Computing Tree Decompositions with Small Independence Number

Clément Dallard
Department of Informatics, University of Fribourg, Switzerland

Fedor V. Fomin
University of Bergen, Norway

Petr A. Golovach
University of Bergen, Norway

Tuukka Korhonen
University of Bergen, Norway

Martin Milanič
FAMNIT and IAM, University of Primorska, Koper, Slovenia

Abstract

The independence number of a tree decomposition is the maximum of the independence numbers of the subgraphs induced by its bags. The tree-independence number of a graph is the minimum independence number of a tree decomposition of it. Several NP-hard graph problems, like maximum weight independent set, can be solved in time $n^{\mathcal{O}(k)}$ if the input n-vertex graph is given together with a tree decomposition of independence number k. Yolov in [SODA 2018] gave an algorithm that given an n-vertex graph G and an integer k, in time $n^{\mathcal{O}(k^3)}$ either constructs a tree decomposition of G whose independence number is $\mathcal{O}(k^3)$ or correctly reports that the tree-independence number of G is larger than k.

In this paper, we first give an algorithm for computing the tree-independence number with a better approximation ratio and running time and then prove that our algorithm is, in some sense, the best one can hope for. More precisely, our algorithm runs in time $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ and either outputs a tree decomposition of G with independence number at most $8k$, or determines that the tree-independence number of G is larger than k. This implies $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$-time algorithms for various problems, like maximum weight independent set, parameterized by the tree-independence number k without needing the decomposition as an input. Assuming Gap-ETH, an $n^{\Omega(k)}$ factor in the running time is unavoidable for any approximation algorithm for the tree-independence number.

Our second result is that the exact computation of the tree-independence number is para-NP-hard: We show that for every constant $k \geq 4$ it is NP-hard to decide if a given graph has the tree-independence number at most k.

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 tree-independence number, approximation, parameterized algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.51

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2207.09993` [13]

Funding The research leading to these results has received funding from the Research Council of Norway via the project BWCA (grant no. 314528), by the Slovenian Research and Innovation Agency (I0-0035, research program P1-0285 and research projects J1-3001, J1-3002, J1-3003, J1-4008, and J1-4084), and by the research program CogniCom (0013103) at the University of Primorska.

1 Introduction

Tree decompositions are among the most popular tools in graph algorithms. The crucial property of tree decompositions exploited in the majority of dynamic programming algorithms is that each bag of the decomposition can interact with an optimal solution only in a bounded number of vertices. The common measure of a tree decomposition is the width, that is, the maximum size of a bag in the decomposition (minus 1). The corresponding graph parameter is the treewidth of a graph. Many problems that are intractable on general graphs can be solved efficiently when the treewidth of a graph is bounded.

However, it is not always the size of a bag that matters. For example, suppose that every bag of the decomposition is a clique, that is, the graph is chordal. Since every independent set intersects each of the clique-bags in at most one vertex, dynamic programming still computes maximum weight independent sets in such graphs in polynomial time even if the bags could be arbitrarily large. An elegant approach to capturing such properties of tree decompositions is the notion of the tree-independence number of a graph. The *independence number* of a tree decomposition is the maximum of the independence numbers (that is, the maximum size of an independent set) of the subgraphs induced by its bags. The *tree-independence number* of a graph G, denoted by $\mathsf{tree}\text{-}\alpha(G)$, is the minimum independence number of a tree decomposition of G. In particular, the tree-independence number of a chordal graph is at most one, and for any graph G, the value $\mathsf{tree}\text{-}\alpha(G)$ does not exceed the treewidth of G (plus 1) or the independence number $\alpha(G)$ of G.

The family of graph classes with bounded tree-independence number forms a significant generalization of graph classes with bounded treewidth. It also contains dense graphs classes, including graph classes with bounded independence number; classes of intersection graphs of connected subgraphs of graphs with bounded treewidth, studied by Bodlaender, Gustedt, and Telle [6], which in particular include classes of H*-graphs*, that is, intersection graphs of connected subgraphs of a subdivision of a fixed multigraph H, introduced in 1992 by Bíró, Hujter, and Tuza [4] and studied more recently in a number of papers [8, 9, 22]; classes of graphs in which all minimal separators have bounded size, studied by Skodinis in 1999 [38]; and, more generally, classes of graphs in which all minimal separators induce subgraphs with bounded independence number, studied by Dallard, Milanič, and Štorgel [15].

Our results. Yolov [40] gave an algorithm that for a given n-vertex graph G and integer k, in time $n^{\mathcal{O}(k^3)}$ either constructs a tree decomposition of G whose independence number is $\mathcal{O}(k^3)$ or correctly reports that the tree-independence number of G is larger than k. Our first main result is the following improvement over Yolov's algorithm.

▶ **Theorem 1.** *There is an algorithm that, given an n-vertex graph G and an integer k, in time $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$ either outputs a tree decomposition of G with independence number at most $8k$, or concludes that the tree-independence number of G is larger than k.*

The performance of the algorithm from Theorem 1 (the running time and the need of approximation) are in some sense optimal, for the following reasons. First, by a simple reduction that given an n-vertex graph G produces a $2n$-vertex graph G' with $\mathsf{tree}\text{-}\alpha(G') = \alpha(G)$ hardness results for the independence number (or clique) translate into hardness results for the tree-independence number. In particular, from the result of Lin [30] it follows that a constant-factor approximation of the tree-independence number is $\mathsf{W[1]}$-hard, and from the result of Chalermsook, Cygan, Kortsarz, Laekhanukit, Manurangsi, Nanongkai, and Trevisan [7] it follows that assuming Gap-ETH[1], there is no $f(k) \cdot n^{o(k)}$-time $g(k)$-approximation algorithm for the tree-independence number for any computable functions f and g. In particular, the time complexity obtained in Theorem 1 is optimal in the sense that an $n^{\Omega(k)}$ factor is unavoidable assuming Gap-ETH, even if the approximation ratio would be drastically weakened.

The above arguments do not exclude the possibility of *exact* computation of the tree-independence number in time $n^{f(k)}$ for some function f. The computational complexity of recognizing graphs with the tree-independence numbers at most k for a fixed integer $k \geq 2$ was asked as an open problem by Dallard, Milanič, and Štorgel [15, Question 8.3][2]. While for values $k = 2$ and $k = 3$ the question remains open, our next result resolves it for any constant $k \geq 4$.[3]

▶ **Theorem 2** (⋆). *For every constant $k \geq 4$, it is* NP*-complete to decide whether* $\mathsf{tree}\text{-}\alpha(G) \leq k$ *for a given graph G.*

Let us observe that Theorem 2 implies also that, assuming P ≠ NP, there is no $n^{f(k)}$-time approximation algorithm for the tree-independence number with approximation ratio less than 5/4.

We supplement our main results with a second NP-completeness proof for a problem closely related to computing the tree-independence number and the algorithm of Theorem 1. We consider the problem where we are given a graph G, two non-adjacent vertices u and v, and an integer k, and the task is to decide if u and v can be separated by removing a set of vertices that induces a subgraph with independence number at most k. We show in Theorem 14 that this problem is NP-complete for any fixed integer $k \geq 3$. This hardness result is motivated by the fact that the algorithm of Theorem 1 finds separators with bounded independence number as a subroutine. While for the algorithm of Theorem 1 we design a $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$-time 2-approximation algorithm for a generalization of this problem, assuming that a tree decomposition of independence number $\mathcal{O}(k)$ is given, this proof shows that this step of the algorithm cannot be turned into an exact algorithm (in our reduction, we can construct along the graph G also a tree decomposition of independence number $\mathcal{O}(k)$ of G).

