{0\} \cup \mathbb{N}$ such that $0 \leq d(v) - (r + 2j)\frac{1}{c} < \mathbf{x}(v)$ is equal to the probability that $0 \leq d(v) - (r + 2j(d(v)))\frac{1}{c} < \mathbf{x}(v)$. Now, the probability that $0 \leq d(v) - (r + 2j(d(v)))\frac{1}{c} < \mathbf{x}(v)$ is at most $c \cdot \mathbf{x}(v)$. The expected weight $w(\widehat{B}_r)$ is $c \cdot \sum_{v \in V(G)} \mathbf{x}(v) \cdot w(v) \leq c \cdot w(\mathbf{x})$. Thus, there exists $r^* \in [0, 1]$ such that $w(\widehat{B}_{r^*}) \leq c \cdot w(\mathbf{x})$. □

Now, the proof of the approximation factor follows from the next claim.

Lemma 5.6. *There exists $i \in \{0, 1, \ldots, n\}$ such that $B_i \subseteq \widehat{B}_{r^*}$.*

Proof. Let i be the smallest index in $\{0, 1, \ldots, n\}$ such that $r^* \leq \frac{i}{n}$. Consider some vertex $v \in B_i$. Then, for some $j \in \{0\} \cup \mathbb{N}$, $d(v) - \mathbf{x}(v) < (\frac{i}{n} + 2j)\frac{1}{c} \leq d(v)$. Since $r^* \leq \frac{i}{n}$, we have that $(r^* + 2j)\frac{1}{c} \leq d(v)$. Since $\mathbf{x}$ is nice, it holds that there exists $t \in \{0\} \cup \mathbb{N}$ such that $d(v) - \mathbf{x}(v) = \frac{t}{n}$. Thus, for any $p < \frac{1}{n}$, it holds that $d(v) - \mathbf{x}(v) < (\frac{i}{n} + 2j - p)\frac{1}{c}$. By the choice of i, $\frac{i}{n} - r^* < \frac{1}{n}$, and therefore $d(v) - \mathbf{x}(v) < (r^* + 2j)\frac{1}{c}$, which implies that $v \in \widehat{B}_{r^*}$. □

Feasibility. We need to prove that for any pair $(s^*, t^*) \in \mathcal{T}$, $G - B_{i^*}$ does not have any path between s^* and t^*. Consider some path $P = (v_1, v_2, \cdots, v_\ell)$ between s^* and t^*. Here, $v_1 = s^*$ and $v_\ell = t^*$. Suppose, by way of contradiction, that $V(P) \cap B_{i^*} = \emptyset$. Then, for all $v_i \in V(P)$, it holds that there is no $j \in \{0\} \cup \mathbb{N}$ such that $0 \leq d(v_i) - (\frac{i^*}{n} + 2j)\frac{1}{c} < \mathbf{x}(v_i)$.

Let $s \in V(F)$ be the closest node to r_F that satisfies $\beta(s) \cap V(P) \neq \emptyset$ (since F is a clique tree and P is a path, the node s is uniquely defined). Let $v_{\widehat{i}}$ be some vertex in $\beta(s) \cap V(P) \neq \emptyset$. For the sake of clarity, let us denote the subpath of P between $v_{\widehat{i}}$ and v_ℓ by $Q = (u_1, u_2, \cdots, u_t)$, where $u_1 = v_{\widehat{i}}$ and $u_t = v_\ell$. Let j^* be the smallest value in $\{0\} \cup \mathbb{N}$ that satisfies $d(u_1) - \mathbf{x}(u_1) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. Note that $d(u_1) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. It is thus well defined to let p denote the largest index in $[t]$ such that $d(u_p) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$.

First, suppose that $p \in [t-1]$. We then have that $(\frac{i^*}{n} + 2j^*)\frac{1}{c} \leq d(u_{p+1})$. For all $2 \leq i \leq t$, it holds that $d(u_i) \leq d(u_{i-1}) + \mathbf{x}(u_i)$. We thus obtain that $d(u_{p+1}) - \mathbf{x}(u_{p+1}) \leq d(u_p) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. This statement implies that $u_{p+1} \in B_{i^*}$, which is a contradiction.

Now, we suppose that $p = t$. Note that $(\frac{i^*}{n} + 2j^* - 2)\frac{1}{c} \leq d(u_1) - \mathbf{x}(u_1)$ (by the minimality of j^*), and $d(u_t) < (\frac{i^*}{n} + 2j^*)\frac{1}{c}$. We get that $d(u_t) < d(u_1) - \mathbf{x}(u_1) + \frac{2}{c}$. In other words, $d(u_t) - d(u_1) + \mathbf{x}(u_1) < \frac{2}{c}$. Let des$(s)$ denote the set consisting of s and its descendants in F. Since F is a clique tree, we have that $V(Q) \subseteq \bigcup_{s' \in \text{des}(s)} \beta(s')$. Thus, any path from r_G to u_t that realizes $d(u_t)$ contains a vertex from $\beta(s)$. Since there exists a path from r_G to u_t that realizes $d(u_t)$, we deduce that there exists a path, P_t, from r_G to u_t that realizes $d(u_t)$ and contains a vertex $x \in N_G[u_1]$. Let P_t^* denote the subpath of P_t between x and u_t, and let P^* denote the path that starts at u_1 and then traverses P_t^*. Then, $\mathbf{x}(V(P^*)) \leq \mathbf{x}(u_1) + \mathbf{x}(V(P_t^*)) = \mathbf{x}(u_1) + d(u_t) - d(x) + \mathbf{x}(x)$. Note that $d(u_1) \leq d(x) + \mathbf{x}(u_1)$, and therefore $\mathbf{x}(V(P^*)) \leq \mathbf{x}(u_1) + d(u_t) - (d(u_1) - \mathbf{x}(u_1)) + \mathbf{x}(x) = \mathbf{x}(u_1) + \mathbf{x}(x) + (d(u_t) - d(u_1) + \mathbf{x}(u_1))$. Since $h(\mathbf{x}) = \emptyset$ and $d(u_t) - d(u_1) + \mathbf{x}(u_1) < \frac{2}{c}$, we get that $\mathbf{x}(V(P^*)) < \frac{4}{c}$. The symmetric analysis of the subpath of P between $u_1 = v_{\widehat{i}}$ and v_1 shows that there exists a path P^{**} between u_1 and v_1 such that $\mathbf{x}(V(P^{**})) < \frac{4}{c}$. Overall, we get that there exists a path, P', between $v_1 = s^*$ and $v_\ell = u_\ell = t^*$ such that $\mathbf{x}(V(P')) < \frac{8}{c}$. Since $c \geq 8$, we reach a contradiction to the assumption that $\mathbf{x}$ is a fractional solution.

6 DISTANCE-HEREDITARY VERTEX DELETION

In this section we prove Theorem 1.4. We start with preliminaries.

Preliminaries. A graph G is *distance hereditary* if every connected induced subgraph H of G, for all $u, v \in V(H)$ the number of vertices in shortest path between u and v in G is same as the number of vertices in shortest path between u and v in H. Another characterization of distance hereditary graphs is the graph not containing an induced sub-graph isomorphic to a house, a gem, a domino or an induced cycle on 5 or more vertices (refer Figure 4). We refer to a house, a gem, a domino or an induced cycle on at least 5 vertices as a *DH-obstruction.* A DH-obstruction on at most 50 vertices is a *small DH-obstruction.* A *biclique* is a graph G with vertex bipartition X, Y each of them being non-empty such that for each $x \in X$ and $y \in Y$ we have $\{x, y\} \in E(G)$. We note here that, X and Y need not be independent sets in a *biclique* G.

Clearly, we can assume that the weight $w(v)$ of each vertex $v \in V(G)$ is positive, else we can insert v into any solution. Our approximation algorithm for WDHVD comprises of two components. The first component handles the special case where the input graph G consists of a biclique C and a distance hereditary H. Here, we also assume that the input graph has no "small" DH-obstruction. We show that when input restricted to these special instances WDHVD admits an $O(\log^2 n)$-factor approximation algorithm.

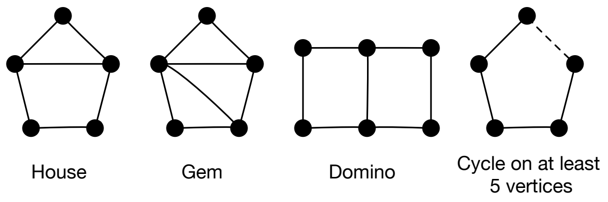

Fig. 4. Obstruction set for distance hereditary graphs

The second component is a recursive algorithm that solves general instances of the problem. Initially, it easily handles "small" DH-obstruction. Then, it gradually disintegrates a general instance until it becomes an instance of the special form that can be solved in polynomial time. More precisely, given a problem instance, the algorithm divides it by finding a maximal biclique M (using an exhaustive search which relies on the guarantee that G has no "small" DH-obstruction) and a small separator S (using an approximation algorithm) that together break the input graph into two graphs significantly smaller than the original graph.

