5-Approximation for $\mathcal{H}$-Treewidth Essentially as Fast as $\mathcal{H}$-Deletion Parameterized by Solution Size

Bart M. P. Jansen
Eindhoven University of Technology, The Netherlands

Jari J. H. de Kroon
Eindhoven University of Technology, The Netherlands

Michał Włodarczyk
University of Warsaw, Poland

Abstract

The notion of $\mathcal{H}$-treewidth, where $\mathcal{H}$ is a hereditary graph class, was recently introduced as a generalization of the treewidth of an undirected graph. Roughly speaking, a graph of $\mathcal{H}$-treewidth at most k can be decomposed into (arbitrarily large) $\mathcal{H}$-subgraphs which interact only through vertex sets of size $\mathcal{O}(k)$ which can be organized in a tree-like fashion. $\mathcal{H}$-treewidth can be used as a hybrid parameterization to develop fixed-parameter tractable algorithms for $\mathcal{H}$-DELETION problems, which ask to find a minimum vertex set whose removal from a given graph G turns it into a member of $\mathcal{H}$. The bottleneck in the current parameterized algorithms lies in the computation of suitable tree $\mathcal{H}$-decompositions.

We present FPT-approximation algorithms to compute tree $\mathcal{H}$-decompositions for hereditary and union-closed graph classes $\mathcal{H}$. Given a graph of $\mathcal{H}$-treewidth k, we can compute a 5-approximate tree $\mathcal{H}$-decomposition in time $f(\mathcal{O}(k)) \cdot n^{\mathcal{O}(1)}$ whenever $\mathcal{H}$-DELETION parameterized by solution size can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function $f(k) \geq 2^k$. The current-best algorithms either achieve an approximation factor of $k^{\mathcal{O}(1)}$ or construct optimal decompositions while suffering from non-uniformity with unknown parameter dependence. Using these decompositions, we obtain algorithms solving ODD CYCLE TRANSVERSAL in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ parameterized by bipartite-treewidth and VERTEX PLANARIZATION in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ parameterized by planar-treewidth, showing that these can be as fast as the solution-size parameterizations and giving the first ETH-tight algorithms for parameterizations by hybrid width measures.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases fixed-parameter tractability, treewidth, graph decompositions

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.66

Related Version *Full Version*: https://arxiv.org/abs/2306.17065 [31]

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 803421, ReduceSearch).

1 Introduction

Background and motivation. Treewidth (see [7, 19] [16, §7]) is a width measure for graphs that is ubiquitous in algorithmic graph theory. It features prominently in the Graph Minors series [47] and frequently pops up unexpectedly [38] in the parameterized complexity [16, 20, 23] analysis of NP-hard graph problems on undirected graphs. The notion of treewidth captures how tree-like a graph is in a certain sense; it is defined as the width of an optimal tree decomposition for the graph. Unfortunately, computing an optimal tree decomposition is NP-hard [3]. As many of the algorithmic applications of treewidth

require a tree decomposition to be able to work, there has been long record of algorithms computing optimal [3, 6, 9, 43] or near-optimal [4, 8, 35] tree decompositions with no end in sight [36], as well as a long series of experimental work on heuristically computing good tree decompositions [10, 11, 17, 18]. In this paper, we present a new fixed-parameter tractable approximation algorithm for the notion of $\mathcal{H}$-treewidth, a generalization of treewidth which has recently attracted significant attention [1, 21, 29, 30]. Before describing our contributions for $\mathcal{H}$-treewidth, we summarize the most important background to motivate the problem.

The popularity of treewidth as a graph parameter can be attributed to the fact that it has very good algorithmic properties (by Courcelle's theorem, any problem that can be formulated in Counting Monadic Second-Order (CMSO_2) logic can be solved in linear time on graphs of bounded treewidth [15]), while also having a very elegant mathematical structure theory. Unfortunately, simple substructures like grids or cliques in a graph can already make its treewidth large. This means that for many input graphs of interest, the treewidth is too large for an approach based on treewidth to be efficient: the running times of many treewidth-based algorithms are of the form $f(k) \cdot n^{\mathcal{O}(1)}$, where f is an exponential function in the treewidth k and n is the total number of vertices of the graph.

Several approaches have been taken to cope with the fact that treewidth is large on graphs with large cliques or large induced grid subgraphs. One approach lies in generalized width measures like cliquewidth or rankwidth [41], by essentially replacing the use of separations of small order (which are encoded in tree decompositions), by separations of large order but in which the interactions between the two sides is well-structured. Unfortunately this generality comes at a price in terms of algorithmic applications [24, 25, 26].

This has recently led Eiben, Ganian, Hamm, and Kwon [21] to enrich the notion of treewidth in a different way. Consider a hereditary class $\mathcal{H}$ of graphs, such as bipartite graphs. The notion of $\mathcal{H}$-treewidth aims to capture how well a graph G can be decomposed into subgraphs belonging to $\mathcal{H}$ which only interact with the rest of the graph via small vertex sets which are organized in a tree-like manner. While we defer formal definitions of $\mathcal{H}$-treewidth to Section 2, an intuitive way to think of the concept is the following: a graph G has $\mathcal{H}$-treewidth at most k if and only if it can be obtained from a graph G_0 with a tree decomposition of width at most k by the following process: repeatedly insert a subgraph H_i belonging to graph class $\mathcal{H}$, such that the neighbors of H_i in the rest of the graph are all contained in a single bag of the tree decomposition of G_0. The $\mathcal{H}$-subgraphs H_i inserted during this process are called *base components* and their neighborhoods have size at most $k+1$. When $\mathcal{H}$ is a graph class of unbounded treewidth, like bipartite graphs, the $\mathcal{H}$-treewidth of a graph can be arbitrarily much smaller than its treewidth. This prompted an investigation of the algorithmic applications of $\mathcal{H}$-treewidth.

In recent works [1, 21, 30], the notion of $\mathcal{H}$-treewidth was used to develop new algorithms to solve vertex-deletion problems. Many classic NP-hard problems in algorithmic graph theory can be phrased in the framework of $\mathcal{H}$-deletion: find a minimum vertex-subset S of the input graph G such that $G - S$ belongs to a prescribed graph class $\mathcal{H}$. Examples include Vertex Cover (where $\mathcal{H}$ is the class of edgeless graphs), Odd Cycle Transversal (bipartite graphs), and Vertex Planarization (planar graphs). All these problems are known to be fixed-parameter tractable [14, 33, 34, 39, 44] when parameterized by the size of a desired solution: there are algorithms that, given an n-vertex graph G and integer k, run in time $f(k) \cdot n^{\mathcal{O}(1)}$ and output a vertex set $S \subseteq V(G)$ of size at most k for which $G - S \in \mathcal{H}$, if such a set exists. These algorithms show that large instances whose optimal solutions are small, can still be solved efficiently. Alternatively, since the mentioned graph classes $\mathcal{H}$ can be defined in CMSO_2, these vertex-deletion problems can be solved in time $f(w) \cdot n$

parameterized by the treewidth w of the input graph via Courcelle's theorem, which shows that instances of small treewidth (but whose optimal solutions may be large) can be solved efficiently.

The notion of $\mathcal{H}$-treewidth (abbreviated as $\mathbf{tw}_{\mathcal{H}}$ from now on) can be used to combine the best of both worlds. It is not difficult to show that if a graph G has a vertex set S of size k for which $G - S \in \mathcal{H}$ (we call such a set an $\mathcal{H}$-deletion set), then the $\mathcal{H}$-treewidth of G is at most k: simply take a trivial tree decomposition consisting of a single bag of size k for the graph $G_0 := G[S]$, so that afterwards the graph G can be obtained from G_0 by inserting the graph $H = G - S$, which belongs to $\mathcal{H}$ and has all its neighbors in a single bag. Since the $\mathcal{H}$-treewidth of G is also never larger than its standard treewidth, the *hybrid* (cf. [2]) parameterization by $\mathcal{H}$-treewidth dominates both the parameterizations by the solution size and the treewidth of the graph. This raises the question whether existing fixed-parameter tractability results for parameterizations of $\mathcal{H}$-DELETION by treewidth or solution size, can be extended to $\mathbf{tw}_{\mathcal{H}}$.