Previous work and applications of Theorem 1. The notion of the tree-independence number is very natural and it is not surprising that it was introduced independently by several researchers [15, 40]. Yolov in [40] introduced a new width measure called *minor-matching hypertree-width*, $\mathsf{tree}\text{-}\mu$.[4] He proved that a number of problems including Maximum Independent Set, k-Colouring, and Graph Homomorphism permit polynomial-time solutions for graphs with bounded minor-matching hypertree width. Furthermore, Yolov showed that the minor-matching hypertree-width of a graph is equal to the tree-independence number of the square of its line graph, that is, $\mathsf{tree}\text{-}\mu(G) = \mathsf{tree}\text{-}\alpha(L(G)^2)$ holds for all graphs G, where $L(G)^2$ is the graph whose vertices are the edges of G, with two distinct edges adjacent if and only if they have nonempty intersection or there is a third edge

[1] Gap-ETH states that for some constant $\epsilon > 0$, distinguishing between a satisfiable 3-SAT formula and one that is not even $(1 - \epsilon)$-satisfiable requires exponential time (see [20, 32]).

[2] There is a linear-time algorithm for deciding if a given graph has the tree-independence number at most 1, because such graphs are exactly the chordal graphs, see, e.g., [25].

[3] The proofs of the statements marked (⋆) are omitted due to space constraints and can be found in the full version of the paper [13].

[4] Minor-matching hypertree-width is defined for hypergraphs, but algorithms for computing decompositions for it are only known for graphs.

intersecting both. Moreover, a tree decomposition of $L(G)^2$ with independence number at most k can be turned into a tree decomposition of G with minor-matching hypertree-width at most k. Then, Yolov gave an $n^{\mathcal{O}(k^3)}$-time $\mathcal{O}(k^2)$-approximation algorithm computing the tree-independence number of an n-vertex graph, implying also the same bounds for computing the minor-matching hypertree-width of a graph. Theorem 1 improves the running time and approximation ratio of Yolov's algorithm. Pipelined with Yolov's reduction, Theorem 1 also implies an 8-approximation of minor-matching hypertree-width of graphs in time $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$.

Theorem 2 implies also the NP-hardness of deciding whether $\mathsf{tree}\text{-}\mu(G) \leq k$ for every constant $k \geq 4$ because a simple reduction that attaches a pendant vertex to every vertex of a graph G produces a graph G' such that $\mathsf{tree}\text{-}\mu(G') = \mathsf{tree}\text{-}\alpha(G)$ (see [40]).

The tree-independence number of a graph was introduced independently by Yolov [40] and Dallard et al. [15].[5] The original motivation for Dallard et al. [15] stems from structural graph theory. In 2020, Dallard, Milanič, and Štorgel [14, 17] initiated a systematic study of (tw, ω)-bounded graph classes, that is, hereditary graph classes in which the treewidth can only be large due to the presence of a large clique. While (tw, ω)-bounded graph classes are known to possess some good algorithmic properties related to clique and coloring problems (see [9, 10, 14, 15, 17]), the extent to which this property has useful algorithmic implications for problems related to independent sets is an open problem. The connection with the tree-independence number follows from Ramsey's theorem, which implies that graph classes with bounded tree-independence number are (tw, ω)-bounded with a polynomial binding function (see [15]). Dallard, Milanič, and Štorgel [16] conjecture the converse, namely, that every (tw, ω)-bounded graph class has bounded tree-independence number. A related research direction in structural graph theory is the study of induced obstructions to bounded treewidth and tree-independence number; see for example the recent work of Abrishami, Alecu, Chudnovsky, Hajebi, Spirkl, and Vušković [1].

In our opinion, the most interesting application of Theorem 1 lies in the area of graph algorithms for NP-hard optimization problems. Dallard et al. [15] and Yolov in [40] have shown that certain NP-hard optimization problems like Maximum Independent Set, Graph Homomorphism, or Maximum Induced Matching problems can be solved in time $n^{\mathcal{O}(k)}$ if the input graph is given with a tree decomposition of independence number at most k. Lima et al. [29] extended this idea to generic packing problems in which any two of the chosen subgraphs have to be at pairwise distance at least d, for even d. They also obtained an algorithmic metatheorem for the problem of finding a maximum-weight sparse (bounded chromatic number) induced subgraph satisfying an arbitrary but fixed property expressible in counting monadic second-order logic (CMSO_2). In particular, the metatheorem implies polynomial-time solvability of several classical problems like finding the largest induced forest (which is equivalent to Minimum Feedback Vertex Set), finding the largest induced bipartite subgraph (which is equivalent to Minimum Odd Cycle Transversal), finding the maximum number of pairwise disjoint and non-adjacent cycles (Maximum Induced Cycle Packing), and finding the largest induced planar subgraph (which is equivalent to Planarization).

However, the weak spot in all these algorithmic approaches is the requirement that a tree decomposition with bounded independence number is given with the input. Theorem 1 fills this gap by constructing a decomposition of bounded independence number in time that asymptotically matches or improves the time required to solve all these optimization problems.

[5] Yolov called it *α-treewidth* in [40].

Theorem 1 appears to be a handy tool in the subarea of computational geometry concerning optimization problems on geometric graphs. Treewidth plays a fundamental role in the design of exact and approximation algorithms on planar graphs (and more generally, H-minor-free graphs) [3, 19, 26]. The main property of such graphs is that they enjoy the bounded local treewidth property. In other words, any planar graph of a small diameter has a small treewidth. A natural research direction is to extend such methods to intersection graphs of geometric objects [23, 31]. However, even for very "simple" objects like unit disks, the corresponding intersection graphs do not have locally bounded treewidth. On the other hand, in many scenarios, the treewidth-based methods on such graphs could be replaced by tree decompositions of bounded independence number. In particular, de Berg, Bodlaender, Kisfaludi-Bak, Marx, and van der Zanden use tree decompositions whose bags are covered by a small number of cliques, and thus of small independence number, to design subexponential-time algorithms on geometric graph classes [18]. Galby, Munaro, and Yang [24] use Theorem 1 for obtaining polynomial-time approximation schemes for several packing problems on geometric graphs. It is interesting to note that algorithms on geometric graphs often require geometric representation of a graph. Sometimes, like for unit disk graphs, finding such a representation is a challenging computational task [27]. On the contrary, Theorem 1 does not need the geometric properties of objects or their geometric representations and thus could be used for developing so-called robust algorithms [35] on geometric graphs [11].

In parameterized algorithms, Fomin and Golovach [21] and Jacob, Panolan, Raman, and Sahlot [28] used tree decompositions where each bag is obtained from a clique by deleting at most k edges or adding at most k vertices, respectively. These type of decompositions are special types of tree decompositions with bounded independence numbers.

The rest of this paper is organized as follows. In Section 2, we overview the proof of our main algorithmic result Theorem 1. In Section 3 we recall definitions. Section 4 is devoted to the proof of Theorem 1. In Section 5, we survey our computational lower bounds. Due to space constraints, the proofs of the hardness results are omitted and are available in the full version of the paper [13]. We conclude in Section 6 with final remarks and open problems.

2 Overview

In this section we sketch the proof of Theorem 1. For simplicity, we sketch here a version with approximation ratio 11 instead of 8. The difference between the 11-approximation and 8-approximation is that for 11-approximation it is sufficient to use 2-way separators, while for 8-approximation we use 3-way separators.

On a high level, our algorithm follows the classical technique of constructing a tree decomposition by repeatedly finding balanced separators. This technique was introduced by Robertson and Seymour in Graph Minors XIII [37], was used for example in [5, 18, 28], and an exposition of it was given in [12, Section 7.6.2] and in [34].