6.1 Biclique+ Distance Hereditary Graph

In this subsection we handle the special case where the input graph G consists of a biclique C and a distance hereditary graph H. More precisely, along with the input graph G and the weight function w, we are also given a biclique C and a distance hereditary graph H such that $V(G) = V(C) \cup V(H)$, where the vertex-sets $V(C)$ and $V(H)$ are disjoint. Here, we also assume that G has no DH-obstruction on at most 50 vertices, which means that every DH-obstruction in G is a chordless cycle of strictly more than 50 vertices. Note that the edge-set $E(G)$ may contain edges between vertices in C and vertices in H. We call this special case the *Biclique + Distance Hereditary special case.* Our objective is to prove the following result.

Lemma 6.1. *The Biclique + Distance Hereditary special case of WDHVD admits an $O(\log^2 n)$-factor approximation algorithm.*

We assume that $n \geq 2^{12}$, else the input instance can be solve by brute-force [1]. Let c be a fixed constant (to be determined later). In the rest of this subsection, we design a $c \cdot \log n$-factor approximation algorithm for the Biclique + Distance Hereditary special case of WDHVD.

Recursion. Our approximation algorithm is a recursive algorithm, which we call DHD-APPROX, and define each call to be of the form $(G', w', C, H', \mathbf{x})$. Here, G' is an induced subgraph of G such that $V(C) \subseteq V(G')$, and H' is an induced subgraph of H. The argument $\mathbf{x}$ is discussed below. We remark that we continue to use n to refer to the size of the vertex-set of the input graph G rather than the current graph G'.

Arguments. While the execution of our algorithm progresses, we keep track of two arguments: the number of vertices in the current distance hereditary graph H' that are assigned a non-zero value by $\mathbf{x}$, which we denote by $\alpha(G')$ and the fractional solution $\mathbf{x}$.

Observation 14. *The measure $\alpha(G')$ can be computed in polynomial time.*

A fractional solution $\mathbf{x}$ is a function $\mathbf{x} : V(G') \rightarrow [0, \infty)$ such that for every chordless cycle Q of G' on at least 5 vertices it holds that $\mathbf{x}(V(Q)) \geq 1$. An optimal fractional solution minimizes the

[1]This assumption simplifies some of the calculations ahead.

weight $w'(\mathbf{x}) = \sum_{v \in V(G')} w'(v) \cdot \mathbf{x}(v)$. Clearly, the solution to the instance (G', w') of WDHVD is at least as large as the weight of an optimal fractional solution. Although we initially compute an optimal fractional solution $\mathbf{x}$ (at the initialization phase that is described below), during the execution of our algorithm, we manipulate this solution so it may no longer be optimal. Prior to any call to DHD-APPROX with the exception of the first call, we ensure that $\mathbf{x}$ satisfies the following invariants:

- **Low-Value Invariant**: For any $v \in V(G')$, it holds that $\mathbf{x}(v) < 1/\log n$.
- **Zero-Biclique Invariant**: For any $v \in V(C)$, it holds that $\mathbf{x}(v) = 0$.

Additionally, we will always maintain that the solutions returned by recursive calls and the base case do not contain any vertex $v \in V(G')$, such that $\mathbf{x}(v) = 0$. We call the above as a **No-Zero Solution Invariant**.

We note that the Low-Value Invariant used here is simpler than the one used in Section 4.1 since it is enough for the purpose of this section.

Goal. The depth of the recursion tree will be bounded by $\Delta = O(\log n)$, where the depth of initial call is 1. The correctness of this claim is proved when we explain how to perform a recursive call. For each recursive call to DHD-APPROX$(G', w', C, H', \mathbf{x})$, we aim to prove the following.

Lemma 6.2. *For any $\delta \in \{1, 2, \ldots, \Delta\}$, each recursive call to* DHD-APPROX *of depth $\delta \geq 2$ returns a solution that is at least* opt *and at most $(\frac{\log n}{\log n+4})^{\delta} \cdot c \cdot \log n \cdot \log(\alpha(G')) \cdot w'(\mathbf{x})$. Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution.*

At the initialization phase, we see that in order to prove Lemma 6.1, it is sufficient to prove Lemma 6.2.

Initialization. Initially, the graphs G' and H' are simply set to be the input graphs G and H, and the weight function w' is simply set to be input weight function w. Moreover, we compute an optimal fractional solution $\mathbf{x} = \mathbf{x}_{\text{init}}$ by using the ellipsoid method. Recall that the following claim holds.

Observation 15. *The solution of the instance (G', w') of WDHVD is lower bounded by $w'(\mathbf{x}_{\text{init}})$.*

Moreover, it holds that $\alpha(G') \leq n$, and therefore to prove Lemma 6.1, it is sufficient to return a solution that is at least opt and at most $c \cdot \log n \cdot \log(\alpha(G)) \cdot w(\mathbf{x})$ (along with a subset that realizes the solution). Part of the necessity of the stronger claim given by Lemma 6.2 will become clear at the end of the initialization phase.

We would like to proceed by calling our algorithm recursively. For this purpose, we first need to ensure that $\mathbf{x}$ satisfies the low-value and zero-biclique invariants, to which end we use the following notation. We let $h(\mathbf{x}) = \{v \in V(G') : \mathbf{x}(v) \geq 1/\log n\}$ denote the set of vertices to which $\mathbf{x}$ assigns high values. Note that we can assume for each $v \in h(\mathbf{x})$, we have $\mathbf{x}(v) \leq 1$. Moreover, given a biclique M in G', we let $(\mathbf{x} \setminus M) : V(G') \to [0, \infty)$ denote the function that assigns 0 to any vertex in M and $(1 + \frac{4}{\log n})\mathbf{x}(v)$ to any other vertex $v \in V(G')$. Now, to adjust $\mathbf{x}$ to be of the desired form both at this phase and at later recursive calls, we rely on the following lemmata.

Lemma 6.3. *Define $\widehat{G} = G' - h(\mathbf{x})$, $\widehat{w} = w'|_{V(\widehat{G})}$ and $\widehat{\mathbf{x}} = \mathbf{x}|_{V(\widehat{G})}$. Then, $c' \cdot \log n \cdot \log(\alpha(\widehat{G})) \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c' \cdot \log n \cdot \log(\alpha(G)) \cdot w'(\mathbf{x})$, where $c' \geq 1$.*

Proof. By the definition of $h(\mathbf{x})$, it holds that $w'(\widehat{\mathbf{x}}) \leq w'(\mathbf{x}) - \frac{1}{\log n} \cdot w'(h(\mathbf{x}))$. Since $\widehat{G}$ is an induced subgraph of G', it also holds that $\alpha(\widehat{G}) \leq \alpha(G')$. Thus, $c' \cdot \log n \cdot \log(\alpha(\widehat{G})) \cdot w'(\widehat{\mathbf{x}}) + w'(h(\mathbf{x})) \leq c' \cdot \log n \cdot \log(\alpha(G')) \cdot (w'(\mathbf{x}) - \frac{1}{\log n} \cdot w'(h(\mathbf{x}))) + w'(h(\mathbf{x})) \leq c' \cdot \log n \cdot \log(\alpha(G)) \cdot w'(\mathbf{x})$. □

Thus, it is safe to update G' to $G' - h(\mathbf{x})$, w' to $w'|_{V(\widehat{G})}$, H' to $H' - h(\mathbf{x})$ and $\mathbf{x}$ to $\mathbf{x}|_{V(\widehat{G})}$, where we ensure that once we obtain a solution to the new instance, we add $w'(h(\mathbf{x}))$ to this solution and $h(\mathbf{x})$ to the set realizing it. Note that if the solution for the new instance satisfies the no-zero solution invariant, then the solution obtained by adding $h(\mathbf{x})$ to it also satisfies the invariant.

Lemma 6.4. *Let Q be a chordless cycle on at least 5 vertices and M be a biclique in G' with vertex partitions as $V(M) = M_1 \uplus M_2$ such that $V(Q) \cap V(M) \neq \emptyset$. Then there is a chordless cycle Q' on at least 5 vertices that intersects M in at most 3 vertices such that $E(Q' \setminus V(M)) \subseteq E(Q \setminus V(M))$, and Q' is of one of the following three types.*

- *$Q' \cap M$ is a single vertex*
- *$Q' \cap M$ is an edge in $G[M]$*
- *$Q' \cap M$ is an induced path on 3 vertices in M.*

Proof. Observe that no chordless cycle on more than 50 vertices may contain two vertices from each of M_1 and M_2, as that would imply a chord in it. Now, if the chordless cycle Q already satisfies the required conditions we output it as Q'.