It was recently shown [1] that when it comes to *non-uniform* fixed-parameter tractability characterizations, the answer to this question is positive. If $\mathcal{H}$ satisfies certain mild conditions, which is the case for all graph classes mentioned so far, then for each value of k there exists an algorithm $\mathcal{A}_{\mathcal{H},k}$ that, given a graph G with $\mathbf{tw}_{\mathcal{H}}(G) \leq k$ and target value t, decides whether or not G has an $\mathcal{H}$-deletion set of size at most t. There is a constant $c_{\mathcal{H}}$ such that each algorithm $\mathcal{A}_{\mathcal{H},k}$ runs in time $\mathcal{O}(n^{c_{\mathcal{H}}})$, so that the overall running time can be bounded by $f(k) \cdot n^{c_{\mathcal{H}}}$; however, no bounds on the function f are given and in general it is unknown how to construct the algorithms whose existence is proven. Another recent paper [29] gave concrete FPT algorithms to solve $\mathcal{H}$-DELETION parameterized by $\mathbf{tw}_{\mathcal{H}}$ for certain cases of $\mathcal{H}$, including the three mentioned ones. For example, it presents an algorithm that solves ODD CYCLE TRANSVERSAL in time $2^{\mathcal{O}(k^3)} \cdot n^{\mathcal{O}(1)}$, parameterized by bipartite-treewidth. The bottleneck in the latter approach lies in the computation of a suitable tree $\mathcal{H}$-decomposition: on a graph of $\mathbf{tw}_{\mathcal{H}}(G) \leq k$, the algorithm runs in $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ time to compute a tree $\mathcal{H}$-decomposition of width $w \in \mathcal{O}(k^3)$, and then optimally solves ODD CYCLE TRANSVERSAL on the decomposition of width w in time $2^{\mathcal{O}(w)} \cdot n^{\mathcal{O}(1)} \leq 2^{\mathcal{O}(k^3)} \cdot n^{\mathcal{O}(1)}$. Note that the parameter dependence of this algorithm is much worse than for the parameterizations by solution size and by treewidth, both of which can be solved in single-exponential time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ [44, 37]. To improve the running times of algorithms for $\mathcal{H}$-DELETION based on hybrid parameterizations, improved algorithms are therefore required to compute approximate tree $\mathcal{H}$-decompositions. These form the subject of our work.

Our contribution: $\mathcal{H}$-treewidth. We develop generic FPT algorithms to approximate $\mathcal{H}$-treewidth, for graph classes $\mathcal{H}$ which are hereditary and closed under taking the disjoint union of graphs. To approximate $\mathcal{H}$-treewidth, all our algorithm needs is access to an oracle for solving $\mathcal{H}$-DELETION parameterized by solution size. The values of the solution size for which the oracle is invoked, will be at most twice as large as the $\mathcal{H}$-treewidth of the graph we are decomposing. Hence existing algorithms for solution-size parameterizations of $\mathcal{H}$-DELETION can be used as a black box to form the oracle. Aside from the oracle calls, our algorithm only takes $8^k \cdot kn(n+m)$ time on an n-vertex graph with m edges. So whenever the solution-size parameterization can be solved in single-exponential time, an approximate tree $\mathcal{H}$-decomposition can be found in single-exponential time. The approximation factor of the algorithm is 5, which is a significant improvement over earlier poly(opt) approximations running in superexponential time. The formal statement of our main result is the following.

▶ **Theorem 1.** *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. There is an algorithm that, using oracle-access to an algorithm $\mathcal{A}$ for $\mathcal{H}$-DELETION, takes as input an n-vertex m-edge graph G, integer k, and either computes a tree $\mathcal{H}$-decomposition of G of width at*

most $5k+5$ consisting of $\mathcal{O}(n)$ nodes, or correctly concludes that $\boldsymbol{tw}_{\mathcal{H}}(G) > k$. The algorithm runs in time $\mathcal{O}(8^k \cdot kn(n+m))$, polynomial space, and makes $\mathcal{O}(8^k n)$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter $2k+2$.

Theorem 1 yields the first constant-factor approximation algorithms for $\mathbf{tw}_{\mathcal{H}}$ that run in single-exponential time. For example, for $\mathcal{H}$ the class of bipartite graphs the running time becomes $\mathcal{O}(72^k \cdot n^2(n+m))$, and for interval graphs we obtain $\mathcal{O}(8^{3k} \cdot n(n+m))$ (the full version [31] gives results for more classes $\mathcal{H}$). Combining these approximate decompositions with existing algorithms that solve $\mathcal{H}$-DELETION on a given tree $\mathcal{H}$-decomposition, we obtain ETH-tight algorithms as a consequence. ODD CYCLE TRANSVERSAL can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, and VERTEX PLANARIZATION can be solved in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ when parameterized by $\mathbf{tw}_{\mathcal{H}}$ for $\mathcal{H}$ the class of bipartite and planar graphs, respectively, without having to supply a decomposition in the input. For VERTEX PLANARIZATION, the previous-best bound [32] was $2^{\mathcal{O}(k^5 \log k)} \cdot n^{\mathcal{O}(1)}$. Note that for the planarization problem, a parameter dependence of $2^{o(k \log k)}$ is impossible assuming the Exponential Time Hypothesis; this already holds for the larger parameterization by treewidth [42]. For ODD CYCLE TRANSVERSAL, an algorithm running in time $2^{o(n)}$ would violate the Exponential Time Hypothesis, which follows by a simple reduction from VERTEX COVER for which such a lower bound is known [16, Theorem 14.6]. This implies that the solution size parameterization cannot be solved in subexponential time.

Compared to existing algorithms to approximate treewidth, the main obstacle we have to overcome in Theorem 1 is identifying the base components of an approximate decomposition in a suitable way. The earlier FPT $\mathsf{poly}(\mathsf{opt})$-approximation for $\mathbf{tw}_{\mathcal{H}}$ effectively reduced the input graph G to a graph G' by repeatedly extracting large $\mathcal{H}$-subgraphs with small neighborhoods, in such a way that the treewidth of G' can be bounded in terms of $\mathbf{tw}_{\mathcal{H}}(G)$, while a tree decomposition of G' can be lifted into an approximate tree $\mathcal{H}$-decomposition of G. Several steps in this process led to losses in the approximation factor. To obtain our 5-approximation, we avoid the translation between G and G', and work directly on decomposing the input graph G.

Our recursive decomposition algorithm works similarly as the Robertson-Seymour 4-approximation algorithm for treewidth [45] (cf. [16, §7.6]). When given a graph G and integer k with $\mathbf{tw}_{\mathcal{H}}(G) \leq k$, the algorithm maintains a vertex set S of size $3k+4$ which forms the boundary between the part of the graph that has already been decomposed and the part that still needs to be processed. If S has a $\frac{2}{3}$-balanced separator R of size $k+1$, we can proceed in the usual way: we split the graph based on R, recursively decompose the resulting parts, and combine these decompositions by adding a bag containing $R \cup S$ as the root. If S does not have a balanced separator of size $k+1$, then we show (modulo some technical details) that for any optimal tree $\mathcal{H}$-decomposition, there is a subset $S' \subseteq S$ of $2k+3$ vertices which belong to a single base component H_0. Our main insight is that such a set S' can be used in a win/win approach, by maintaining an $\mathcal{H}$-deletion set X during the decomposition process that initially contains all vertices. To make progress in the recursion, we would like to split off a base component containing S' via a separator U of size at most $2k+2$, while adding U to the boundary of the remainder of the graph to be decomposed. To identify an induced $\mathcal{H}$-subgraph with small neighborhood that can serve as a base component, we compute a minimum (S', X)-separator U (we allow U to intersect the sets S', X). Any connected component H of $G - U$ that contains a vertex from S' does not contain any vertex of the $\mathcal{H}$-deletion set X, so H is an induced subgraph of $G - X$ which implies $H \in \mathcal{H}$ for hereditary $\mathcal{H}$. Hence if there is an (S', X)-separator U of size at most $2k+2$, we can use it to split off base components neighboring U that eliminate $2k+3$ vertices from S from the boundary, thereby making room to insert U into the boundary without blowing up its size.