The challenge in applying this technique is the need to compute separators that are both balanced and small with respect to the independence numbers of the involved vertex sets. Our main technical contribution is an approximation algorithm for finding such separators. In what follows, we will sketch an algorithm that given a graph G, a parameter k, and a set of vertices $W \subseteq V(G)$ with independence number $\alpha(W) = 9k$ either finds a partition (S, C_1, C_2) of $V(G)$ such that each S, C_1, and C_2 are nonempty, S separates C_1 from C_2, $\alpha(S) \leq 2k$, and $\alpha(W \cap C_i) \leq 7k$ for both $i = 1, 2$, or determines that the tree-independence number of G is larger than k. (For $S \subseteq V(G)$ we use $\alpha(S)$ to denote the independence number of the induced subgraph $G[S]$.) Our algorithm for finding such balanced separators works in two parts. First, we reduce finding balanced separators to finding separators, and then we give a 2-approximation algorithm for finding separators.

At first glance, it is not even clear that small tree-independence number guarantees the existence of such balanced separators. To prove the existence of balanced separators and to reduce the finding of balanced separators to finding separators between specified sets of vertices, instead of directly enforcing that both sides of the separation have a small independence number, we enforce that both sides of the separation have sufficiently large independence number. More precisely, we pick an arbitrary independent set $I \subseteq W$ of size $|I| = 9k$. By making use of the properties of tree decompositions, it is possible to show that there exists a separation (S, C_1, C_2) with $|I \cap C_i| \leq 6k$ for $i \in \{1, 2\}$ and $\alpha(S) \leq k$. Hence $|I \cap C_i| \geq 2k$ for $i \in \{1, 2\}$. By enforcing the condition $|I \cap C_i| \geq 2k$ for both $i \in \{1, 2\}$, we will have that $\alpha(W \cap C_i) \leq 7k$ for both $i \in \{1, 2\}$. (Note that $\alpha(W \cap C_1) + \alpha(W \cap C_2) \leq \alpha(W)$.) Therefore, to find a balanced separator for W or to conclude that the tree-independence number of G is larger than k, it is sufficient to select an arbitrary independent set $I \subseteq W$ with $|I| = 9k$, do $2^{\mathcal{O}(k)}$ guesses for sets $I \cap C_1$ and $I \cap C_2$, and for each of the guesses search for a separator between the sets $I \cap C_1$ and $I \cap C_2$.

In the separator finding algorithm the input includes two sets $V_1 = I \cap C_1$ and $V_2 = I \cap C_2$, and the task is to find a set of vertices S disjoint from both V_1 and V_2 separating V_1 from V_2 with $\alpha(S) \leq 2k$ or to conclude that no such separator S with $\alpha(S) \leq k$ exists. Our algorithm works by first using multiple stages of different branching steps, and then arriving at a special case which is solved by rounding a linear program. We explain some details in what follows.

First, by using iterative compression around the whole algorithm, we can assume that we have a tree decomposition with independence number $\mathcal{O}(k)$ available. We show that any set $S \subseteq V(G)$ with $\alpha(S) \leq k$ can be covered by $\mathcal{O}(k)$ bags of the tree decomposition. This implies that by first guessing the covering bags, we reduce the problem to the case where we search for a separator $S \subseteq R$ for some set $R \subseteq V(G)$ with independence number $\alpha(R) = \mathcal{O}(k^2)$.

Then, we use a branching procedure to reduce the problem to the case where $R \subseteq N[V_1 \cup V_2]$. In the branching, we select a vertex $v \in R \setminus N[V_1 \cup V_2]$, and branch into three subproblems, corresponding to including v into V_1, into V_2, or into a partially constructed solution S. The key observation here is that if we branch on vertices $v \in R \setminus N[V_1 \cup V_2]$, then the branches where v is included in V_1 or in V_2 reduce the value $\alpha(R \setminus N[V_1 \cup V_2])$. By first handling the case with $\alpha(R \setminus N[V_1 \cup V_2]) \geq 2k$ by branching on $2k$ vertices at the same time and then branching on single vertices, this branching results in $2^{\mathcal{O}(\alpha(R))} n^{\mathcal{O}(k)} = 2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ instances where $R \subseteq N[V_1 \cup V_2]$.

Finally, when we arrive at the subproblem where $R \subseteq N[V_1 \cup V_2]$, we design a 2-approximation algorithm by rounding a linear program. For $v \in R$, let us have variables x_v with $x_v = 1$ indicating that $v \in S$ and $x_v = 0$ indicating that $v \notin S$. Because $R \subseteq N[V_1 \cup V_2]$, the fact that $S \subseteq R$ separates V_1 from V_2 can be encoded by only using inequalities of form $x_v + x_u \geq 1$. To bound the independence number of S, for every independent set I of size $|I| = 2k+1$ we add a constraint that $\sum_{v \in I} x_v \leq k$. Now, a separator S with $\alpha(S) \leq k$ corresponds to an integer solution. We then find a fractional solution and round x_v to 1 if $x_v \geq 1/2$ and to 0 otherwise. Note that this satisfies the $x_v + x_u \geq 1$ constraints, so the rounded solution corresponds to a separator. To bound the independence number of the rounded solution, note that $\sum_{v \in I} x_v \leq 2k$ for independent sets I of size $|I| = 2k + 1$, therefore implying that $\alpha(S) \leq 2k$.

3 Preliminaries

We denote the vertex set and the edge set of a graph $G = (V, E)$ by $V(G)$ and $E(G)$, respectively. The *neighborhood* of a vertex v in G is the set $N_G(v)$ of vertices adjacent to v in G, and the *closed neighborhood* of v is the set $N_G[v] = N_G(v) \cup \{v\}$. These concepts are extended to sets $X \subseteq V(G)$ so that $N_G[X]$ is defined as the union of all closed neighborhoods of vertices in X, and $N_G(X)$ is defined as the set $N_G[X] \setminus X$. The *degree* of v, denoted by $d_G(v)$, is the cardinality of the set $N_G(v)$. When there is no ambiguity, we may omit the subscript G in the notations of the degree, and open and closed neighborhoods, and thus simply write $d(v)$, $N(v)$, and $N[v]$, respectively.

Given a set $X \subseteq V(G)$, we denote by $G[X]$ the subgraph of G induced by X. We also write $G \setminus X = G[V(G) \setminus X]$. Similarly, given a vertex $v \in V(G)$, we denote by $G \setminus v$ the graph obtained from G by deleting v. A graph G is *chordal* if it has no induced cycles of length at least four.

A *clique* (resp. *independent set*) in a graph G is a set of pairwise adjacent (resp. non-adjacent) vertices. The *independence number* of G, denoted by $\alpha(G)$, is the maximum size of an independent set in G. For a set of vertices $X \subseteq V(G)$, the independence number of X is $\alpha(X) = \alpha(G[X])$.

Let $(V_1, V_2, \ldots, V_t)$ be a tuple of disjoint subsets of $V(G)$. A $(V_1, V_2, \ldots, V_t)$*-separator* is a set $S \subseteq V(G)$ such that $S \cap V_i = \emptyset$ for each $i \in [t]$, and in the graph $G \setminus S$ there is no path from V_i to V_j for all pairs $i \neq j$. Such a separator is sometimes called a t-way separator. Note that if V_i is adjacent to V_j for some $i \neq j$, then no $(V_1, V_2, \ldots, V_t)$-separator exists.