First consider the case, when $Q \cap M$ contains exactly two vertices that do not have an edge between them. Then the two vertices, say v_1, v_2, are both either in M_1 or in M_2. Suppose that they are both in M_1 and consider some vertex $u \in M_2$ (as M is a biclique, we have $M_1, M_2 \neq \emptyset$). Let P_1 is the longer of the two path segments of Q between v_1 and v_2, and note that it must have at least 3 edges. Then observe that $G'[V(P_1) \cup \{u, v_1, v_2\}]$ contains a DH-obstruction, Q', as v_1, v_2 have different distances depending on if u is included in an induced subgraph or not. As Q is a chordless cycle, P_1 is an induced path in G'. Thus, Q' must contain u. If Q' contains none of v_1, v_2, then it is an obstruction that contains only u from M. Otherwise, Q' either contains an edge, or an induced path on 3 vertices, from M. Also as G' has no small obstructions, Q' is a chordless cycle on more than 50 vertices.

Now consider the case when $Q \cap M$ contains exactly three vertices. Observe that it cannot contain two vertices of M_1 and one vertex of M_2, or vice versa, as Q does not satisfy the required conditions. Therefore, $Q \cap M$ contains exactly three vertices from M_1 (or from M_2), which again do not form an induced path of length 3. So there is an independent set of size 2 in $Q \cap M$, and now, as before, we can again obtain the chordless cycle Q' on at least 5 vertices with $E(Q' \setminus M) \subseteq E(Q \setminus M)$. Before we consider the other cases, we have the following claim.

Claim 1. *Let M be a biclique in G' with vertex partition as $V(M) = M_1 \uplus M_2$. Then $G'[M]$ has no induced P_4.*

Proof. Let P be any induced path of length 4 in $G'[M]$. Then, either $V(P) \subseteq M_1$ or $V(P) \subseteq M_2$. Now consider any such path P in M_1 and some vertex $u \in M_2$. Then $G'[P \cup \{u\}]$ contains a DH-obstruction of size 5 which is a contradiction to the fact that G' has no small obstructions. □

Next, let $Q \cap M$ contain 4 or more vertices. Note that in this case all these vertices are all either in M_1 or in M_2 since otherwise, Q would not be a chordless cycle in G' on at least 5 vertices. Let us assume these vertices lie in M_1 (other case is symmetric). Let $v_1, v_2, v_3, \cdots, v_\ell \in M_1 \cap Q$ be the sequence of vertices obtained when we traverse Q starting from an arbitrary vertex, where $\ell \geq 4$. By Claim 1 they cannot form an induced path on 4 vertices, i.e. $G'[V(Q) \cap M_1]$ consists of at least two connected components. Without loss of generality we may assume that v_1 and v_ℓ are in different components. Observe that the only possible edges between these vertices may be at most two of the edges (v_1, v_2), (v_2, v_3), and (v_3, v_ℓ). Hence, we conclude that either v_1, v_3 or v_2, v_ℓ are a distance of at least 3 in Q. Let us assume that v_2, v_ℓ are at distance 3 or more in Q, and the other case is symmetric and $P_{23}, P_{3\ell}$ be the paths not containing v_1 in Q between v_2 and v_3, and

v_3 and v_ℓ, respectively. Notice that for any $u \in M_2$ the graph $G'[\{u\} \cup V(P_{23}) \cup V(P_{3\ell})]$ contains a DH-obstruction. Since the graph is free of all small obstruction, this DH-obstruction, denoted $\hat{Q}$, must be a chordless cycle on at least 5 vertices. Furthermore this obstruction can contain at most 2 vertices from $\{v_2, v_3, v_\ell\}$, as otherwise there would be a chord in it. Hence $\hat{Q} \cap M$ contains strictly fewer vertices than $Q \cap M$. Moreover, we have $E(\hat{Q} \setminus M) \subseteq E(Q \setminus M)$. Now, by a recursive application of this lemma to $\hat{Q}$, we obtain the required Q'. □

A consequence of the above lemma is that, whenever M is a biclique in G', we may safely ignore any DH-obstruction that intersects M in more than 3 vertices. This leads us to the following lemma.

Lemma 6.5. *Given a biclique M in G', the function $(\mathbf{x} \setminus M)$ is a valid fractional solution such that $w'(\mathbf{x} \setminus M) \leq (1 + \frac{4}{\log n})w'(\mathbf{x})$.*

Proof. To prove that $(\mathbf{x} \setminus M)$ is a valid fractional solution, let Q be some chordless cycle (not on 4 vertices) in G'. We need to show that $(\mathbf{x} \setminus M)(V(Q)) \geq 1$. By our assumption Q can contain at most 3 vertices from M. Thus, since $\mathbf{x}$ is a valid fractional solution, it holds that $\mathbf{x}(V(Q) \setminus V(M)) \geq 1 - \frac{3}{\log n}$. By the definition of $(\mathbf{x} \setminus M)$, this fact implies that $(\mathbf{x} \setminus M)(V(Q)) = (\mathbf{x} \setminus M)(V(Q) \setminus V(M)) \geq (1 + \frac{4}{\log n})(1 - \frac{3}{\log n}) = 1 + \frac{1}{\log n} - \frac{12}{(\log n)^2} \geq 1$, where the last inequality relies on the assumption $n \geq 2^{12}$.

For the proof of the second part of the claim, note that $w'(\mathbf{x} \setminus M) = (1 + \frac{4}{\log n})\, w'(\mathbf{x}|_{V(G') \setminus V(M)}) \leq (1 + \frac{4}{\log n})w'(\mathbf{x})$. □

We call DHD-APPROX recursively with the fractional solution $(\mathbf{x} \setminus C)$, and by Lemma 6.5, $w'(\mathbf{x} \setminus C) \leq (1 + \frac{4}{\log n})w'(\mathbf{x})$. If Lemma 6.2 were true, we return a solution that is at least opt and at most $(\frac{\log n}{\log n + 4}) \cdot c \cdot \log n \cdot \log(\alpha(G)) \cdot w(\mathbf{x} \setminus M) \leq c \cdot \log n \cdot \log(\alpha(G)) \cdot w(\mathbf{x})$ as desired. In other words, to prove Lemma 6.1, it is sufficient that we next focus only on the proof of Lemma 6.2. The proof of this lemma is done by induction. When we consider some recursive call, we assume that the solutions returned by the additional recursive calls that it performs, which are associated with graphs $\widetilde{G}$ such that $\alpha(\widetilde{G}) \leq \frac{3}{4}\alpha(G')$, complies with the conclusion of the lemma.

Termination. Once G' becomes a distance hereditary graph, we return 0 as our solution and $\emptyset$ as the set that realizes it. Clearly, we thus satisfy the demands of Lemma 6.2 and the no-zero solution invariant is satisfied. In fact, we thus also ensure that the execution of our algorithm terminates once $\alpha(G') < \log n$.

Lemma 6.6. *If $\alpha(G') < \log n$, then G' is a distance hereditary graph.*

Proof. Suppose that G' is not a distance hereditary graph. Then, it contains an obstruction Q. Since $\mathbf{x}$ is a valid fractional solution, it holds that $\mathbf{x}(V(Q)) \geq 1$. But $\mathbf{x}$ satisfies the low-value invariant therefore, it holds that $\mathbf{x}(V(Q)) < |V(Q)|/\log n$. These two observations imply that $|V(Q)| > \log n$. Furthermore, at least $\log n$ of these vertices are assigned a non-zero value by $\mathbf{x}$, i.e. $\alpha(G') \geq \log n$. Therefore, if $\alpha(G') < \log n$, then G' must be a distance hereditary graph. □

The fact that, the recursive calls are made onto graphs where the distance hereditary subgraph contains at most $3|V(G')|/4$ vertices that have non-zero values, we observe the following.

Observation 16. *The maximum depth of the recursion tree is bounded by $q \cdot \log n$ for some fixed constant q.*

Recursive Call. Since H' is a distance hereditary graph, it has a rank-width-one decomposition $(\mathcal{T}, \phi)$, where $\mathcal{T}$ is a binary tree and ϕ is a bijection from $V(G')$ to the leaves of $\mathcal{T}$. Furthermore, rank-width of $\mathcal{T}$ is 1, which means that for any edge of the tree, by deleting it, we obtain a partition of the leaves in $\mathcal{T}$. This partition induces a cut of the graph, where the set of edges crossing this cut forms a biclique M, with vertex partition as $V(M) = M_1 \uplus M_2$ in the graph. By standard arguments on trees, we deduce that $\mathcal{T}$ has an edge that defines a partition such that after we remove the biclique edges between M_1 and M_2 from G' we obtain two (not necessarily connected) graphs, H_1 and H_2, such that $|V(H_1)|, |V(H_2)| \leq \frac{3}{4}|V(H')|$ and $M_1 \subseteq H_1, M_2 \subseteq H_2$. Note that the bicliques M and C are vertex disjoint. We proceed by replacing the fractional solution $\mathbf{x}$ by $(\mathbf{x} \setminus M)$. For the sake of clarity, we denote $\mathbf{x}^* = (\mathbf{x} \setminus M)$. Let $G_1 = G'[V(H_1) \cup V(C) \cup V(M)]$, $G_2 = G'[V(H_2) \cup V(C) \cup V(M)]$.