Of course, it may be that all (S', X)-separators are larger than $2k + 2$; by Menger's theorem, this happens exactly when there is a family $\mathcal{P}$ of $2k+3$ vertex-disjoint (S', X)-paths. Only $k + 1$ paths in $\mathcal{P}$ can *escape* the base component H_0 covering S' since its neighborhood has size at most $k + 1$, so that $k + 2$ of them end in a vertex of the deletion set X that lies in H_0. The key point is now that this situation implies that X is redundant in a technical sense: if we let X' denote the endpoints of $k + 2$ (S', X)-paths starting and ending in H_0, we can obtain a smaller $\mathcal{H}$-deletion set by replacing X' by the neighborhood of H_0, which has size at most $k + 1$. This replacement is valid as long as $\mathcal{H}$ is hereditary and union-closed. Using an oracle for $\mathcal{H}$-DELETION parameterized by solution size, we can therefore efficiently find a smaller $\mathcal{H}$-deletion set when we know X'. While the algorithm does not know X' in general, this type of argument leads to the win/win: either there is a small (S', X)-separator which we can use to split off a base component, or there is a large family of vertex-disjoint (S', X)-paths which allows the $\mathcal{H}$-deletion set to be improved. As the latter can only happen $|V(G)|$ times, we must eventually identify a base component to split off, allowing the recursion to proceed.

Our contribution: $\mathcal{H}$-elimination distance. The $\mathcal{H}$-elimination distance $\mathbf{ed}_{\mathcal{H}}(G)$ of a graph G is a parameter [12, 13] that extends treedepth [40] similarly to how $\mathcal{H}$-treewidth extends treewidth. For hereditary and union-closed classes $\mathcal{H}$, the $\mathcal{H}$-elimination distance of a graph G is the minimum number of rounds needed to turn G into a member of $\mathcal{H}$, when a round consists of removing one vertex from each connected component. Such an elimination process can be represented by a tree structure called *$\mathcal{H}$-elimination forest*. Aside from the fact that computing the $\mathcal{H}$-elimination distance may reveal interesting properties of a graph G, a second motivation for studying this parameter is that it can facilitate *polynomial-space* algorithms for solving $\mathcal{H}$-DELETION, while the parameterization by $\mathbf{tw}_{\mathcal{H}}$ (which is never larger) typically gives rise to exponential-space algorithms. At a high level, the state of the art for computing $\mathbf{ed}_{\mathcal{H}}$ is similar as for $\mathbf{tw}_{\mathcal{H}}$: there is an exact non-uniform FPT algorithm with unspecified parameter dependence that works as long as $\mathcal{H}$ satisfies some mild conditions [1], while uniform $\mathsf{poly}(\mathsf{opt})$-approximation algorithms running in time $2^{k^{\mathcal{O}(1)}} \cdot n^{\mathcal{O}(1)}$ are known for several concrete graph classes $\mathcal{H}$ [30].

By leveraging similar ideas as for Theorem 1, we also obtain improved FPT-approximation algorithms for $\mathbf{ed}_{\mathcal{H}}$. The following theorem gives algorithms for two settings: one for an algorithm using polynomial space whenever the algorithm $\mathcal{A}$ for $\mathcal{H}$-DELETION does, which is the case for most of the considered graph classes, and one for an exponential-space algorithm with a better approximation ratio.

▶ **Theorem 2.** *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. There exists an algorithm that, using oracle-access to an algorithm $\mathcal{A}$ for $\mathcal{H}$-DELETION, takes as input an n-vertex graph G and integer k, runs in time $n^{\mathcal{O}(1)}$, makes $n^{\mathcal{O}(1)}$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter $2k$, and either concludes that $\mathbf{ed}_{\mathcal{H}}(G) > k$ or outputs an $\mathcal{H}$-elimination forest of depth $\mathcal{O}(k^3 \log^{3/2} k)$.*

Under the same assumptions, there is an algorithm that runs in time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$, makes $n^{\mathcal{O}(1)}$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter $2k$, and either concludes that $\mathbf{ed}_{\mathcal{H}}(G) > k$ or outputs an $\mathcal{H}$-elimination forest of depth $\mathcal{O}(k^2)$.

In the previous work [30, 32] such a dependence on k was possible only in two cases: when $\mathcal{H}$ is the class of bipartite graphs or when $\mathcal{H}$ is defined by a finite family of forbidden induced subgraphs. In the general case, our result effectively shaves off a single k-factor in the depth of the returned decomposition and in the exponent of the running time, compared to the previously known approximations. Theorem 2 entails better approximation algorithms for $\mathcal{H}$-elimination distance for classes of e.g. chordal, interval, planar, bipartite permutation, or distance-hereditary graphs.

Organization. The remainder of the paper is organized as follows. We continue by presenting formal preliminaries in Section 2. In Section 3 we treat $\mathcal{H}$-treewidth, developing the theory and subroutines needed to prove Theorem 1. Due to space limitations, the resulting proof of Theorem 1 is deferred to the full version [31], which also provides a list of applications for concrete graph classes. The proof of Theorem 2 can also be found in the full version. We conclude in Section 4.

2 Preliminaries

Graphs and graph classes. We consider finite, simple, undirected graphs. We denote the vertex and edge sets of a graph G by $V(G)$ and $E(G)$ respectively, with $|V(G)| = n$ and $|E(G)| = m$. For a set of vertices $S \subseteq V(G)$, by $G[S]$ we denote the graph induced by S. We use shorthand $G - v$ and $G - S$ for $G[V(G) \setminus \{v\}]$ and $G[V(G) \setminus S]$, respectively. The open neighborhood $N_G(v)$ of $v \in V(G)$ is defined as $\{u \in V(G) \mid uv \in E(G)\}$. The closed neighborhood of v is $N_G[v] = N_G(v) \cup \{v\}$. For $S \subseteq V(G)$, we have $N_G[S] = \bigcup_{v \in S} N_G[v]$ and $N_G(S) = N_G[S] \setminus S$. We define the boundary $\partial_G(S)$ of the vertex set S as $N_G(V(G) \setminus S)$, i.e., those vertices of S which have a neighbor outside S.

A class of graphs $\mathcal{H}$ is called *hereditary* if for any $G \in \mathcal{H}$, every induced subgraph of G also belongs to $\mathcal{H}$. Furthermore, $\mathcal{H}$ is *union-closed* if for any $G_1, G_2 \in \mathcal{H}$ the disjoint union of G_1 and G_2 also belongs to $\mathcal{H}$. For a graph class $\mathcal{H}$ and a graph G, a set $X \subseteq V(G)$ is called an $\mathcal{H}$-deletion set in G if $G - X \in \mathcal{H}$. For a graph class $\mathcal{H}$, the parameterized problem $\mathcal{H}$-DELETION takes a graph G and parameter k as input, and either outputs a minimum-size $\mathcal{H}$-deletion set in G or reports that there is no such set of size at most k.

Separators. For two (not necessarily disjoint) sets $X, Y \subseteq V(G)$ in a graph G, a set $P \subseteq V(G)$ is an (X, Y)-separator if no connected component of $G - P$ contains a vertex from both $X \setminus P$ and $Y \setminus P$. Such a separator may intersect $X \cup Y$. Equivalently, P is an (X, Y)-separator if each (X, Y)-path contains a vertex of P. The minimum cardinality of such a separator is denoted $\lambda_G(X, Y)$. By Menger's theorem, $\lambda_G(X, Y)$ is equal to the maximum cardinality of a set of pairwise vertex-disjoint (X, Y)-paths. A pair (A, B) of subsets of $V(G)$ is a *separation* in G if $A \cup B = V(G)$ and G has no edges between $A \setminus B$ and $B \setminus A$. Its order is defined as $|A \cap B|$.