A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ where T is a tree and every node t of T is assigned a vertex subset $X_t \subseteq V(G)$ called a bag such that the following conditions are satisfied: (1) every vertex of G is in at least one bag, (2) for every edge $uv \in E(G)$ there exists a node $t \in V(T)$ such that X_t contains both u and v, and (3) for every vertex $u \in V(G)$ the subgraph T_u of T induced by the set $\{t \in V(T) : u \in X_t\}$ is connected (that is, a tree). The *independence number* of a tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of a graph G, denoted by $\alpha(\mathcal{T})$, is defined as follows:

$$\alpha(\mathcal{T}) = \max_{t \in V(T)} \alpha(X_t)\,.$$

The *tree-independence number* of G, denoted by $\mathsf{tree}\text{-}\alpha(G)$, is the minimum independence number among all tree decompositions of G.

4 An 8-approximation algorithm for tree-independence number

In this section we prove Theorem 1, that is, we give a $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$-time algorithm for either computing tree decompositions with independence number at most $8k$ or deciding that the tree-independence number of the graph is more than k. Our algorithm consists of three parts. First, we give a $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$-time 2-approximation algorithm for finding 3-way separators with independence number at most k, with the assumption that a tree decomposition with independence number $\mathcal{O}(k)$ is given with the input. Then, we apply this separator finding algorithm to find balanced separators, and then apply balanced separators in the fashion of the Robertson-Seymour treewidth approximation algorithm [37] to construct a tree decomposition with independence number at most $8k$. The requirement for having a tree decomposition with independence number $\mathcal{O}(k)$ as an input in the separator algorithm is satisfied by iterative compression (see, e.g., [12]), as we explain at the end of Section 4.3.

The presentation of the algorithm in this section is in reverse order compared to the presentation we gave in Section 2.

4.1 Finding approximate separators

In this subsection, we show the following theorem.

▶ **Theorem 3.** *There is an algorithm, that given a graph G, an integer k, a tree decomposition $\mathcal{T}$ of G with independence number $\alpha(\mathcal{T}) = \mathcal{O}(k)$, and three disjoint sets of vertices $V_1, V_2, V_3 \subseteq V(G)$, in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ either reports that no (V_1, V_2, V_3)-separator with independence number at most k exists, or returns a (V_1, V_2, V_3)-separator with independence number at most $2k$.*

To prove Theorem 3, we define a more general problem that we call PARTIAL 3-WAY α-SEPARATOR, which is the same as the problem of Theorem 3 except that two sets S_0 and R are given with the input, and we are only looking for separators S with $S_0 \subseteq S \subseteq S_0 \cup R$.

▶ **Definition 4** (PARTIAL 3-WAY α-SEPARATOR)**.** *An instance of PARTIAL 3-WAY α-SEPARATOR is a 5-tuple $(G, (V_1, V_2, V_3), S_0, R, k)$, where G is a graph, V_1, V_2, V_3, S_0, and R are disjoint subsets of $V(G)$, and k is an integer. A solution to PARTIAL 3-WAY α-SEPARATOR is a (V_1, V_2, V_3)-separator S such that $S_0 \subseteq S \subseteq S_0 \cup R$. A 2-approximation algorithm for PARTIAL 3-WAY α-SEPARATOR either returns a solution S with $\alpha(S) \leq 2k$ or determines that there is no solution S with $\alpha(S) \leq k$.*

We give a 2-approximation algorithm for PARTIAL 3-WAY α-SEPARATOR. Then Theorem 3 will follow by setting $S_0 = \emptyset$ and $R = V(G) \setminus (V_1 \cup V_2 \cup V_3)$.

We give our 2-approximation algorithm by giving 2-approximation algorithms for increasingly more general cases of PARTIAL 3-WAY α-SEPARATOR. First, we give a linear programming based 2-approximation algorithm for the special case when $R \subseteq N(V_1 \cup V_2 \cup V_3)$. Then, we use branching and the first algorithm to give a 2-approximation algorithm for the case when $\alpha(R) = \mathcal{O}(k^2)$. Then, we use the input tree decomposition to reduce the general case to the case where $\alpha(R) = \mathcal{O}(k^2)$.

Let us make some observations about trivial instances of PARTIAL 3-WAY α-SEPARATOR. First, we can assume that $\alpha(S_0) \leq k$, as otherwise any solution S has $\alpha(S) > k$ and we can return no immediately. All our algorithms include an $n^{\mathcal{O}(k)}$ factor in the time complexity, so it can be assumed that this condition is always tested. Then, we can also always return no if V_i is adjacent to V_j, where $i \neq j$. This can be checked in polynomial time, so we can assume that this condition is always tested. Note that testing this condition implies that $N(V_1 \cup V_2 \cup V_3) = N(V_1) \cup N(V_2) \cup N(V_3)$. For simplicity, we write $N(V_1 \cup V_2 \cup V_3)$ as it is shorter.

We start with the linear programming based 2-approximation algorithm for the case when $R \subseteq N(V_1 \cup V_2 \cup V_3)$.

▶ **Lemma 5.** *There is an $n^{\mathcal{O}(k)}$-time 2-approximation algorithm for PARTIAL 3-WAY α-SEPARATOR when $R \subseteq N(V_1 \cup V_2 \cup V_3)$.*

Proof. First, note that we may assume that for all $i, j \in \{1, 2, 3\}$, $i \neq j$, there is no path in the graph $G \setminus (R \cup S_0)$ between a vertex of V_i and a vertex of V_j, since otherwise the given instance has no solution at all. Furthermore, under this assumption, we can safely replace each set V_i with the set of vertices reachable from a vertex in V_i in the graph $G \setminus (R \cup S_0)$. We thus arrive at an instance such that $R \subseteq N(V_1 \cup V_2 \cup V_3) \subseteq R \cup S_0$. We then make an integer programming formulation of the problem (with $n^{\mathcal{O}(k)}$ constraints) and show that it gives a 2-approximation by rounding a fractional solution.

For every vertex $v \in R \cup S_0$ we introduce a variable x_v, with the interpretation that $x_v = 1$ if $v \in S$ and $x_v = 0$ otherwise. We force S_0 to be in the solution by adding constraints $x_v = 1$ for all $v \in S_0$. Then, we say that a pair (v_i, v_j) of vertices with $v_i \in N(V_i)$, $v_j \in N(V_j)$,

$i \neq j$, is *connected* if there is a v_i, v_j-path in the graph $G \setminus ((S_0 \cup R) \setminus \{v_i, v_j\})$. For any pair (v_i, v_j) of connected vertices we introduce a constraint $x_{v_i} + x_{v_j} \geq 1$ indicating that v_i or v_j should be selected to the solution. Note that here it can happen that $v_i = v_j$, corresponding to the case when $v_i \in N(V_i) \cap N(V_j)$, resulting in a constraint forcing v_i to be in the solution. Finally, for every independent set $I \subseteq R \cup S_0$ of size $|I| = 2k + 1$, we introduce a constraint $\sum_{v \in I} x_v \leq k$.

We observe that any solution S with $\alpha(S) \leq k$ corresponds to a solution to the integer program. In particular, any (V_1, V_2, V_3)-separator S must satisfy the $x_{v_i} + x_{v_j} \geq 1$ constraints for connected pairs (v_i, v_j), as otherwise there would be a V_i,V_j-path in $G \setminus S$, and a separator with $\alpha(S) \leq k$ satisfies the independent set constraints as otherwise it would contain an independent set larger than k.