We adjust the current instance by relying on Lemma 6.3 so that $\mathbf{x}^*$ satisfies the low-value invariant (in the same manner as it is adjusted in the initialization phase). In particular, we remove $h(\mathbf{x}^*)$ from G',H', G_1, H_1, G_2 and H_2, and we let $(G^*, w^*, C, H^*, \mathbf{x}^*)$, G_1^*, H_1^* G_2^* and H_2^* denote the resulting instance and graphs. Observe that, now we have $\alpha(G_1^*), \alpha(G_2^*) \leq \frac{3}{4}\alpha(G')$. We will return a solution that is at least opt and at most $(\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log n \cdot \log(\alpha(G')) \cdot w^*(\mathbf{x}^*)$, along with a set that realizes it.[2] In the analysis we will argue this it is enough for our purposes.

Next, we define two subinstances, $I_1^* = (G_1^*, w^*|_{V(G_1^*)}, C, H_1^*, \mathbf{x}^*|_{V(G_1^*)})$ and $I_2^* = (G_2^*, w^*|_{V(G_2^*)}, C, H_2^*, \mathbf{x}^*|_{V(G_2^*)})$. We solve each of these subinstances by a recursive call to DHD-APPROX, and thus we obtain two solutions of sizes, s_1^* to I_1^* and s_2^* to I_2^*, and two sets that realize these solutions, S_1^* and S_2^*. Bty the inductive hypothesis, we have the following observations.

Observation 17. *$S_1^* \cup S_2^*$ intersects any chordless cycle on at least 6 vertices in G^* that lies entirely in either G_1^* or G_2^*. Furthermore, for $i \in \{1, 2\}$, S_i^* does not contain any vertex $v \in V(G_i^*)$, where $\mathbf{x}^*|_{V(G_i^*)}(v) = 0$.*

Observation 18. *Given $i \in \{1, 2\}$, $s_i^* \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log n \cdot \log(\alpha(G_i^*)) \cdot w(\mathbf{x}_i^*)$.*

Moreover, since $\mathbf{x}^*(V(C) \cup V(M)) = 0$, we also have the following observation.

Observation 19. *$w^*(\mathbf{x}_1^*) + w^*(\mathbf{x}_2^*) = w^*(\mathbf{x}^*)$.*

We say that a cycle in G^* is *bad* if it is a chordless cycle not on four vertices that belongs entirely to neither G_1^* nor G_2^*. We will follow approach similar to Section 4.1, for handling these cycles. Next, we formally describe how we intersect bad cycles.

Bad Cycles. As the graph no longer contains small obstructions, it is clear that any remaining obstruction is a chordless cycle on at more than 50 vertices and is a *bad cycle.* Let $\widetilde{G}_1 = G_1^* - S_1^*$ and $\widetilde{G}_2 = G_2^* - S_2^*$. Next we introduce the notion of *important bad cycles*; we will later show that considering only these cycles in enough for handling all the bad cycles.

Definition 6.7. We say that a bad cycle Q is *important*, if the following conditions are satisfied.

(1) For each $X \in \{C, M\}$, we have $V(Q) \cap V(X)$ is either a single vertex, it induces an edge, or it induces a path on three vertices in G'.
(2) $Q \setminus (V(M) \cup V(C))$ has exactly two connected components, $P_1' = P_1'(Q)$ and $P_2' = P_2'(Q)$, where $V(P_1') \subseteq V(\widetilde{G}_1 \setminus (V(M) \cup V(C)))$ and $V(P_2') \subseteq V(\widetilde{G}_2 \setminus (V(M) \cup V(C)))$. For $i \in \{1, 2\}$, let v_i and u_i be the neighbors in Q, of the endpoints a_i and b_i of P_i', respectively. Then, $v_i \in V(C)$ and $u_i \in V(M)$.

[2]Here, the coefficient $(\frac{\log n}{\log n+4})^{\delta}$ has been replaced by the smaller coefficient $(\frac{\log n}{\log n+4})^{\delta+1}$.

For $i \in \{1,2\}$, the path $P_i(Q)$ between v_i and u_i, that contains P'_i as the subpath and contains the edges $\{v_i, a_i\}$ and $\{b_i, u_i\}$, is the $\widetilde{G}_i$*-part* of Q. Also, we say that (v_1, u_1, v_2, u_2) is an *important quadruple* of vertices, where we have $v_1, v_2 \in V(C)$ and $u_1, u_2 \in V(C)$.[3]

In the next lemma we will show existence of a special important bad cycle, for every bad cycle.

Lemma 6.8. *For any a bad cycle Q', there is also an important bad cycle Q such that $E(Q \setminus (V(C) \cup V(M))) \subseteq E(Q' \setminus (V(C) \cup V(M)))$.*

Proof. Let Q' be a bad cycle. Let us recall that the input graph G' can be partitioned into the biclique C and a distance hereditary graph H'. Hence $Q' \cap C \neq \emptyset$. We apply Lemma 6.4 to obtain a chordless cycle $\widehat{Q}$, on at least 50, which either has a single vertex from C, or an edge from C, or an induced path on three vertices from C. Notice that if $\widehat{Q} \cap M = \emptyset$, then using Observation 17 we can obtain that $\widehat{Q}$ is a bad chordless cycle in $\widetilde{G}_1$, or in $\widetilde{G}_2$, which is a contradiction. Thus we can conclude that $V(\widehat{Q}) \cap V(M) \neq \emptyset$. Since, $M \cap C = \emptyset$, again by using Lemma 6.4 we can obtain a bad chordless cycle Q, on at least 50, which either has a single vertex from M, or an edge from M, or an induced path on three vertices from M. Notice that Q is an important bad cycle with $E(Q \setminus (V(C) \cup V(M))) \subseteq E(Q' \setminus (V(C) \cup V(M)))$. □

The above lemma (Lemma 6.8) implies that it is safe to restrict our attention to the subset of important bad cycles. In the next lemma we prove some useful properties of important bad cycles.

Lemma 6.9. *Consider an important bad cycle Q, and let P_1 and P_2 be the $\widetilde{G}_1$-part and $\widetilde{G}_2$-part of it, respectively. For any $i \in \{1,2\}$ and an induced path $\widehat{P}_i$ in $\widetilde{G}_i$, with the same endpoints as P_1, whose no internal vertex is $V(C) \cup V(M)$, $Q' = G'[(V(Q) \cap (V(M) \cup V(C))) \cup V(\widehat{P}_1) \cup V(P_2)]$ is also an important bad cycle in $G' \setminus (S_1^* \cup S_2^*)$.*

Proof. Observe that, P_1 and $\widehat{P}_1$ are paths between the same endpoints in $\widetilde{G}_1$, which is a distance hereditary graph. Therefore, $\widehat{P}_1$ is an induced path of the same length as P_1. Note that no internal vertex of $\widehat{P}_1$ can be adjacent to an internal vertex of P_2. An internal vertex of $\widehat{P}_1$ cannot have a neighbor in $V(Q) \cap (V(C) \cap V(M))$, as otherwise, the distance hereditary property of $\widetilde{G}_1$ would be violated. Thus we can conclude that Q' is an important bad cycle in $G' \setminus (S_1^* \cup S_2^*)$. □

The above lemma will allow us to reduce the problem of computing a solution that intersects all important bad cycles, to computing a solution for an instance of Weighted Multicut. To this end, we will compute important quadruples and define some subsets of paths these quadruples.

Compute the set ImpQrd, of all important quadruples (see Definition 6.7). Notice that ImpQrd can be computed in polynomial time. For $(v, u, v', u') \in$ ImpQrd, let $\mathcal{P}_1(v, u)$ denote the set of any induced path P_1 between v and u whose internal vertices belong only to $\widetilde{G}_1$. Symmetrically, we let $\mathcal{P}_2(v, u)$ denote the set of any path P_2 between v and u whose internal vertices belong only to $\widetilde{G}_2$.

We now examine how the fractional solution $\mathbf{x}^*$ handles pairs (v, u) of vertices $v \in V(C)$ and $u \in V(M)$.