▶ **Observation 3.** *For two sets $X, Y \subseteq V(G)$, it holds that $\lambda_G(X, Y) \leq k$ if and only if there exists a separation (A, B) in $V(G)$ such that $X \subseteq A$, $Y \subseteq B$, and $|A \cap B| \leq k$.*

The following theorem summarizes how, given vertex sets $X, Y \subseteq V(G)$ and a bound k, we can algorithmically find a small-order separation or a large system of vertex-disjoint paths. The statement follows from the analysis of the Ford-Fulkerson algorithm for maximum (X, Y)-flow in which each vertex has a capacity of 1. If the algorithm has not terminated within k iterations, then the flow of value $k + 1$ yields $k + 1$ vertex-disjoint paths. If it terminates earlier, a suitable separation can be identified based on reachability in the residual network of the last iteration.

▶ **Theorem 4** (Ford-Fulkerson, see [16, Thm. 8.2] and [49, §9.2])**.** *There is an algorithm that, given an n-vertex m-edge graph G, sets $X, Y \subseteq V(G)$, and integer k, runs in time $\mathcal{O}(k(n + m))$ and determines whether $\lambda_G(X, Y) \leq k$. If so, the algorithm also returns a separation (A, B) in G with $X \subseteq A$, $Y \subseteq B$, and $|A \cap B| \leq k$. Otherwise, the algorithm returns a family of $k + 1$ vertex-disjoint (X, Y)-paths.*

$\mathcal{H}$-treewidth. We continue by giving a formal definition of a tree $\mathcal{H}$-decomposition.

▶ **Definition 5.** *For a graph class $\mathcal{H}$, a tree $\mathcal{H}$-decomposition of graph G is a triple (T, χ, L) where $L \subseteq V(G)$, T is a rooted tree, and $\chi \colon V(T) \to 2^{V(G)}$, such that:*

1. *For each $v \in V(G)$ the nodes $\{t \mid v \in \chi(t)\}$ form a non-empty connected subtree of T.*
2. *For each edge $uv \in E(G)$ there is a node $t \in V(T)$ with $\{u, v\} \subseteq \chi(t)$.*
3. *For each vertex $v \in L$, there is a unique $t \in V(T)$ with $v \in \chi(t)$, and t is a leaf of T.*
4. *For each node $t \in V(T)$, the graph $G[\chi(t) \cap L]$ belongs to $\mathcal{H}$.*

The width *of a tree $\mathcal{H}$-decomposition is defined as $\max(0, \max_{t \in V(T)} |\chi(t) \setminus L| - 1)$. The $\mathcal{H}$-treewidth of a graph G, denoted $\boldsymbol{tw}_{\mathcal{H}}(G)$, is the minimum width of a tree $\mathcal{H}$-decomposition of G. The connected components of $G[L]$ are called base components.*

A pair (T, χ) is a (standard) tree decomposition *if $(T, \chi, \emptyset)$ satisfies all conditions of an $\mathcal{H}$-decomposition; the choice of $\mathcal{H}$ is irrelevant.*

For a rooted tree decomposition (T, χ), T_t denotes the subtree of T rooted at $t \in V(T)$, while $\chi(T_t) = \bigcup_{x \in V(T_t)} \chi(x)$. Similarly as treewidth, $\mathcal{H}$-treewidth is a monotone parameter with respect to taking induced subgraphs.

▶ **Observation 6.** *Let $\mathcal{H}$ be a hereditary class of graphs, G be a graph, and H be an induced subgraph of G. Then $\boldsymbol{tw}_{\mathcal{H}}(H) \leq \boldsymbol{tw}_{\mathcal{H}}(G)$.*

3 Approximating $\mathcal{H}$-treewidth

We make preparations for the proof of Theorem 1. First, we formalize the concept of a potential base component using the notion of an $(\mathcal{H}, \ell)$-separation and relate it to *redundant* subsets in a solution to $\mathcal{H}$-DELETION. Next, we prove a counterpart of the balanced-separation property for graphs of bounded $\mathcal{H}$-treewidth and explain how it allows us to apply a win/win approach in a single step of the decomposition algorithm.

3.1 Redundancy and $(\mathcal{H}, \ell)$-separations

We summon the following concept from the previous work on $\mathcal{H}$-treewidth [30] to capture $\mathcal{H}$-subgraphs with small neighborhoods.

▶ **Definition 7.** *For disjoint $C, S \subseteq V(G)$, the pair (C, S) is called an $(\mathcal{H}, \ell)$-separation in G if (1) $G[C] \in \mathcal{H}$, (2) $|S| \leq \ell$, and (3) $N_G(C) \subseteq S$.*

This notion is tightly connected to the base components of tree $\mathcal{H}$-decompositions. For any tree $\mathcal{H}$-decomposition (T, χ, L) of width k of a graph G, for any node $t \in T$, the graph $G[\chi(t) \cap L]$ belongs to $\mathcal{H}$ so that $C := \chi(t) \cap L$ satisfies Definition 7. The open neighborhood of $\chi(t) \cap L$ is a subset of $S := \chi(t) \setminus L$, which follows from the fact that vertices of L only occur in a single bag, while each edge has both endpoints covered by a single bag. Since $|S| \leq k + 1$ by definition of the width of a tree $\mathcal{H}$-decomposition, this leads to the following observation.

▶ **Observation 8.** *Let (T, χ, L) be a tree $\mathcal{H}$-decomposition of a graph G of width k. For each node $t \in V(T)$, the pair $(\chi(t) \cap L, \chi(t) \setminus L)$ is an $(\mathcal{H}, k+1)$-separation in G.*

The following concept will be useful when working with $(\mathcal{H}, \ell)$-separations.

▶ **Definition 9.** *For an $(\mathcal{H}, k)$-separation (C, S) and set $Z \subseteq V(G)$, we say that (C, S) covers Z if $Z \subseteq C$, or weakly covers Z if $Z \subseteq C \cup S$. Set $Z \subseteq V(G)$ is called (weakly) $(\mathcal{H}, \ell)$-separable if there exists an $(\mathcal{H}, \ell)$-separation that (weakly) covers Z.*

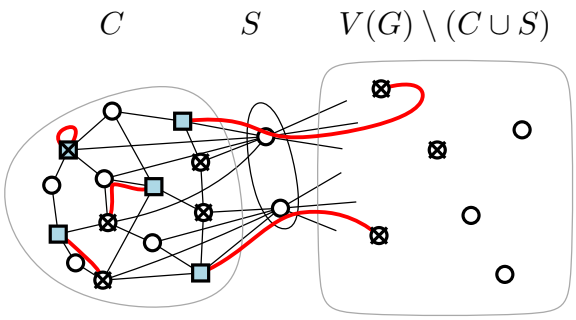

Figure 1 Illustration for Lemma 12: an $(\mathcal{H}, \ell)$-separation (C, S) in a graph G where $\mathcal{H}$ is the class of triangle-free graphs and $\ell = 2$. Vertices marked with a cross form an $\mathcal{H}$-deletion set X in G, while the set Z of size $2\ell + 1 = 5$ marked with blue squares is weakly $(\mathcal{H}, \ell)$-separable. A set of $2\ell+1$ vertex-disjoint (Z, X)-paths $\mathcal{P}$ is highlighted, witnessing $\lambda_G(Z, X) = |Z|$. Since $|X \cap C| > \ell$, the set X is not a minimum $\mathcal{H}$-deletion set in G: it can be improved by replacing $X \cap C$ with S. When (C, S) weakly covering Z exists but is unknown to the algorithm, we can still improve X since the set of X-endpoints of $\mathcal{P}$ also form a redundant set in X.

We introduce both notions to keep consistency with the earlier work [30] but in fact we will be interested only in weak coverings. Following the example above, the set $Z = \chi(t)$ is weakly $(\mathcal{H}, k + 1)$-separable but not necessarily $(\mathcal{H}, k + 1)$-separable.