Then, we show that any integral solution that satisfies the $x_{v_i} + x_{v_j} \geq 1$ constraints for connected pairs (v_i, v_j) and the constraints $x_v = 1$ for $v \in S_0$ corresponds to a (V_1, V_2, V_3)-separator. Suppose that all such constraints are satisfied by an integral solution corresponding to a set S, but there is a path from V_i to V_j with $i \neq j$ in $G \setminus S$. Take a shortest path connecting $N(V_i) \setminus S$ to $N(V_j) \setminus S$ in $G \setminus S$ for any $i \neq j$, and note that the intermediate vertices of such path do not intersect $N(V_1 \cup V_2 \cup V_3)$, as otherwise we could get a shorter path, and therefore do not intersect R. By $S_0 \subseteq S$, we have that the intermediate vertices do not intersect $R \cup S_0$, so this path is in fact a path in the graph $G \setminus ((S_0 \cup R) \setminus \{v_i, v_j\})$, where $v_i \in N(V_i) \setminus S$ is the first vertex and $v_j \in N(V_j) \setminus S$ is the last vertex. However, in this case (v_i, v_j) is a connected pair and we have a constraint $x_{v_i} + x_{v_j} \geq 1$, which would be violated by such an integral solution, so we get a contradiction.

We solve the linear program in polynomial time (which is $n^{\mathcal{O}(k)}$ as the number of variables is $\mathcal{O}(n)$ and the number of constraints is $n^{\mathcal{O}(k)}$), and round the solution by rounding x_v to 1 if $x_v \geq 1/2$ and otherwise to 0. This rounding will satisfy the $x_{v_i} + x_{v_j} \geq 1$ constraints for connected pairs, so the resulting solution corresponds to a separator. Then, by the constraints $\sum_{v \in I} x_v \leq k$ for independent sets I of size $2k + 1$, the rounded solution will satisfy $\sum_{v \in I} x_v \leq 2k$ for independent sets I of size $2k+1$. Therefore, there are no independent sets of size $2k + 1$ in the resulting solution, so its independence number is at most $2k$. ◀

We will pipeline Lemma 5 with branching to obtain a 2-approximation algorithm for PARTIAL 3-WAY α-SEPARATOR in a setting where $\alpha(R)$ is small. In our final algorithm, $\alpha(R)$ will be bounded by $\mathcal{O}(k^2)$, and this is the part that causes the $2^{\mathcal{O}(k^2)}$ factor in the time complexity.

Let us observe that we can naturally branch on vertices in R in instances of PARTIAL 3-WAY α-SEPARATOR.

▶ **Lemma 6** (Branching). *Let $\mathcal{I} = (G, (V_1, V_2, V_3), S_0, R, k)$ be an instance of* PARTIAL 3-WAY α-SEPARATOR*, and let $v \in R$. If S is a solution of $\mathcal{I}$, then S is also a solution of at least one of*

1. $(G, (V_1 \cup \{v\}, V_2, V_3), S_0, R \setminus \{v\}, k)$,
2. $(G, (V_1, V_2 \cup \{v\}, V_3), S_0, R \setminus \{v\}, k)$,
3. $(G, (V_1, V_2, V_3 \cup \{v\}), S_0, R \setminus \{v\}, k)$, *or*
4. $(G, (V_1, V_2, V_3), S_0 \cup \{v\}, R \setminus \{v\}, k)$.

Moreover, any solution of any of the instances 1-4 is also a solution of $\mathcal{I}$.

Proof. If S is a solution of $\mathcal{I}$, we can partition $V(G) \setminus S$ to parts $V_1' \supseteq V_1$, $V_2' \supseteq V_2$, and $V_3' \supseteq V_3$ by including the vertices in any connected component of $G \setminus S$ containing vertices of V_i into V_i' for all $i \in \{1, 2, 3\}$, and including the vertices in connected components containing no vertices of V_1, V_2, V_3 into V_1', resulting in a partition (V_1', V_2', V_3') of $V(G) \setminus S$ such that S is a (V_1', V_2', V_3')-separator. Therefore, the first branch corresponds to when $v \in V_1'$, the second when $v \in V_2'$, the third when $v \in V_3'$, and the fourth when $v \in S$.

The direction that any solution of any of the instances 1-4 is also a solution of the original instance is immediate from the fact that we do not remove vertices from any V_i or S_0. ◀

Lemma 6 allows to branch into four subproblems, corresponding to the situation whether a vertex from R goes to V_1, V_2, V_3, or to S_0. Now, our goal is to branch until we derive an instance with $R \subseteq N(V_1 \cup V_2 \cup V_3)$, which can be solved by Lemma 5. In particular, we would like to branch on vertices $v \in R \setminus N(V_1 \cup V_2 \cup V_3)$. The following lemma shows that we can use $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$ as a measure of progress in the branching.

▶ **Lemma 7.** *For any vertex $v \in R \setminus N(V_1 \cup V_2 \cup V_3)$ it holds that $\alpha(R \setminus (N[v] \cup N(V_1 \cup V_2 \cup V_3))) < \alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$.*

Proof. If $\alpha(R \setminus (N[v] \cup N(V_1 \cup V_2 \cup V_3))) \geq \alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$, then we could take an independent set $I \subseteq R \setminus (N[v] \cup N(V_1 \cup V_2 \cup V_3))$ such that $|I| = \alpha(R \setminus N(V_1 \cup V_2 \cup V_3)$ and construct an independent set $I \cup \{v\} \subseteq R \setminus N(V_1 \cup V_2 \cup V_3)$ of size $|I \cup \{v\}| > \alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$, which is a contradiction. ◀

Lemma 7 implies that when branching on a vertex $v \in R \setminus N(V_1 \cup V_2 \cup V_3)$, the branches where v is moved to V_1, V_2, or V_3 decrease $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$. This will be the main idea of our algorithm for the case when $\alpha(R)$ is bounded, which we give next.

▶ **Lemma 8.** *There is a $2^{\mathcal{O}(\alpha(R))} n^{\mathcal{O}(k)}$-time 2-approximation algorithm for PARTIAL 3-WAY α-SEPARATOR.*

Proof. We give a branching algorithm, whose base case is the case when $R \subseteq N(V_1 \cup V_2 \cup V_3)$, which is solved by Lemma 5. The main idea is to analyze the branching by the parameter $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$, in particular with $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3)) = 0$ corresponding to the base case. The branching itself will be "exact" in the sense that all four cases of Lemma 6 are always included, in particular, the 2-approximation is caused only by the application of Lemma 5.

First, while $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3)) \geq 2k$, which can be checked in $n^{\mathcal{O}(k)}$ time, we branch as follows. We select an independent set $I \subseteq R \setminus N(V_1 \cup V_2 \cup V_3)$ of size $|I| = 2k$, and for all of its vertices we branch on whether to move it into V_1, V_2, V_3, or S_0, i.e., according to Lemma 6. Because I is an independent set, at most k of the vertices can go to S_0, so at least k go to V_1, V_2, or V_3. Also for the reason that I is an independent set, Lemma 7 can be successively applied for all of the vertices that go to V_1, V_2, or V_3. Therefore, this decreases $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$ by at least k, so the depth of this recursion is at most $\alpha(R)/k$, so the total number of nodes in this branching tree is at most $(4^{2k})^{\alpha(R)/k} = 2^{\mathcal{O}(\alpha(R))}$.

Then, we can assume that $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3)) < 2k$. We continue with a similar branching, this time branching on single vertices. In particular, as long as $R \setminus N(V_1 \cup V_2 \cup V_3)$ is nonempty, we select a vertex in it and branch on whether to move it into V_1, V_2, V_3, or S_0. To analyze the size of this branching tree, note that by Lemma 7, when moving a vertex into V_1, V_2, or V_3, the value of $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3))$ decreases by at least one. Therefore, as initially $\alpha(R \setminus N(V_1 \cup V_2 \cup V_3)) < 2k$, any root-to-leaf path of the branching tree contains less than $2k$ edges that correspond to such branches, and therefore any root-leaf path can be characterized by specifying the $< 2k$ indices corresponding to such edges, and then for these indices whether they correspond to V_1, V_2, or V_3. The length of any root-leaf path is at most n because $|R \setminus N(V_1 \cup V_2 \cup V_3)|$ decreases in every branch, and therefore the number of different root-to-leaf paths is at most $n^{2k} 3^{2k} = n^{\mathcal{O}(k)}$, and therefore the total number of nodes in this branching tree is $n^{\mathcal{O}(k)}$.