Lemma 6.10. *Consider an important quadruple $(v, u, v', u') \in$ ImpQrd. If for some $i \in \{1,2\}$, there is $P \in \mathcal{P}_i(v, u)$, with $\mathbf{x}^*(V(P)) < 1/2$, then for $j \in \{1,2\} \setminus \{i\}$, we have $\mathbf{x}^*(V(P')) \geq 1/2$, for every $P' \in \mathcal{P}_j(v', u')$.*

[3]Possibly, $v_1 = v_2$ or $u_1 = u_2$ (or both). We note that the same quadruple may be important with respect to many bad cycles. Whenever we refer to an important quadruple, then there will exist at least one important bad cycle that certifies that it is important.

Proof. Suppose, by way of contradiction, that the lemma is incorrect. Consider an important quadruple $(v, u, v', u') \in \mathsf{ImpQrd}$ for which the condition of the lemma is not satisfied. Suppose that there is a path $P_1 \in \mathcal{P}_1(v, u)$ such that $\mathbf{x}^*(V(P_1)) < 1/2$, and a path $P_2 \in \mathcal{P}_2(v', u')$ such that $\mathbf{x}^*(V(P_2)) < 1/2$. (All other cases can be handled symmetrically.) As $(v, u, v', u') \in \mathsf{ImpQrd}$, there is a bad cycle Q, for which this is an important quadruple. Now using Lemma 6.9, we can obtain a DH-obstruction, Q', where $V(Q') \subseteq \{v, u, v', u'\} \cup V(P_1) \cup V(P_2)$. The above together with the fact that for each $w \in V(C) \cup V(M)$, $\mathbf{x}^*(w) = 0$, contradicts that $\mathbf{x}^*$ is a valid fractional solution. □

Given $i \in \{1, 2\}$, let $2\mathbf{x}_i^*$ denote the fractional solution that assigns to each vertex the value assigned by $\mathbf{x}_i^*$ times 2. Let $\widetilde{H}_1$ and $\widetilde{H}_2$ be the graphs $\widetilde{G}_1$ and $\widetilde{G}_2$, restricted to vertices in H'. That is, $\widetilde{H}_1 = \widetilde{G}_1[V(\widetilde{G}_1) \cap V(H')]$ and $\widetilde{H}_2 = \widetilde{G}_2[V(\widetilde{G}_2) \cap V(H')]$. For every $(v, u, v', u') \in \mathsf{ImpQrd}$, we perform the following operation. We initialize $\mathcal{T}_1(v, u, v', u') = \mathcal{T}_2(v, u, v', u') = \emptyset$. Insert each pair in $\{(a, b) : a \in N_{\widetilde{G}_1}(v) \cap V(\widetilde{H}_1), b \in N_{\widetilde{G}_1}(u) \cap V(\widetilde{H}_1), \widetilde{H}_1$ has a path between a and $b\}$ into $\mathcal{T}_1(v, u, v', u')$. Next, we insert each pair in $\{(a', b') : a' \in N_{\widetilde{G}_2}(v') \cap V(\widetilde{H}_2), b' \in N_{\widetilde{G}_2}(u') \cap V(\widetilde{H}_2)$, $\widetilde{H}_2$ has a path between a' and $b'\}$ into $\mathcal{T}_2(v, u, v', u')$. We remark that the vertices in a pair in $\mathcal{T}_1(v, u, v', u')$ or $\mathcal{T}_1(v, u, v', u')$ are not necessarily distinct. The following observation follows from Lemma 6.9 and 6.10.

Observation 20. *For $(v, u, v', u') \in \mathsf{ImpQrd}$, there exists an index $i = i(v, u, v', u') \in \{1, 2\}$, such that for every $(a, b) \in \mathcal{T}_i(v, u, v', u')$ and for every path P between a and b in $\widetilde{H}_i$, we have $2\mathbf{x}_i^*(V(P)) \geq 1$.*

Proof. Note that for $(a, b) \in \mathcal{T}_1(v, u, v', u')$ and a path P_1' between a and b in $\widetilde{H}_1$, the path P_1, where $V(P_1) = V(P_1') \cup \{v, u\}$ and $E(P_1) = E(P_1') \cup \{v, a\}, \{u, b\}$, belongs to the set $\mathcal{P}_1(v, u)$. Similarly, for $(a', b') \in \mathcal{T}_2(v, u, v', u')$ and a path P_2' between a' and b' in $\widetilde{H}_2$, the path P_2, where $V(P_2) = V(P_2') \cup \{v', u'\}$ and $E(P_2) = E(P_2') \cup \{v', a'\}, \{u', b'\}$, belongs to the set $\mathcal{P}_2(v', u')$. Thus using Lemma 6.9 and 6.10, and the fact that $\mathbf{x}^*(V(C) \cup V(M)) = 0$, we can obtain that there is $i = i(v, u, v', u') \in \{1, 2\}$, such that for every $(a, b) \in \mathcal{T}_i(v, u, v', u')$ and for every path P between a and b in $\widetilde{H}_i$, we have $2\mathbf{x}_i^*(V(P)) \geq 1$. □

The following lemma translates Observation 20 into an algorithm.

Lemma 6.11. *For every $(v, u, v', u') \in \mathsf{ImpQrd}$, one can compute (in polynomial time) an index $i = i(v, u, v', u') \in \{1, 2\}$ such that for every $(a, b) \in \mathcal{T}_i(v, u)$ and every path P in $\widetilde{H}_i$ between a and b, we have $2\mathbf{x}_i^*(V(P)) \geq 1$.*

Proof. Consider $(v, u, v', u') \in \mathsf{ImpQrd}$. If there is $i \in \{1, 2\}$ such that $\mathcal{T}_i(v, u, v', u') = \emptyset$, then we have trivially obtained the required index which is $i(v, u, v', u') = i$. Otherwise, we proceed as follows. For any index $i \in \{1, 2\}$, we perform the following procedure. For each pair $(a, b) \in \mathcal{T}_i(v, u)$, we use Dijkstra's algorithm to compute the minimum weight of a path between a and b in the graph $\widetilde{H}_i$ where the weights are given by $2\mathbf{x}_i^*$. In case for every pair (a, b) the minimum weight is at least 1, we have found the desired index $i(v, u, v', u')$. Moreover, by Observation 20, for at least one index $i \in \{1, 2\}$, the minimum over the minimum weights associated with the pairs (a, b) should be at least 1 (if this value is at least 1 for both indices, we arbitrarily decide to fix $i(v, u, v', u') = 1$). □

At this point, we need to rely on approximate solutions to the Weighted Multicut problem which is given by theorem below (Theorem 6.12).

Theorem 6.12 ([19]). *Given an instance of* Weighted Multicut, *one can find (in polynomial time) a solution that is at least* opt *and at most $d \cdot \log n \cdot$* fopt *for some fixed constant $d > 0$, along with a set that realizes it.*

Recall that we additionally need to satisfy the no-zero solution invariant, and thus we need to make sure that while using the algorithm given by Theorem 6.12, we do not put any vertex v in the solution for which $\mathbf{x}^*(v) = 0$. To this end, for $i \in \{1, 2\}$, we construct a new weight function $\widetilde{w}_i : V(\widetilde{H}_i) \longrightarrow \mathbb{R}$, where for each $v \in V(\widetilde{H}_i)$, if $\mathbf{x}_i^*(v) \neq 0$, then $\widetilde{w}_i(v) = w'(v)$, and otherwise $\mathbf{x}_i^*(v) = 0$ and we set $\widetilde{w}_i(v) = d \cdot \log n \cdot (\sum_{u \in \widehat{H}_i} w'(u) \cdot \mathbf{x}_i^*(u)) + 1$, where d is the constant given by Theorem 6.12.

We are now ready to create our Weighted Multicut instances. The first instance is $J_1 = (\widetilde{H}_1, \widetilde{w}_1, \mathcal{T}_1)$ and the second instance is $J_2 = (\widetilde{H}_2, \widetilde{w}_2, \mathcal{T}_2)$, where the sets $\mathcal{T}_1$ and $\mathcal{T}_2$ are defined as follows. We initialize $\mathcal{T}_1 = \emptyset$. For every $(v, u, v', u') \in \mathsf{ImpQrd}$, if $i(v, u, v', u') = 1$ (see Lemma 6.11), we insert each pair in $\mathcal{T}_1(v, u, v', u')$ into $\mathcal{T}_1$. The definition of $\mathcal{T}_2$ is symmetric to the one of $\mathcal{T}_1$.