Next, we introduce the notion of redundancy for solutions to $\mathcal{H}$-DELETION.

▶ **Definition 10.** *For an $\mathcal{H}$-deletion set X in G we say that a subset $X' \subseteq X$ is* redundant *in X if there exists a set $X'' \subseteq V(G)$ smaller than $|X'|$ such that $(X \setminus X') \cup X''$ is also an $\mathcal{H}$-deletion set in G.*

We remark that redundancy has been studied in the context of local-search strategies (cf. [27, 28]). It is known that for VERTEX COVER finding a redundant subset X' in a solution X is FPT in graphs of bounded local treewidth but W[1]-hard in general, when parameterized by the size of $|X'|$ [22]. However, when X' is given, one can easily check whether it is redundant using an algorithm for $\mathcal{H}$-DELETION parameterized by the solution size, due to the following observation.

▶ **Observation 11.** *Let X be an $\mathcal{H}$-deletion set in a graph G. A subset $X' \subseteq X$ is redundant in X if and only if the graph $G - (X \setminus X')$ has an $\mathcal{H}$-deletion set smaller than $|X'|$.*

An important observation is that when $X' \subseteq X$ of size at least $\ell + 1$ is weakly $(\mathcal{H}, \ell)$-separable, then it is redundant in X by a simple exchange argument. This fact has been already leveraged in previous work [1, 30] when analyzing the structure of minimum-size $\mathcal{H}$-deletion sets, which clearly cannot contain any redundant subsets. We exploit it in a different context, to prove that if there is a large flow between an $\mathcal{H}$-deletion set X and a weakly $(\mathcal{H}, \ell)$-separable set Z, then X has a redundant subset. Subsequently, we will show that this redundant subset can efficiently be detected.

▶ **Lemma 12.** *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. Consider a graph G, an $\mathcal{H}$-deletion set X in G, and a weakly $(\mathcal{H}, \ell)$-separable set $Z \subseteq V(G)$. Suppose that there exists a subset $X' \subseteq X$ of size $2\ell + 1$ such that $\lambda_G(Z, X') = 2\ell + 1$. Then X' is redundant in X.*

Proof. Let (C, S) be an $(\mathcal{H}, \ell)$ separation in G with $Z \subseteq C \cup S$. From the definition of an $(\mathcal{H}, \ell)$-separation, we have $G[C] \in \mathcal{H}$ while $N_G(C) \subseteq S$ and $|S| \leq \ell$.

By Menger's theorem, the cardinality of a maximum packing of vertex-disjoint (Z, X')-paths equals $\lambda_G(Z, X')$. Hence there exists a family $\mathcal{P} = \{P_1, \ldots, P_{2\ell+1}\}$ of vertex-disjoint paths, each of which connects a unique vertex $z_i \in Z$ to a unique vertex $x_i \in X'$ (possibly $x_i = z_i$). At most $|S|$ of these paths intersect the separator S of the $(\mathcal{H}, \ell)$-separation (see Figure 1). Let $X'' = \{x_i \mid P_i \cap S = \emptyset\}$ denote the X'-endpoints of those paths not intersecting S, and let $\mathcal{P}'$ be the corresponding paths. Each path P_i in $\mathcal{P}'$ is disjoint from S and has an endpoint $z_i \in Z$. Since $Z \subseteq C \cup S$, the z_i endpoint belongs to C. As $N_G(C) \subseteq S$ and P_i does not intersect S, the other endpoint x_i also belongs to C. Hence all vertices of X'' belong to C, and there are at least $2\ell + 1 - \ell = \ell + 1$ of them.

Let $X^* := (X \setminus X'') \cup S$, and observe that $|X^*| < |X|$ since $|X''| \geq \ell + 1$ while $|S| \leq \ell$. We prove that $G - X^* \in \mathcal{H}$, by showing that S is an $\mathcal{H}$-deletion set in $G - (X \setminus X'')$. Since $\mathcal{H}$ is union-closed, it suffices to argue that each connected component H of $G - ((X \setminus X'') \cup S)$ belongs to $\mathcal{H}$. If H contains no vertex of X'', then H is an induced subgraph of $G - X \in \mathcal{H}$ and therefore $H \in \mathcal{H}$ since the graph class is hereditary. If H contains a vertex of $X'' \subseteq C$, then the component H is an induced subgraph of $G[C]$ since $N_G(C) \subseteq S$ is part of the set X^*. Hence H is an induced subgraph of $G[C] \in \mathcal{H}$, which implies $H \in \mathcal{H}$ as $\mathcal{H}$ is hereditary. This shows that X^* is indeed an $\mathcal{H}$-deletion set.

Since $(X \setminus X'') \cup S$ is an $\mathcal{H}$-deletion set smaller than X, the set X'' is redundant in X. As $X' \supseteq X''$, it follows that X' is redundant as well. ◀

3.2 The win/win strategy

The classic 4-approximation algorithm for computing a (standard) tree decomposition is based on the existence of balanced separators in graphs of bounded treewidth. In a graph G of treewidth $\leq k$, any set S of $3k + 4$ vertices can by partitioned into $S = S_A \cup S_B$ in such a way that $|S_A|, |S_B| \leq 2k + 2$ and $\lambda_G(S_A, S_B) \leq k + 1$ [16, Corollary 7.21]. This is not always possible if we only have a bound on $\mathcal{H}$-treewidth $\mathbf{tw}_{\mathcal{H}}(G) \leq k$ because a large subset S' of S might lie in a single well-connected base component of a tree $\mathcal{H}$-decomposition, i.e., a base component whose standard treewidth is large. But then S' is weakly $(\mathcal{H}, k + 1)$-separable, which can also be exploited when constructing a decomposition. We show that this is in fact the only scenario in which we cannot split S in a balanced way.

▶ **Lemma 13.** *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. Let G be a graph with $\boldsymbol{tw}_{\mathcal{H}}(G) \leq k$. For any set $S \subseteq V(G)$ of size $3k + 4$, at least one of the following holds.*

1. *There is a partition $S = S_A \cup S_B$ such that $|S_A|, |S_B| \leq 2k + 2$ and $\lambda_G(S_A, S_B) \leq k + 1$.*
2. *There is a set $S' \subseteq S$ of size $2k + 3$ which is weakly $(\mathcal{H}, k + 1)$-separable.*

Proof. Consider an optimal tree $\mathcal{H}$-decomposition (T, χ, L) of G, so that $|\chi(t) \setminus L| \leq k + 1$ for each $t \in V(T)$. Let $r \in V(T)$ be its root. We start by showing that (2) holds if some leaf bag of the decomposition contains $2k + 3$ vertices from S.

So suppose there exists a leaf $t \in V(T)$ with $|\chi(t) \cap S| \geq 2k + 3$, and let $S' \subseteq \chi(t) \cap S$ be an arbitrary subset of size exactly $2k + 3$. Observation 8 ensures that $(C^* := \chi(t) \cap L, S^* := \chi(t) \setminus L)$ is an $(\mathcal{H}, k + 1)$-separation, which weakly covers $\chi(t)$ and therefore S'. Hence (2) holds.

In the remainder, it suffices to show that (1) holds when there is no leaf $t \in V(T)$ with $|\chi(t) \cap S| \geq 2k + 3$. Pick a deepest node t^* in the rooted tree T for which $|S \cap \chi(T_{t^*})| \geq 2k + 3$. Then t^* is not a leaf since the previous case did not apply, so by definition of tree $\mathcal{H}$-decomposition we have $\chi(t^*) \cap L = \emptyset$. Let $D_1, \ldots, D_p$ be the connected components of $G - \chi(t^*)$. Since the pair (T, χ) satisfies all properties of a standard tree decomposition, the bag $\chi(t^*)$ is a separator in G so that for each component D_i, there is a single tree T^i in the unrooted forest $T - t^*$ such that T^i contains all nodes whose bags contain some $v \in V(D_i)$; see for example [46, (2.3)].