Therefore, the total size of both of the branching trees put together is $2^{\mathcal{O}(\alpha(R))}n^{\mathcal{O}(k)}$, so our algorithm works by $2^{\mathcal{O}(\alpha(R))}n^{\mathcal{O}(k)}$ applications of Lemma 5, resulting in a $2^{\mathcal{O}(\alpha(R))}n^{\mathcal{O}(k)}$-time algorithm. ◀

Finally, what is left is to reduce the general case to the case where $\alpha(R) = \mathcal{O}(k^2)$. We do not know if this can be done in general, but we manage to do it by using a tree decomposition with independence number $\mathcal{O}(k)$. To this end, we show the following lemma. Given a graph G, a tree decomposition $\mathcal{T}$ of G, and a set $W \subseteq V(G)$, we say that W is *covered* by a set $\mathcal{B}$ of bags of $\mathcal{T}$ if every vertex in W is contained in at least one of the bags in $\mathcal{B}$. The lemma generalizes the well-known fact that any clique W of G, that is, a set with $\alpha(W) = 1$, is covered by a single bag of the tree decomposition.

▶ **Lemma 9.** *Let G be a graph, $\mathcal{T}$ a tree decomposition of G, and W a nonempty set of vertices of G. Then W is covered by a set of at most $2\alpha(W) - 1$ bags of $\mathcal{T}$.*

Proof. We denote $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$. Every edge $ab \in E(T)$ corresponds to a partition $(X_a \cap X_b, C_a, C_b)$ of $V(G)$, where C_a is the set of vertices of $G \setminus (X_a \cap X_b)$ in the bags of $\mathcal{T}$ that are closer to a than b, C_b is the set of vertices of $G \setminus (X_a \cap X_b)$ in the bags of $\mathcal{T}$ that are closer to b than a, and $X_a \cap X_b$ is a (C_a, C_b)-separator.

First, assume that for every edge ab either $C_a \cap W = \emptyset$ or $C_b \cap W = \emptyset$. If both $C_a \cap W = C_b \cap W = \emptyset$, then $W \subseteq X_a \cap X_b$, and we cover W by a single bag X_a (or X_b). Thus we may assume that for every edge exactly one of the sets $C_a \cap W$ and $C_b \cap W$ is nonempty. We orient the edge ab from a towards b if $C_b \cap W \neq \emptyset$, and from b to a if $C_a \cap W \neq \emptyset$. Because T is a tree, there exists a node $t \in V(T)$ such that all edges incident with t are oriented towards t. This implies that X_t covers W because otherwise some edge would be oriented away from t.

Note that if $\alpha(W) = 1$, then W is a clique and indeed for every edge ab either $C_a \cap W = \emptyset$ or $C_b \cap W = \emptyset$. The remaining case is that $\alpha(W) \geq 2$ and that there exists an edge ab such that both $|C_a \cap W| \geq 1$ and $|C_b \cap W| \geq 1$ hold. In this case, we use induction on $\alpha(W)$. Since $X_a \cap X_b$ is a (C_a, C_b)-separator, we have that $\alpha(C_a \cap W) + \alpha(C_b \cap W) \leq \alpha(W)$. Therefore we can take the union of a smallest set of bags covering $C_a \cap W$, a smallest set of bags covering $C_b \cap W$, and the bag X_a. By induction, this set of bags covering W contains at most $2\alpha(W \cap C_a) - 1 + 2\alpha(W \cap C_b) - 1 + 1 \leq 2\alpha(W) - 1$ bags. ◀

With Lemma 9, we can use a tree decomposition $\mathcal{T}$ with independence number $\alpha(\mathcal{T}) = \mathcal{O}(k)$ to 2-approximate the general case of PARTIAL 3-WAY α-SEPARATOR as follows. By Lemma 9, any solution S with $\alpha(S) \leq k$ is covered by at most $2k - 1$ bags of $\mathcal{T}$. Therefore, with $\mathcal{T}$ available (and having $n^{\mathcal{O}(1)}$ bags by standard arguments), we can guess a set of at most $2k - 1$ bags of $\mathcal{T}$ that covers S, and intersect R by the union of these bags, resulting in $\alpha(R) \leq \alpha(\mathcal{T})(2k - 1) = \mathcal{O}(k^2)$. Therefore, we solve the general case by $n^{\mathcal{O}(k)}$ applications of Lemma 8 with $\alpha(R) = \mathcal{O}(k^2)$, resulting in a total time complexity of $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(k)}$. This completes the proof of Theorem 3.

4.2 From separators to balanced separators

In this subsection we show that Theorem 3 can be used to either find certain balanced separators for sets $W \subseteq V(G)$ or to show that the tree-independence number of the graph is large.

To enforce the "balance" condition, we cannot directly enforce that the separator separates a given set W into sets of small independence numbers. (This would result in time complexity exponential in $|W|$ instead of $\alpha(W)$.) Instead, we fix a maximum independent set in W, and enforce that this independent set is separated in a balanced manner. As long as the independent set is large enough, this will enforce that the separator is balanced also with respect to α. The following lemma, which is an adaptation of a well-known lemma given in Graph Minors II [36] is the starting point of this approach.

▶ **Lemma 10.** *Let G be a graph with the tree-independence number at most k and $I \subseteq V(G)$ an independent set. Then there exists a partition (S, C_1, C_2, C_3) of $V(G)$ (where some C_i can be empty) such that S is a (C_1, C_2, C_3)-separator, $\alpha(S) \leq k$, and $|I \cap (C_i \cup C_j)| \geq |I|/2 - k$ for any pair $i, j \in \{1, 2, 3\}$ with $i \neq j$.*

Proof. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree decomposition of G with $\alpha(\mathcal{T}) \leq k$. As in Lemma 9, we introduce the following notation. Every edge $ab \in E(T)$ of T corresponds to a partition $(X_a \cap X_b, C_a, C_b)$ of $V(G)$, where C_a is the set of vertices of $G \setminus (X_a \cap X_b)$ in the bags of $\mathcal{T}$ that are closer to a than b, C_b is the set of vertices of $G \setminus (X_a \cap X_b)$ in the bags of $\mathcal{T}$ that are closer to b than a, and $X_a \cap X_b$ is a (C_a, C_b)-separator. We orient the edge ab from a to b if $|C_b \cap I| > |I|/2$, from b to a if $|C_a \cap I| > |I|/2$, and otherwise arbitrarily. Now, because T is a tree, there exists a node $t \in V(T)$ such that all of its incident edges are oriented towards it. Therefore, for all connected components C of $G \setminus X_t$, we see that $|V(C) \cap I| \leq |I|/2$, as otherwise some edge would be oriented out of t.

As long as the number of such connected components C is at least 4, we can take two of them with the smallest values of $|V(C) \cap I|$ and merge them, in the end arriving at a partition (X_t, C_1, C_2, C_3) of $V(G)$ such that X_t is a (C_1, C_2, C_3)-separator, $\alpha(X_t) \leq k$, and $|I \cap C_i| \leq |I|/2$ for all $i \in \{1, 2, 3\}$. Then, because $|I \cap C_i| \leq |I|/2$, we have that $|I \cap (V(G) \setminus C_i)| \geq |I|/2$. Therefore, since $|I \cap X_t| \leq k$, it follows that $|I \cap (C_j \cup C_\ell)| \geq |I|/2 - k$, where (i, j, ℓ) is any permutation of the set $\{1, 2, 3\}$. ◀

Next we use Lemma 10 together with the separator algorithm of Theorem 3 to design an algorithm for finding α-balanced separators of sets W with $\alpha(W) = 6k$.