As for all $v \in V(C) \cup V(M)$ it holds that $\mathbf{x}_1^*(v) = \mathbf{x}_2^*(v) = 0$, we deduce that $2\mathbf{x}_1^*$ and $2\mathbf{x}_2^*$ are valid solutions to J_1 and J_2, respectively. Moreover, for $i \in \{1, 2\}$ the construction of $\widetilde{w}_i$ ensures that $\widetilde{w}_i(v) \cdot \mathbf{x}_i^*(v) = w'(v) \cdot \mathbf{x}_i^*(v)$, for each $v \in V(\widetilde{H}_i)$. Thus, by calling the algorithm given by Theorem 6.12 with each instance, we obtain a solution r_1 to the first instance, along with a set R_1 that realizes it, such that $r_1 \leq 2d \cdot \log|V(\widetilde{H}_1)| \cdot \widetilde{w}_1(\mathbf{x}_1^*) \leq 2d \cdot \log|V(G_1^*)| \cdot w^*(\mathbf{x}_1^*)$, and we also obtain a solution r_2 to the second instance, along with a set R_2 that realizes it, such that $r_2 \leq 2d \cdot \log|V(\widetilde{H}_2)| \cdot \widetilde{w}_2(\mathbf{x}_2^*) \leq 2d \cdot \log|V(G_2^*)| \cdot w^*(\mathbf{x}_2^*)$. By the construction of $\widetilde{w}_1$ and $\widetilde{w}_2$, we can obtain that for no vertex $v \in R_1 \cup R_2$, we have $\mathbf{x}^*(v) = 0$. Now by Observation 17 and Lemma 6.8, we have obtained a set $S^* = S_1^* \cup S_2^* \cup R_1 \cup R_2$ for which we have the following observation.

Observation 21. *S^* intersects any chordless cycle in G^*. Furthermore, $w^*(S^*) \leq s_1^* + s_2^* + r_1 + r_2$ and there is no $v \in S^*$, such that $\mathbf{x}^*(v) = 0$.*

We start by showing that $s_1^* + s_2^* + r_1 + r_2 + w(h(x)) \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log n \cdot \log(\alpha(G')) \cdot w^*(\mathbf{x}^*)$. Recall that for any $i \in \{1, 2\}$, $r_i \leq 2d \cdot \log|V(G_i^*)| \cdot w^*(\mathbf{x}_i^*)$. Thus, by Observation 18 and since for any $i \in \{1, 2\}$, $|V(G_i^*)| \leq n$ and $\alpha(G_i^*) \leq \frac{3}{4}\alpha(G')$, we have that

$$w^*(S^*) \leq (\frac{\log n}{\log n + 4})^{\delta+1} \cdot c \cdot \log n \cdot \log(\frac{3}{4}\alpha(G')) \cdot (w^*(\mathbf{x}_1^*) + w^*(\mathbf{x}_2^*)) + 2d \cdot \log n \cdot (w^*(\mathbf{x}_1^*) + w^*(\mathbf{x}_2^*)).$$

By Observation 19, we further deduce that

$$w^*(S^*) \leq \left((\frac{\log n}{\log n + 4})^{\delta+1} \cdot c \cdot \log(\frac{3}{4}\alpha(G')) + 2d\right) \cdot \log n \cdot w^*(\mathbf{x}^*).$$

Now, it only remains to show that $(\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log(\frac{3}{4}\alpha(G')) + 2d \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log \alpha(G')$, which is equivalent to $2d \leq (\frac{\log n}{\log n+4})^{\delta+1} \cdot c \cdot \log(\frac{4}{3})$. Observe that $\delta \leq q \cdot \log n - 1$ for some fixed constant q. Indeed, it initially holds that $\alpha(G) \leq n$, at each recursive call, the number of vertices assigned a non-zero value by $\mathbf{x}^*$ decreases to at most a factor of $3/4$ of its previous value, and the execution terminates once this value drops below $(\log n)/2$. Thus, it is sufficient to choose the constant c so that $2d \leq (\frac{\log n}{\log n+4})^{q \cdot \log n} \cdot c \cdot \log(\frac{4}{3})$. As the term $(\frac{\log n}{\log n+4})^{q \cdot \log n}$ is lower bounded by $1/(e^{4q})$, it is sufficient that we fix $c = 2 \cdot e^{4q} \cdot d \cdot 1/\log(\frac{4}{3})$.

Note that $2d \leq (\frac{\log n}{\log n+4})^{q \cdot \log n} \cdot c \cdot \log(\frac{4}{3})$, where $d \geq 1$. Therefore, $(\frac{\log n}{\log n+4})^{q \cdot \log n} \cdot c \geq 1$. This together with Lemma 6.3 and 6.5 implies that $w'(S^*) + w'(h(x)) \leq (\frac{\log n}{\log n+4})^{\delta} \cdot c \cdot \log(\alpha(G'))w'(x)$, which proves Lemma 6.2.

6.2 General Graphs

In this section we handle general instances by developing a $d \cdot \log^2 n$-factor approximation algorithm for WDHVD, Gen-DHD-APPROX, thus proving the correctness of Theorem 1.4.

The Recursive Algorithm. We define each call to our algorithm Gen-DHD-APPROX to be of the form (G', w'), where (G', w') is an instance of WDHVD such that G' is an induced subgraph of G, and we denote $n' = |V(G')|$. We ensure that after the initialization phase, the graph G' never contains a DH-obstruction on at most 50 vertices. We call this invariant the *$\mathcal{O}_{50}$-free invariant.* In particular, this guarantee ensures that the graph G' always contains only a small number of maximal bicliques, as stated in the following lemma.

Lemma 6.13 (Lemma 3.5 [32]). *Let G be a graph on n vertices with no DH-obstruction on at most 6 vertices. Then G contains at most $(n^3 + 5n)/6$ maximal bicliques, and they can be enumerated in polynomial time.*

Goal. For each recursive call Gen-DHD-APPROX(G', w'), we aim to prove the following.

Lemma 6.14. Gen-DHD-APPROX *returns a solution that is at least* opt *and at most* $\frac{d}{2} \cdot \log^3 n' \cdot \text{opt}$. *Moreover, it returns a subset $U \subseteq V(G')$ that realizes the solution. Here d is a constant, which will be determined later.*

At each recursive call, the size of the graph G' becomes smaller. Thus, when we prove that Lemma 6.14 is true for the current call, we assume that the approximation factor is bounded by $\frac{d}{2} \cdot \log^3 \widehat{n} \cdot \text{opt}$ for any call where the size $\widehat{n}$ of the vertex-set of its graph is strictly smaller than n'.

Initialization. We are given (G, w) as input, and first we need to ensure that the $\mathcal{O}_{50}$-free invariant is satisfied. For this purpose, we update G as follows. First, we let $\mathcal{O}_{50}$ denote the set of all DH-obstruction on at most 50 vertices of G. Clearly, $\mathcal{O}_{50}$ can be computed in polynomial time and it holds that $|\mathcal{O}_{50}| \leq n^{\mathcal{O}(1)}$. Now, we construct an instance of Weighted 50-Hitting Set, where the universe is $V(G)$, the family of all sets of size at most 50 in $\mathcal{O}_{50}$, and the weight function is w'. Since each DH-obstruction must be intersected, therefore, the optimal solution to our Weighted 50-Hitting Set instance is at most opt. By using the standard c'-approximation algorithm for Weighted c'-Hitting Set [33], which is suitable for any fixed constant c', we obtain a set $S \subseteq V(G)$ that intersects all the DH-obstruction in $\mathcal{O}_{50}$ and whose weight is at most $50 \cdot \text{opt}$. Having the set S, we remove its vertices from G to obtain the graph G', and $w' = w|_{G'}$. Now that the $\mathcal{O}_{50}$-free invariant is satisfied, we can call Gen-DHD-APPROX on (G', w') and to the outputted solution, we add $w(S)$ and S.

We note that during the execution of the algorithm, we update G' only by removing vertices from it, and thus it will always be safe to assume that the $\mathcal{O}_{50}$-free invariant is satisfied. Now, by Lemma 6.14, we obtain a solution of weight at most $\frac{d}{2} \cdot \log^3 n \cdot \text{opt} + 50 \cdot \text{opt} \leq d \cdot \log^3 n \cdot \text{opt}$, then combined with S, it allows us to conclude the correctness of Theorem 1.4.

Termination. Observe that due to Lemma 6.13, we can test in polynomial time, if our current graph G' is of the special kind that can be partitioned into a biclique and a distance hereditary graph: we examine each maximal biclique of G', and check whether after its removal we obtain a distance hereditary graph. Once G' becomes such a graph that consists of a biclique and a distance hereditary graph, we solve the instance (G', w') by calling algorithm DHD-APPROX. Observe that this returns a solution of value $\mathcal{O}(\log^2 n \cdot \text{opt})$ which is also $\mathcal{O}(\log^3 n \cdot \text{opt})$.