The choice of t^* ensures that $|V(D_i) \cap S| < 2k+3$ for all $i \in [p]$: when vertices of D_i are contained in bags of a tree rooted at a child of t^* this follows from the fact that t^* is a deepest node for which $|S \cap \chi(T_{t^*})| \geq 2k+3$; when vertices of D_i are contained in the tree T^i of $T - t^*$ having the parent of t^*, this follows from the fact that $\chi(T_{t^*})$ contains at least $2k+3$ vertices from S, none of which appear in D_i since a vertex occurring in $\chi(T_{t^*})$ and in a bag outside T_{t^*}, is contained in $\chi(t^*)$ and therefore part of the separator $\chi(t^*)$ used to obtain the component D_i. Hence none of the vertices of $S \cap \chi(t^*)$ can appear in D_i, which means there are at most $|S| - (2k+3) \leq k+1$ vertices in $V(D_i) \cap S$.

Since $|S| = 3k+4$ and no component D_i contains at least $2k+3$ vertices from S, the components can be partitioned into two parts $\mathcal{D}_1, \mathcal{D}_2$ such that $\sum_{D_i \in \mathcal{D}_j} |V(D_i) \cap S| \leq 2k+2$ for each $j \in \{1,2\}$. If some component contains at least $k+2$ vertices from S, then that component is a part by itself, ensuring the remainder has at most $3k+4-(k+2) \leq 2k+2$ vertices from S; if no component contains at least $k+2$ vertices from S, then any inclusion-minimal subset of components having at least $k+2$ vertices from S has at most $2k+2$ of them.

Define $S'_A := \bigcup_{D_i \in \mathcal{D}_1} V(D_i) \cap S$ and $S'_B := \bigcup_{D_i \in \mathcal{D}_2} V(D_i) \cap S$, and assume without loss of generality that $|S'_A| \geq |S'_B|$. Note that $|S'_A \cup S'_B| = |S \setminus \chi(t^*)| \geq 2k+3$, so that the larger side S'_A contains at least $k+2$ vertices. To turn S'_A, S'_B into the desired partition of S, it suffices to take $S_A = S'_A$ and $S_B = S'_B \cup (\chi(t^*) \cap S) = S \setminus S_A$. It is clear that $|S_A| = |S'_A| \geq k+2$, while $|S_B| = |S| - |S_A| \geq 3k+4-(2k+2) \geq k+2$. The fact that $|S_A|, |S_B| \geq k+2$ while they partition S with $|S| = 3k+4$ implies $|S_A|, |S_B| \leq 2k+2$ as desired. Since $\chi(t^*)$ separates S'_A from S'_B, it separates S_A from S_B and we have $\lambda_G(S_A, S_B) \leq |\chi(t^*)| = |\chi(t^*) \setminus L| \leq k+1$. ◀

We can now translate the last two lemmas into an algorithmic statement, which will be used as a subroutine in the main algorithm. When $\mathbf{tw}_{\mathcal{H}}(G) \leq k$ and $S \subseteq V(G)$ is of size $3k+4$, then we can either split it in a balanced way, split off a base component, or detect a redundancy in a given $\mathcal{H}$-deletion set and reduce its size. Each of these outcomes will guarantee some progress for the task of constructing a tree $\mathcal{H}$-decomposition.

▶ **Lemma 14.** *Let $\mathcal{H}$ be a hereditary and union-closed class of graphs. There is an algorithm that, using oracle-access to an algorithm $\mathcal{A}$ for $\mathcal{H}$-DELETION, takes as input an n-vertex m-edge graph G, integer k, $\mathcal{H}$-deletion set X in G, and a set $S \subseteq V(G)$ of size $3k+4$, runs in time $\mathcal{O}(8^k \cdot k(n+m))$ and polynomial space, makes $\mathcal{O}(8^k)$ calls to $\mathcal{A}$ on induced subgraphs of G and parameter $2k+2$, and terminates with one of the following outcomes.*

1. *A partition $S = S_A \cup S_B$ and a separation (A, B) in G are returned, such that $S_A \subseteq A$, $S_B \subseteq B$, $|S_A| \leq 2k+2$, $|S_B| \leq 2k+2$, and $|A \cap B| \leq k+1$.*
2. *A subset $S' \subseteq S$ and a separation (A, B) in G are returned, such that $S' \subseteq A$, $X \subseteq B$, $|S'| = 2k+3$, and $|A \cap B| \leq 2k+2$. (This implies that $G[A \setminus B] \in \mathcal{H}$.)*
3. *An $\mathcal{H}$-deletion set X' in G is returned, that is smaller than X.*
4. *The algorithm correctly concludes that $\boldsymbol{tw}_{\mathcal{H}}(G) > k$.*

Proof. The algorithm starts by trying to reach the first outcome. For each partition $S_A \cup S_B$ of S in which both parts have at most $2k+2$ vertices, it performs at most $k+2$ iterations of the Fold-Fulkerson algorithm to test whether $\lambda_G(S_A, S_B) \leq k+1$. If so, then the algorithm outputs a corresponding separation (A, B) in G with $S_A \subseteq A$, $S_B \subseteq B$, and $|A \cap B| = \lambda_G(S_A, S_B) \leq k+1$. By Theorem 4, this can be done in time $\mathcal{O}(k(n+m))$.

Next, the algorithm attempts to reach the second outcome. For each subset $S' \subseteq S$ of size $2k+3$, it performs at most $2k+3$ iterations of the Ford-Fulkerson algorithm to test whether $\lambda_G(S', X) \leq 2k+2$. If so, the algorithm extracts a corresponding separation (A, B) with $S' \subseteq A$, $X \subseteq B$, and $|A \cap B| \leq 2k+2$, and outputs it.

If the algorithm has not terminated so far, it will reach the third or fourth outcome. It proceeds as follows.

1. For each subset $S' \subseteq S$ of size $2k+3$, we have $\lambda_G(S', X) > 2k+2$ since we could not reach the second outcome. As $|S'| = 2k+3$ this implies $\lambda_G(S', X) = 2k+3$. By Menger's theorem, there is a packing $\mathcal{P}_{S'}$ of $2k+3$ vertex-disjoint (S', X)-paths, and such a packing can be extracted from the final stage of the Ford-Fulkerson computation.
2. Let $X'_{S'} \subseteq X$ be the endpoints in the set X of the paths $\mathcal{P}_{S'}$, so that $|X'_{S'}| = |S'| = 2k+3$.
3. We invoke algorithm $\mathcal{A}$ on the graph $G - (X \setminus X'_{S'})$ and parameter value $2k+2$, to find a minimum-size $\mathcal{H}$-deletion set in $G - (X \setminus X'_{S'})$ or conclude that such a set has size more than $2k+2$. If $\mathcal{A}$ returns a solution Y of size at most $2k+2$, then $(X \setminus X'_{S'}) \cup Y$ is an $\mathcal{H}$-deletion set in G smaller than X and we return it as the third outcome.

If none of the preceding steps for any $S' \subseteq S$ of size $2k+3$ caused the algorithm to give an output, then we conclude that $\mathbf{tw}_{\mathcal{H}}(G) > k$ and terminate.