▶ **Lemma 11.** *There is an algorithm that for a given graph G, an integer k, a tree decomposition of G with independence number $\mathcal{O}(k)$, and a set $W \subseteq V(G)$ with $\alpha(W) = 6k$, in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ either concludes that the tree-independence number of G is larger than k, or finds a partition (S, C_1, C_2, C_3) of $V(G)$ such that S is a (C_1, C_2, C_3)-separator, $\alpha(S) \leq 2k$, at most one of C_1, C_2, and C_3 is empty, and $\alpha(W \cap C_i) \leq 4k$ for each $i \in \{1, 2, 3\}$.*

Proof. First, we take an arbitrary independent set $I \subseteq W$ of size $|I| = 6k$, which can be found in $n^{\mathcal{O}(k)}$ time. If the tree-independence number of G is at most k, then, by Lemma 10, there exists a partition (S, C_1, C_2, C_3) of $V(G)$ such that S is a (C_1, C_2, C_3)-separator, $\alpha(S) \leq k$, and $|I \cap (C_i \cup C_j)| \geq |I|/2 - k \geq 2k$ for any pair $i, j \in \{1, 2, 3\}$ with $i \neq j$. We guess the intersection of such a partition with I, in particular we guess the partition $(S \cap I, C_1 \cap I, C_2 \cap I, C_3 \cap I)$, immediately enforcing that it satisfies the constraints $|I \cap (C_i \cup C_j)| \geq 2k$ for $i \neq j$.

For each such guess, we use Theorem 3 to either find a $(C_1 \cap I, C_2 \cap I, C_3 \cap I)$-separator with independence number at most $2k$ or to decide that no such separator with independence number at most k exists. The set S of the partition guaranteed by Lemma 10 is indeed a $(C_1 \cap I, C_2 \cap I, C_3 \cap I)$-separator with independence number at most k, so if the algorithm reports for every guess that no such separator exists, we return that G has the independence number larger than k.

Otherwise, for some guess a $(C_1 \cap I, C_2 \cap I, C_3 \cap I)$-separator S' with $\alpha(S') \leq 2k$ is found, and we return the partition (S', C'_1, C'_2, C'_3), where C'_i is the union of the vertex sets of components of $G \setminus S'$ that contain a vertex of $C_i \cap I$, for all $i \in \{1, 2, 3\}$. Because

$|I \cap (C_i \cup C_j)| \geq 2k$ for $i \neq j$, we see that $\alpha(W \cap C_\ell) \leq \alpha(W) - 2k \leq 4k$, where (i, j, ℓ) is an arbitrary permutation of the set $\{1, 2, 3\}$, because otherwise we could use the union of $I \cap (C_i \cup C_j)$ and a maximum independent set in $W \cap C_\ell$ to construct an independent set in W of size larger than $\alpha(W)$. Also, because $|I \cap (C_i \cup C_j)| \geq 2k$ for any pair $i \neq j$, the set C_i cannot be empty for more than one i.

The algorithm works by first finding an independent set of size $6k$ and then using the algorithm of Theorem 3 at most 4^{6k} times, so the total time complexity is $\mathcal{O}(n^{6k}) + 4^{6k} \cdot 2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)} = 2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$. ◀

4.3 Constructing the decomposition

Everything is prepared for the final step of the proof – the algorithm constructing a tree decomposition with independence number at most $8k$ by using the balanced separator algorithm of Lemma 11. Our algorithm constructs a tree decomposition from root to the leaves by maintaining an "interface" W and breaking it with balanced separators. This is a common strategy used for various algorithms for constructing tree decompositions and branch decompositions. In our case, perhaps the largest hurdle in the proof is the analysis that the size of the recursion tree and the constructed decomposition is polynomial in n.

A rooted tree decomposition is a tree decomposition where one node is designated as the root.

▶ **Lemma 12.** *There is an algorithm that for a given graph G, an integer k, a tree decomposition of G with independence number $\mathcal{O}(k)$, and a set $W \subseteq V(G)$ with $\alpha(W) \leq 6k$, in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ either determines that the tree-independence number of G is larger than k or returns a rooted tree decomposition $\mathcal{T}$ of G with independence number at most $8k$ such that W is contained in the root bag of $\mathcal{T}$.*

Proof. The algorithm will be based on recursively constructing the decomposition, using W as the interface in the recursion. First, if $\alpha(G) \leq 6k$, we return the trivial tree decomposition with only one bag $V(G)$. Otherwise, we start by inserting arbitrary vertices of G into W until the condition $\alpha(W) = 6k$ holds.

Then, we apply the algorithm of Lemma 11 to find a partition (S, C_1, C_2, C_3) of $V(G)$ such that S is a (C_1, C_2, C_3)-separator, $\alpha(S) \leq 2k$, $\alpha(W \cap C_i) \leq 4k$ for each $i \in \{1, 2, 3\}$, and at most one of C_1, C_2, C_3 is empty, or to determine that the tree-independence number of G is larger than k, in this case returning no immediately. Then, we construct the tree decomposition recursively as follows: for each $i \in \{1, 2, 3\}$, we recursively use the algorithm with the graph $G_i = G[C_i \cup S]$ and the set $W_i = (C_i \cap W) \cup S$. Let $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ be the obtained tree decompositions and let r_1, r_2, r_3 be their root nodes. We create a new root node r with a bag $X_r = S \cup W$, and connect r_1, r_2, and r_3 as children of r.

Lemma 11 guarantees that $\alpha(C_i \cap W) \leq 4k$ and $\alpha(S) \leq 2k$, and therefore $\alpha(W_i) \leq 6k$. Also, $\alpha(S \cup W) \leq 8k$ because $\alpha(W) \leq 6k$ and $\alpha(S) \leq 2k$. Therefore, the independence number of the constructed tree decomposition is at most $8k$. The constructed tree decomposition satisfies all conditions of tree decompositions: Because S is a separator between C_1, C_2, and C_3, when recursing into the graphs $G_i = G[C_i \cup S]$ for $i \in \{1, 2, 3\}$, the union of the graphs G_1, G_2, and G_3 includes all vertices and edges of G. Therefore, by induction, every vertex and edge will be contained in some bag of the constructed tree decomposition (the base case is $\alpha(G) \leq 6k$). By induction, the decomposition satisfies also the connectivity condition: if a vertex occurs in G_i and G_j for $i \neq j$, then it is in S and therefore in the bag X_r and also in the sets W_i and W_j and therefore in the root bags X_{r_i} and X_{r_j} of $\mathcal{T}_i$ and $\mathcal{T}_j$.

It remains to argue that the size of the recursion tree (and equivalently the size of the decomposition constructed) is $n^{\mathcal{O}(1)}$. First, by the guarantee of Lemma 11 that C_i is empty for at most one $i \in \{1, 2, 3\}$, we have that each G_i has strictly fewer vertices than G, and therefore the constructed tree has height at most n. We say that a constructed node t is a forget node if its bag contains a vertex v so that its parent's bag does not contain v. The number of forget nodes is at most n because a vertex can be forgotten only once in a tree decomposition.