Recursive Call. Similar to the case for WCVD, instead computing a balanced separators with a maximal clique and some additional vertices, here we find a balanced separator that comprises of a biclique and some additional, but small number of vertices. Existence of such a separator is guaranteed by Lemma 6.15. From Lemma 6.13, it follows that the graph with no DH-obstruction of size at most 50 contains at most $O(n^3)$ maximal bicliques and they can enumerated in polynomial time. We use the weighted variant of Lemma 3.8 from [32] in Lemma 6.15. The proof of Lemma 6.15 remains exactly the same as that in Lemma 3.8 of [32].

Lemma 6.15 (Lemma 3.8 [32]). *Let G' be a connected graph on n' vertices not containing any DH-obstruction of size at most* 50 *and $w : V(G) \longrightarrow \mathbb{R}$ be a weight function. Then in polynomial time we can find a balanced vertex separator $K \uplus X$ such that the following conditions are satisfied.*

- *K is a biclique in G or an empty set;*
- $w(X) \leq q \cdot \log n' \cdot \text{opt}$, *where q is some fixed constant.*

Here, opt *is the weight of the optimum solution to WDHVD of G.*

We note that we used the $O(\log n')$-factor approximation algorithm by Leighton and Rao [34] in Lemma 6.15 to find the balanced separator, instead of the $O(\sqrt{\log n'})$-factor approximation algorithm by Feige et al. [12], as the algorithm by Feige et al. is randomized. Let us also remark that if K is a biclique, then there is a bipartition of the vertices in K into $A \uplus B$, where both A and B are non-empty, which will be crucially required in later arguments.

Next, we apply in Lemma 6.15 to (G', w') to obtain a pair (K, X). Since $K \cup X$ is a balanced separator for G', we can partition the set of connected components of $G' \setminus (M \cup S)$ into two sets, $\mathcal{A}_1$ and $\mathcal{A}_2$, such that for $V_1 = \bigcup_{A \in \mathcal{A}_1} V(A)$ and $V_2 = \bigcup_{A \in \mathcal{A}_2} V(A)$ it holds that $n_1, n_2 \leq \frac{2}{3}n'$ where $n_1 = |V_1|$ and $n_2 = |V_2|$. We then define two inputs of (the general case) WDHVD: $I_1 = (G'[V_1], w'_{V_1})$ and $I_2 = (G'[V_2], w'_{V_2})$. Let opt_1 and opt_2 denote the optimal solutions to I_1 and I_2, respectively. Observe that since $V_1 \cap V_2 = \emptyset$, it holds that $\text{opt}_1 + \text{opt}_2 \leq \text{opt}$. We solve each of the two sub-instances by recursively calling algorithm Gen-DH-APPROX. By the inductive hypothesis, we obtain two sets, S_1 and S_2, such that $G'[V_1] \setminus S_1$ and $G'[V_2] \setminus S_2$ are both distance hereditary graphs, and $w'(S_1) \leq \frac{d}{2} \cdot \log^3 n_1 \cdot \text{opt}_1$ and $w'(S_2) \leq \frac{d}{2} \cdot \log^3 n_2 \cdot \text{opt}_2$. Now, if K were an empty set then it is easy to see that $X \cup S_1 \cup S_2$ is a feasible solution to the instance (G', w'). Now let us bound the total weight of this subset.

$$\begin{aligned}
w'(X \cup S_1 \cup S_2) &\leq w'(X) + w'(S_1) + w'(S_2)\\
&\leq q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot (\log^3 n_1 \cdot \text{opt}_1 + \log^3 n_2 \cdot \text{opt}_2)\\
&\quad \text{Recall that } n_1, n_2 \leq \tfrac{2}{3}n' \text{ and } \text{opt}_1 + \text{opt}_2 \leq \text{opt}.\\
&< q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot \log^3 \tfrac{2}{3}n' \cdot \text{opt}\\
&< \tfrac{d}{2} \cdot \log^3 n' \cdot \text{opt}
\end{aligned}$$

The more interesting case is when K is a biclique. Then, we first remove $X \cup S_1 \cup S_2$ from the graph, and note that the above bound also holds for this subset of vertices. Now observe that the graph $G'' = G' - (X \cup S_1 \cup S_2)$ can be partitioned into a biclique K and a distance hereditary graph $H = G[(V_1 \cup V_2) \setminus (S_1 \cup S_2)]$, along with the weight function $w'' = w'_{V(G'')}$. Thus we have an instance of the Biclique + Distance Hereditary Graph spacial case of WDHVD. Furthermore, note that we retained a fractional feasible solution $\mathbf{x}$ to the LP of the initial input G', w', which upperbounds the value of a fractional feasible solution $\mathbf{x}''$ to the LP of the instance G'', w''. We apply the algorithm DHD-APPROX on $(G'', w'', K, H, \mathbf{x}'')$ which outputs a solution S such that $w''(S) = w'(S) = O(\log^2 n \cdot \text{opt})$.

Observe that, any obstruction in $G' \setminus S$ is either completely contained in $G'[V_1 \setminus S]$, or completely contained in $G'[V_2 \setminus S]$ or it contains at least one vertex from K. This observation, along with the fact that $G'[(V_1 \cup V_2 \cup K) \setminus (S_1 \cup S_2 \cup \widehat{S})]$ is a distance hereditary graph, implies that $G' \setminus T$ is a distance hereditary graph where $T = X \cup S_1 \cup S_2 \cup \widehat{S}$. Thus, it is now sufficient to show that $w'(T) \leq \frac{d}{2} \cdot (\log n')^3 \cdot \text{opt}$. By the discussion above, we have that DHD-APPROX returns a solution of value $c \log^2 n \cdot \text{opt}$, where c is some constant.

$$\begin{aligned} w'(T) &\leq w'(S) + w'(S_1) + w'(S_2) + w'(\widehat{S_1}) + w'(\widehat{S_2}) \\ &\leq q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot (\log^3 n_1 \cdot \text{opt}_1 + \log^3 n_2 \cdot \text{opt}_2) + c \cdot \log^2 n' \cdot \text{opt}. \end{aligned}$$

Recall that $n_1, n_2 \leq \frac{2}{3} n'$ and $\text{opt}_1 + \text{opt}_2 \leq \text{opt}$. Thus, we have that

$$\begin{aligned} w'(T) &\leq q \cdot \log n' \cdot \text{opt} + \tfrac{d}{2} \cdot (\log^3 \tfrac{2}{3} n') \cdot \text{opt} + c \cdot \log^2 n' \cdot \text{opt} \\ &\leq \tfrac{d}{2} \cdot \log^3 n' \cdot \text{opt} + (c - d \log \tfrac{3}{2}) \cdot \log^2 n' \cdot \text{opt}. \end{aligned}$$

Overall, we conclude that to ensure that $w'(T) \leq \frac{d}{2} \cdot \log^3 n' \cdot \text{opt}$, it is sufficient to ensure that $c - d \log \frac{3}{2} \leq 0$, which can be done by fixing $d = \dfrac{c}{\log \frac{3}{2}}$.

7 CONCLUSION

In this paper, we designed $O(\log^{O(1)} n)$-approximation algorithms for Weighted Planar $\mathscr{F}$-Minor-Free Deletion, Weighted Chordal Vertex Deletion and Weighted Distance Hereditary Vertex Deletion (or Weighted Rankwidth-1 Vertex Deletion). These algorithms are the first ones for these problems whose approximation factors are bounded by $O(\log^{O(1)} n)$. Along the way, we also obtained a constant-factor approximation algorithm for Weighted Multicut on chordal graphs. All our algorithms are based on the same recursive scheme. We believe that the scope of applicability of our approach is very wide. We would like to conclude our paper with the following concrete open problems.

- Does Weighted Planar $\mathscr{F}$-Minor-Free Deletion admit a constant-factor approximation algorithm? Furthermore, studying families $\mathscr{F}$ that do not necessarily contain a planar graph is another direction for further research.
- Does Weighted Chordal Vertex Deletion admit a constant-factor approximation algorithm?
- Does Weighted Rankwidth-η Vertex Deletion admit a $O(\log^{O(1)} n)$-factor approximation algorithm?
- On which other graph classes does Weighted Multicut admits a constant-factor approximation?

ACKNOWLEDGMENTS

We sincerely thank Nikhil Bansal and Seeun William Umboh for explaining us their paper and for several discussions on WP$\mathscr{F}$-MFD.

Akanksha Agrawal is currently funded by the PBC Fellowship Program for Outstanding Post-Doctoral Researchers from China and India.

Daniel Lokshtanov is currently funded by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (no. 715744), and United States - Israel Binational Science Foundation (no. 2018302).

Saket Saurabh is currently funded by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (no. 819416), and Swarnajayanti Fellowship (no. DST/SJF/MSA01/2017-18).

Meirav Zehavi is currently funded by Israel Science Foundation grant no. 1176/18, and United States - Israel Binational Science Foundation (no. 2018302).