Correctness. We proceed to argue for correctness of the algorithm. It is clear that if the algorithm terminates with one of the first three outcomes, then its output is correct. We proceed to show that if $\mathbf{tw}_{\mathcal{H}}(G) \leq k$, then it will indeed terminate in one of those outcomes. So assume $\mathbf{tw}_{\mathcal{H}}(G) \leq k$, which means we may apply Lemma 13 to S and G. If Case 1 of Lemma 13 holds, then the algorithm will detect the corresponding separation in the first phase of the algorithm and terminate with a suitable separation. So assume Case 2 holds, so that there is a set $S' \subseteq S$ of size $2k+3$ which is weakly $(\mathcal{H}, k+1)$-separable. Since the set S' is a candidate for reaching the second outcome, if that outcome is not reached we have $\lambda_G(S', X) > 2k+2$ and hence $\lambda_G(S', X) = 2k+3 = |S'|$. Consider the family of (S', X)-paths $\mathcal{P}_{S'}$ constructed by the algorithm for this choice of S' and let $X'_{S'}$ be their endpoints in X. The paths $\mathcal{P}_{S'}$ show that $\lambda_G(S', X'_{S'}) = |S'| = |X'_{S'}| = 2k+3$. Now we can apply Lemma 12 for $\ell = k+1$ to infer that $X'_{S'}$ is redundant in X, which implies that $G - (X \setminus X'_{S'})$ has an $\mathcal{H}$-deletion set smaller than $|X'_{S'}| = 2k+3$. Hence algorithm $\mathcal{A}$ outputs an $\mathcal{H}$-deletion set smaller than $|X'_{S'}|$ and the algorithm terminates with the third outcome.

Since the algorithm reaches one the first three outcomes when $\mathbf{tw}_{\mathcal{H}}(G) \leq k$, the algorithm is correct when it reaches the last outcome.

Running time and oracle calls. Each of the three phases of the algorithm consist of enumerating subsets $S' \subseteq S$, of which there are $2^{|S|} \leq 2^{3k+4} = \mathcal{O}(8^k)$. For each such set S', the algorithm performs $\mathcal{O}(k)$ rounds of the Ford-Fulkerson algorithm in time $\mathcal{O}(k(n+m))$. In the last phase, the algorithm additionally invokes $\mathcal{A}$ on an induced subgraph of G for each S' to find an $\mathcal{H}$-deletion set of size at most $2k+2$ if one exists. It follows that the running time of the algorithm (not accounting for the time spent by $\mathcal{A}$) is $\mathcal{O}(8^k \cdot k(n+m))$. The space usage is easily seen to be polynomial in the input size since the algorithm is iterative. This concludes the proof of Lemma 14. ◀

3.3 The decomposition algorithm

We retrace the proof of [16, Theorem 7.18] which gives the classic algorithm for approximating (standard) treewidth. Consider sets $S \subseteq W \subseteq V(G)$ such that $\partial_G(W) \subseteq S$ and $|S| = 3k+4$; we aim to construct a tree decomposition of $G[W]$ which contains S in its root bag. We can consider all ways to partition S into $S_A \cup S_B$ such that $|S_A|, |S_B| \leq 2k+2$ and compute a minimum (S_A, S_B)-separator. Since $|S| = 3k+4$, there are $2^{3k+4} = \mathcal{O}(8^k)$ such partitions.

When $\mathbf{tw}(G) \leq k$, we are guaranteed that for some partition $S = S_A \cup S_B$ we will find a separator in $G[W]$ of size $\leq k+1$ which yields the separation (A_W, B_W) in $G[W]$ satisfying $S_A \subseteq A_W$, $S_B \subseteq B_W$, and $|A_W \cap B_W| \leq k+1$. Then the boundary $\partial_G(A_W)$ is contained in $S_A \cup (A_W \cap B_W)$, and similarly $\partial_G(B_W) \subseteq S_B \cup (A_W \cap B_W)$. We create instances $(A_W, S_A \cup (A_W \cap B_W))$ and $(B_W, S_B \cup (A_W \cap B_W))$ to be solved recursively, analogously as (W, S). Note that each of the sets $S_A \cup (A_W \cap B_W)$, $S_B \cup (A_W \cap B_W)$ has less than $3k+4$ vertices, so we can augment each of them with one more vertex before making the recursive call while preserving the size invariant. This step ensures that the recursion tree has at most $|V(G)|$ nodes. After computing tree decompositions for $G[A]$ and $G[B]$ we merge them by creating a new root with a bag $S \cup (A_W \cap B_W)$ of size at most $4k+5$. Hence, we are able to construct a tree decomposition of width $4k+4$ assuming that one of width k exists.

There are two differences between the outlined algorithm and ours, while the recursive scheme stays the same. First, due to scenario (2) in Lemma 14 we need to handle the cases where we can directly create a base component containing at least $2k+3$ vertices from S. The lower bound $2k+3$ is greater than the separator size $2k+2$ so we will move on to a subproblem where S is significantly smaller. We need to include the separator of size $2k+2$ in the root bag, together with S, so we obtain a slightly weaker bound on the maximum bag size, that is $5k+6$. Next, due to scenario (3) we might not make direct progress in the recursive scheme but instead we reduce the size of an $\mathcal{H}$-deletion set X that we maintain (which initially contains all vertices). This situation can happen at most $|V(G)|$ many times, so eventually we will reach outcome (1) or (2).

The approach sketched above leads to a proof of Theorem 1. The details are deferred to the full version [31] due to space restrictions. Apart from carefully combining the ingredients collected so far, in the proof we take care to optimize the number of calls to the $\mathcal{H}$-DELETION oracle, leading to the clean bound of $\mathcal{O}(8^k n)$ oracle calls advocated in the introduction.

4 Conclusion

We contributed to the algorithmic theory of hybrid graph parameterizations, by showing how a 5-approximation to $\mathbf{tw}_{\mathcal{H}}$ can be obtained using an algorithm for the solution-size parameterization of $\mathcal{H}$-DELETION as a black box. This makes the step of computing a tree $\mathcal{H}$-decomposition now essentially as fast as that of solving $\mathcal{H}$-DELETION parameterized by solution size. Our new decomposition algorithm combines with existing algorithms to solve $\mathcal{H}$-DELETION on a given tree $\mathcal{H}$-decomposition, to deliver algorithms that solve $\mathcal{H}$-DELETION parameterized by $\mathbf{tw}_{\mathcal{H}}$. For Odd Cycle Transversal and Vertex Planarization, the parameter dependence of the resulting algorithm is equal to the worst of the parameter dependencies of the solution-size and treewidth-parameterizations. We believe that this is not a coincidence, and offer the following conjecture.

▶ **Conjecture 15.** *Let $\mathcal{H}$ be a hereditary and union-closed graph class. If $\mathcal{H}$-DELETION can be solved in time $f(s) \cdot n^{\mathcal{O}(1)}$ parameterized by solution size s, and in time $h(w) \cdot n^{\mathcal{O}(1)}$ parameterized by treewidth w, then $\mathcal{H}$-DELETION can be solved in time $(f(\mathcal{O}(k)) + h(\mathcal{O}(k))) \cdot n^{\mathcal{O}(1)}$ parameterized by $\mathcal{H}$-treewidth k.*

The conjecture is a significant strengthening of the equivalence, with respect to non-uniform fixed-parameter tractability, between solving $\mathcal{H}$-DELETION parameterized by solution size and computing $\mathbf{tw}_{\mathcal{H}}$ given by Agrawal et al. [1]. It essentially states that there is no *price of generality* to pay for using the hybrid parameterization by $\mathbf{tw}_{\mathcal{H}}$. After three decades in which the field of parameterized complexity has focused on parameterizations by solution size, this would lead to a substantial shift of perspective. We believe Theorem 1 is an important ingredient in this direction.

To understand the relative power of the parameterizations by solution size, treewidth, and $\mathcal{H}$-treewidth, the remaining bottleneck lies in using the tree $\mathcal{H}$-decomposition to compute a minimum $\mathcal{H}$-deletion set. Can the latter be done as efficiently when using a tree $\mathcal{H}$-decomposition as when using a standard tree decomposition? For problems like Odd Cycle Transversal and Vertex Planarization, this is indeed the case. But when the current-best dynamic-programming algorithm over a tree decomposition uses advanced techniques, it is currently not clear how to lift such an algorithm to work on a tree $\mathcal{H}$-decomposition. Can $\mathcal{H}$-deletion for $\mathcal{H}$ the class of interval graphs be solved in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$ parameterized by $\mathbf{tw}_{\mathcal{H}}$? Such a running time can be obtained for the parameterization by treewidth by adapting the approach of Saitoh, Yoshinaka, and Bodlaender [48].