This research has received funding from the European Research Council under ERC grants (no. 306992 and no. 715744).

REFERENCES

[1] Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. 2017. Feedback Vertex Set Inspired Kernel for Chordal Vertex Deletion. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Barcelona, 1383–1398. https://doi.org/10.1137/1.9781611974782.90

[2] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. 1999. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics* 12, 3 (1999), 289–297.

[3] Nikhil Bansal, Daniel Reichman, and Seeun William Umboh. 2017. LP-based Robust Algorithms for Noisy Minor-free and Bounded Treewidth Graphs. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Barcelona, 1964–1979. https://doi.org/10.1137/1.9781611974782.128

[4] Reuven Bar-Yehuda and Shimon Even. 1981. A Linear-Time Approximation Algorithm for the Weighted Vertex Cover Problem. *Journal of Algorithms* 2, 2 (1981), 198–203.

[5] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. 1998. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM J. Comput.* 27, 4 (1998), 942–959.

[6] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. 1992. Automatic Generation of Linear-Time Algorithms from Predicate Calculus Descriptions of Problems on Recursively Constructed Graph Families. *Algorithmica* 7, 5&6 (1992), 555–581.

[7] Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. 2006. On the Hardness of Approximating Multicut and Sparsest-Cut. *Computational Complexity* 15, 2 (2006), 94–114.

[8] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. 2000. Linear Time Solvable Optimization Problems on Graphs of Bounded Clique-Width. *Theory of Computing Systems* 33, 2 (2000), 125–150.

[9] Bruno Courcelle and Stephan Olariu. 2000. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics* 101, 1-3 (2000), 77–114.

[10] Reinhard Diestel. 2012. *Graph Theory, 4th Edition.* Graduate texts in mathematics, Vol. 173. Springer, Berlin.

[11] M Farber. 1989. On diameters and radii of bridged graphs. *Discrete Mathematics* 73 (1989), 249–260.

[12] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. 2008. Improved Approximation Algorithms for Minimum Weight Vertex Separators. *SIAM J. Comput.* 38, 2 (2008), 629–657.

[13] Samuel Fiorini, Gwenaël Joret, and Ugo Pietropaoli. 2010. Hitting Diamonds and Growing Cacti. In *Integer Programming and Combinatorial Optimization, 14th International Conference (IPCO) (Lecture Notes in Computer Science)*, Vol. 6080. Springer, Lausanne, 191–204.

[14] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. 2016. Hitting Forbidden Minors: Approximation and Kernelization. *SIAM Journal on Discrete Mathematics* 30, 1 (2016), 383–410.

[15] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. 2012. Planar $\mathcal{F}$-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *Proceedings of IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, New Brunswick, 470–479. https://doi.org/10.1109/FOCS.2012.62

[16] Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. 2011. Bidimensionality and EPTAS. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, San Francisco, 748–759. https://doi.org/10.1137/1.9781611973082.59

[17] Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. 2012. Bidimensionality and geometric graphs. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Kyoto, 1563–1575. https://doi.org/10.1137/1.9781611973099.124

[18] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. 2010. Bidimensionality and Kernels. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Texas, 503–510. https://doi.org/10.1137/1.9781611973075.43

[19] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. 1996. Approximate Max-Flow Min-(Multi)Cut Theorems and Their Applications. *SIAM J. Comput.* 25, 2 (1996), 235–251.

[20] Bernd Gärtner and Jivr'i Matouvsek. 2007. *Understanding and using linear programming.* Springer, Berlin.

[21] Daniel Golovin, Viswanath Nagarajan, and Mohit Singh. 2006. Approximating the k-multicut problem. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* ACM Press, Philadelphia, 621–630.

[22] Martin Charles Golumbic. 1980. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York.

[23] Anupam Gupta, Euiwoong Lee, Jason Li, Pasin Manurangsi, and Michal Wlodarczyk. 2019. Losing Treewidth by Separating Subsets. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* SIAM, San Diego, 1731–1749. https://doi.org/10.1137/1.9781611975482.104

[24] Peter L Hammer and Frédéric Maffray. 1990. Completely separable graphs. *Discrete applied mathematics* 27, 1 (1990), 85–99.

[25] Petr Hlinený, Sang-il Oum, Detlef Seese, and Georg Gottlob. 2008. Width Parameters Beyond Tree-width and their Applications. *Comput. J.* 51, 3 (2008), 326–362.

[26] Edward Howorka. 1977. A characterization of distance-hereditary graphs. *The quarterly journal of mathematics* 28, 4 (1977), 417–420.

[27] Bart M. P. Jansen and Marcin Pilipczuk. 2017. Approximation and Kernelization for Chordal Vertex Deletion. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA).* SIAM, Barcelona, 1399–1418. https://doi.org/10.1137/1.9781611974782.91

[28] Bart M. P. Jansen and Marcin Pilipczuk. 2018. Approximation and Kernelization for Chordal Vertex Deletion. *SIAM Journal on Discrete Mathematics* 32, 3 (2018), 2258–2301.

[29] Leonid G Khachiyan. 1980. Polynomial algorithms in linear programming. *U. S. S. R. Comput. Math. and Math. Phys.* 20, 1 (1980), 53–72.

[30] Subhash Khot. 2002. On the power of unique 2-prover 1-round games. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC).* ACM, Montréal, 767–775. https://doi.org/10.1145/509907.510017

[31] Eun Jung Kim and O-joung Kwon. 2018. Erdős-Pósa property of chordless cycles and its applications. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* SIAM, New Orleans, 1665–1684. https://doi.org/10.1137/1.9781611975031.109

[32] Eun Jung Kim and O-Joung Kwon. 2017. A Polynomial Kernel for Distance-Hereditary Vertex Deletion. In *Algorithms and Data Structures.* Springer, Cham, 509–520. https://doi.org/10.1007/978-3-319-62127-2_43

[33] Jon M. Kleinberg and Éva Tardos. 2005. *Algorithm design.* Addison-Wesley, Boston.

[34] T Leighton and S Rao. 1999. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM* 46 (1999), 787—-832.

[35] John M. Lewis and Mihalis Yannakakis. 1980. The Node-Deletion Problem for Hereditary Properties is NP-Complete. *J. Comput. System Sci.* 20, 2 (1980), 219–230.

[36] Carsten Lund and Mihalis Yannakakis. 1993. The Approximation of Maximum Subgraph Problems. In *Proceedings of the 20nd International Colloquium on Automata, Languages and Programming (ICALP)*, Vol. 700. Springer, Berlin, 40–51. https://doi.org/10.1007/3-540-56939-1_60

[37] John W Moon and Leo Moser. 1965. On cliques in graphs. *Israel journal of Mathematics* 3, 1 (1965), 23–28.

[38] G. L. Nemhauser and L. E. Trotter, Jr. 1974. Properties of vertex packing and independence system polyhedra. *Mathematical Programming* 6 (1974), 48–61.

[39] Sang-il Oum. 2005. Rank-width and vertex-minors. *Journal of Combinatorial Theory, Series B* 95, 1 (2005), 79–100.

[40] Sang-il Oum. 2008. Approximating rank-width and clique-width quickly. *ACM Transactions on Algorithms* 5, 1 (2008), 10:1–10:20.

[41] Sang-il Oum. 2017. Rank-width: Algorithmic and structural results. *Discrete Applied Mathematics* 231 (2017), 15–24.

[42] Sang-il Oum and Paul D. Seymour. 2006. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B* 96, 4 (2006), 514–528.

[43] Neil Robertson and P D Seymour. 1986. Graph Minors. V. Excluding a Planar Graph. *Journal of Combinatorial Theory Series B* 41, 1 (1986), 92–114.

[44] Neil Robertson and Paul D. Seymour. 1995. Graph Minors .XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B* 63, 1 (1995), 65–110.

[45] Neil Robertson and Paul D. Seymour. 2004. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B* 92, 2 (2004), 325–357.

[46] Horst Sachs. 1970. On the Berge conjecture concerning perfect graphs. *Combinatorial Structures and their Applications* 37 (1970), 384.

[47] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. 1994. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discrete Mathematics* 127, 1-3 (1994), 293–304.

[48] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. 1977. A New Algorithm for Generating All the Maximal Independent Sets. *SIAM J. Comput.* 6, 3 (1977), 505–517.

[49] Mihalis Yannakakis. 1979. The Effect of a Connectivity Requirement on the Complexity of Maximum Subgraph Problems. *J. ACM* 26, 4 (1979), 618–630.

[50] Mihalis Yannakakis. 1994. Some Open Problems in Approximation. In *Proceedings of 2nd Italian Conference on Algorithms and Complexity, Second (CIAC)*. Springer, Berlin, 33–39. https://doi.org/10.1007/3-540-57811-0_4