While we have not touched on the subject here, we expect our ideas to also be applicable when $\mathcal{H}$ is a *scattered graph class*, i.e., when $\mathcal{H}$ consists of graphs where each connected component is contained in one of a finite number of graph classes $\mathcal{H}_1, \ldots, \mathcal{H}_t$. It is known [30] that, when Vertex Cover can be solved in polynomial time on each graph class $\mathcal{H}_i$, then Vertex Cover is FPT parameterized by the width of a given tree $\mathcal{H}$-decomposition. We expect that Theorem 1 can be generalized to work with scattered graph classes $\mathcal{H}$, as long as there is an oracle to solve $\mathcal{H}_i$-deletion parameterized by solution size for each individual class $\mathcal{H}_i$. To accommodate this setting, the algorithm maintains an $\mathcal{H}_i$-deletion set X_i for *each* graph class $\mathcal{H}_i$. A step of the decomposition algorithm then either consists of finding a balanced separation of S, splitting off a base component, or improving *one of the* deletion sets X_i (which can occur only $t \cdot |V(G)|$ times).

The decomposition algorithm we presented has an approximation factor of 5. It may be possible to obtain a smaller approximation ratio at the expense of a worse base of the exponent, by repeatedly splitting large bags [5, 35, 36]. For obtaining single-exponential $\mathcal{H}$-deletion algorithms, the advantage of the improved approximation factor would be immediately lost due to the increased running time and therefore we did not pursue this direction.

A final direction for future work concerns the optimization of the polynomial part of the running time. For standard treewidth, a 2-approximation can be computed in time $2^{\mathcal{O}(k)} \cdot n$ [35], which was obtained after a long series of improvements (cf. [8, Table 1]) on both the approximation factor and dependence on n. Can a constant-factor approximation to $\mathcal{H}$-treewidth be computed in time $2^{\mathcal{O}(k)} \cdot (n+m)$ for graph classes $\mathcal{H}$ like bipartite graphs?

References

1 Akanksha Agrawal, Lawqueen Kanesh, Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Deleting, eliminating and decomposing to hereditary classes are all FPT-equivalent. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 1976–2004. SIAM, 2022. doi:10.1137/1.9781611977073.79.

2 Akanksha Agrawal and M. S. Ramanujan. Distance from triviality 2.0: Hybrid parameterizations. In Cristina Bazgan and Henning Fernau, editors, *Combinatorial Algorithms – 33rd International Workshop, IWOCA 2022, Trier, Germany, June 7-9, 2022, Proceedings*, volume 13270 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2022. doi:10.1007/978-3-031-06678-8_1.

3 Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987. doi:10.1137/0608024.

4 Mahdi Belbasi and Martin Fürer. An improvement of reed's treewidth approximation. *J. Graph Algorithms Appl.*, 26(2):257–282, 2022. doi:10.7155/jgaa.00593.

5 Patrick Bellenbaum and Reinhard Diestel. Two short proofs concerning tree-decompositions. *Comb. Probab. Comput.*, 11(6):541–547, 2002. doi:10.1017/S0963548302005369.

6 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

7 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

8 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.

9 Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996. doi:10.1006/jagm.1996.0049.

10 Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations I. Upper bounds. *Inf. Comput.*, 208(3):259–275, 2010. doi:10.1016/j.ic.2009.03.008.

11 Hans L. Bodlaender and Arie M. C. A. Koster. Treewidth computations II. Lower bounds. *Inf. Comput.*, 209(7):1103–1119, 2011. doi:10.1016/j.ic.2011.04.003.

12 Jannis Bulian and Anuj Dawar. Graph isomorphism parameterized by elimination distance to bounded degree. *Algorithmica*, 75(2):363–382, 2016. doi:10.1007/s00453-015-0045-3.

13 Jannis Bulian and Anuj Dawar. Fixed-parameter tractable distances to sparse graph classes. *Algorithmica*, 79(1):139–158, 2017. doi:10.1007/s00453-016-0235-7.

14 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theor. Comput. Sci.*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

15 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012. doi:10.1017/CBO9780511977619.

16 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

17 Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The first parameterized algorithms and computational experiments challenge. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 30:1–30:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.30.

18 Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:12, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.IPEC.2017.30.

19 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

20 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

21 Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 42:1–42:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.42.

22 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *J. Comput. Syst. Sci.*, 78(3):707–719, 2012. doi:10.1016/j.jcss.2011.10.003.

23 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

24 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Clique-width: on the price of generality. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 825–834. SIAM, 2009. doi:10.1137/1.9781611973068.90.

25 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010. doi:10.1137/080742270.

26 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM J. Comput.*, 43(5):1541–1563, 2014. doi:10.1137/130910932.

27 Jiong Guo, Sepp Hartung, Rolf Niedermeier, and Ondrej Suchý. The parameterized complexity of local search for TSP, more refined. *Algorithmica*, 67(1):89–110, 2013. doi:10.1007/s00453-012-9685-8.

28 Jiong Guo, Danny Hermelin, and Christian Komusiewicz. Local search for string problems: Brute-force is essentially optimal. *Theor. Comput. Sci.*, 525:30–41, 2014. doi:10.1016/j.tcs.2013.05.006.

29 Bart M. P. Jansen and Jari J. H. de Kroon. FPT algorithms to compute the elimination distance to bipartite graphs and more. In Lukasz Kowalik, Michal Pilipczuk, and Pawel Rzazewski, editors, *Graph-Theoretic Concepts in Computer Science – 47th International Workshop, WG 2021, Warsaw, Poland, Revised Selected Papers*, volume 12911 of *Lecture Notes in Computer Science*, pages 80–93. Springer, 2021. doi:10.1007/978-3-030-86838-3_6.

30 Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1757–1769, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451068.

31 Bart M. P. Jansen, Jari J. H. de Kroon, and Michal Włodarczyk. 5-approximation for H-treewidth essentially as fast as H-deletion parameterized by solution size. *CoRR*, abs/2306.17065, 2023. arXiv:2306.17065.

32 Bart M. P. Jansen, Jari J. H. de Kroon, and Michał Włodarczyk. Vertex deletion parameterized by elimination distance and even less. *CoRR*, abs/2105.04660, 2021. URL: https://arxiv.org/abs/2105.04660.

33 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1802–1811. SIAM, 2014. doi:10.1137/1.9781611973402.130.

34 Ken-ichi Kawarabayashi. Planarity allowing few error vertices in linear time. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 639–648. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.45.

35 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 184–192. IEEE, 2021. doi:10.1109/FOCS52979.2021.00026.

36 Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth. *CoRR*, abs/2211.07154, 2022. doi:10.48550/arXiv.2211.07154.

37 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

38 Dániel Marx. Four shorts stories on surprising algorithmic uses of treewidth. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms – Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2020. doi:10.1007/978-3-030-42071-0_10.

39 Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

40 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity – Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

41 Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

42 Marcin Pilipczuk. A tight lower bound for vertex planarization on graphs of bounded treewidth. *Discret. Appl. Math.*, 231:211–216, 2017. doi:10.1016/j.dam.2016.05.019.

43 Bruce A. Reed. Finding approximate separators and computing tree width quickly. In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 221–228. ACM, 1992. doi:10.1145/129712.129734.

44 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.

45 N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

46 Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.

47 Neil Robertson and Paul D. Seymour. Graph minors. IV. Tree-width and well-quasi-ordering. *J. Comb. Theory, Ser. B*, 48(2):227–254, 1990. doi:10.1016/0095-8956(90)90120-O.

48 Toshiki Saitoh, Ryo Yoshinaka, and Hans L. Bodlaender. Fixed-treewidth-efficient algorithms for edge-deletion to interval graph classes. In Ryuhei Uehara, Seok-Hee Hong, and Subhas C. Nandy, editors, *WALCOM: Algorithms and Computation – 15th International Conference and Workshops, WALCOM 2021, Yangon, Myanmar, February 28 – March 2, 2021, Proceedings*, volume 12635 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2021. doi:10.1007/978-3-030-68211-8_12.

49 A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.