Recall that in the start of each recursive call, on a graph G_i and a subset W_i, we either recognize that $\alpha(G_i) \leq 6k$, creating a leaf node in this case, or add vertices to W_i until $\alpha(W_i) = 6k$. In the latter case, as these added vertices were not initially in W_i, they are not in the bag of the parent node, and therefore the node constructed in such a call will be a forget node if any such vertices are added. Therefore, the new node constructed can be a non-forget non-leaf node only if $\alpha(W_i) = 6k$ already for the initial input W_i. Then, we observe that $\alpha(W_i) = 6k$ can hold for the initial input W_i only if $\alpha(C_i \cap W) = 4k$ did hold for the corresponding component C_i of the parent and the corresponding set W. Therefore, as $\alpha(C_i \cap W) = 4k$ can hold for at most one $i \in \{1, 2, 3\}$, we have that any node can have at most one non-forget non-leaf child node.

It follows that non-forget non-leaf nodes can be decomposed into maximal paths going between a node and its ancestor, and these paths have a length at most n by the height of the tree. Each such path either starts at the root or its highest node is a child of a forget node. Thus, the number of maximal paths of non-forget non-leaf nodes is at most n, and therefore the number of non-forget non-leaf nodes is at most n^2. The number of leaf nodes is at most three times the number of non-leaf nodes, so the total number of nodes is $\mathcal{O}(n^2)$.

Therefore, the algorithm works by $\mathcal{O}(n^2)$ applications of the algorithm of Lemma 11, and therefore its time complexity is $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$. ◀

It remains to observe that by using iterative compression, we can satisfy the requirement of Lemma 12 to have a tree decomposition with independence number $\mathcal{O}(k)$ as an input (in particular, here the independence number will be at most $8k + 1$), and therefore Lemma 12 implies Theorem 1.

In more detail, we order the vertices of G as $v_1, \ldots, v_n$, and iteratively compute tree decompositions with independence number at most $8k$ for induced subgraphs $G[\{v_1, \ldots, v_i\}]$, for increasing values of i. The iterative computation guarantees that when computing the tree decomposition for $G[\{v_1, \ldots, v_i\}]$, we have the tree decomposition for $G[\{v_1, \ldots, v_{i-1}\}]$ with independence number at most $8k$ available, which can be used to obtain a tree decomposition with independence number at most $8k + 1$ of $G[\{v_1, \ldots, v_i\}]$ by adding v_i to each bag to be used as the input tree decomposition. More precisely, we use Lemma 12 to either determine in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ that the tree-independence number of $G[\{v_1, \ldots, v_i\}]$ is larger than k or obtain a rooted tree decomposition $\mathcal{T}$ of $G[\{v_1, \ldots, v_i\}]$ with independence number at most $8k$. As the tree-independence number does not increase when taking induced subgraphs, if for some induced subgraph we conclude that the tree-independence number is larger than k, we can conclude the same holds also for G. Otherwise, after n steps we will have a rooted tree decomposition $\mathcal{T}$ of G with independence number at most $8k$.

5 Hardness of computing tree-independence number

In this section, we complement our main algorithmic result by complexity lower bounds. First, we use the W[1]-hardness of independent set approximation by Lin [30] and the Gap-ETH result of Chalermsook et al. [7] to prove the following theorem.

▶ **Theorem 13** (⋆). *For any constant $c \geq 1$, there is no algorithm running in $f(k) \cdot n^{\mathcal{O}(1)}$ time for a computable function $f(k)$ that, given an n-vertex graph and a positive integer k, can distinguish between the cases* $\mathsf{tree}\text{-}\alpha(G) \leq k$ *and* $\mathsf{tree}\text{-}\alpha(G) > ck$*, unless FPT = W[1]. Moreover, assuming* Gap-ETH*, for any computable function $g(k) \geq k$, there is no algorithm running in $f(k) \cdot n^{o(k)}$ time for a computable function $f(k)$ that, given an n-vertex graph and a positive integer k, can distinguish between the cases* $\mathsf{tree}\text{-}\alpha(G) \leq k$ *and* $\mathsf{tree}\text{-}\alpha(G) > g(k)$.

Theorem 13 implies that it is W[1]-hard to decide whether $\mathsf{tree}\text{-}\alpha(G) \leq k$ for the parameterization by k. However, the problem is, in fact, harder. We prove that it is NP-complete to decide whether $\mathsf{tree}\text{-}\alpha(G) \leq 4$.

▶ **Theorem 2** (⋆). *For every constant $k \geq 4$, it is NP-complete to decide whether* $\mathsf{tree}\text{-}\alpha(G) \leq k$ *for a given graph G.*

Finally, we show that, for every fixed integer $k \geq 3$, deciding if two given vertices of a graph can be separated by removing a set of vertices that induces a graph with independence number at most k is NP-complete. To put this result in perspective, note that the case with $k = 1$ is polynomial since we can compute all clique cutsets in polynomial time using Tarjan's algorithm [39]. We leave open the case with $k = 2$.

▶ **Theorem 14** (⋆). *For every integer $k \geq 3$, it is NP-complete to decide, given a graph H and two distinct vertices $u, v \in V(H)$, if there exists a u,v-separator S such that $\alpha(H[S]) \leq k$.*

6 Conclusion

The main result of our paper is an algorithm that, given an n-vertex graph G and an integer k, in time $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ either outputs a tree decomposition of G with independence number at most $8k$, or concludes that the tree-independence number of G is larger than k. This yields also the same result for computing the minor-matching hypertree-width of a graph [40]. Our results allow to solve in $2^{\mathcal{O}(k^2)} n^{\mathcal{O}(k)}$ time a plethora of problems when the inputs are restricted to graphs of tree-independence number k or minor-matching hypertree-width k [15, 29, 40]. We now show that this result is tight in several aspects.

First, one could ask what is the most general width-parameter defined by a min-max formula over the bags of a tree decomposition (see, e.g. [2, 33]) that allows to solve problems like Maximum Independent Set in polynomial time when bounded? For parameters where the width of a bag depends only on the induced subgraph of the bag, this turns out to be $\mathsf{tree}\text{-}\alpha$. In particular, we recall that Maximum Independent Set is NP-hard on graphs with each edge subdivided twice, but such graphs admit a tree decomposition where one bag is a large independent set, and the induced subgraphs of the other bags are isomorphic to 4-vertex paths. It follows that if the width-measure of a bag is monotone, i.e., it does not increase when taking induced subgraphs, it must be unbounded whenever α is unbounded. In other words, if there would be a width parameter $\mathsf{tree}\text{-}\lambda$ defined as the minimum, over all tree decomposition, of the maximum of $\lambda(G[X_t])$ over the bags X_t of the tree decomposition, where λ is a monotone graph invariant, then either Maximum Independent Set is already NP-hard when λ is a constant, or the parameterization by $\mathsf{tree}\text{-}\alpha$ is more general than the parameterization by $\mathsf{tree}\text{-}\lambda$.

The width parameter $\mathsf{tree}\text{-}\mu$, the minor-matching hypertree-width, escapes this argument because it does not only depend on the subgraphs induced by the bags, but also on the neighborhoods of the bags. In particular, for $\mathsf{tree}\text{-}\mu$ the width of a bag X_t is defined as the maximum cardinality of an induced matching in G whose every edge intersects X_t. A similar example shows that this type of parameters where the width of a bag X_t depends on $G[N[X_t]]$ cannot be generalized much more: If we start from Maximum Independent Set on cubic graphs and subdivide each edge four times, we obtain graphs where Maximum Independent Set is NP-hard, but that admit tree decompositions that contain one large bag X_t such that every connected component of $G[N[X_t]]$ is a 3-vertex path, while for all other bags X_t their closed neighborhood $N[X_t]$ has bounded size.

Further, we remind that our main result is computationally tight. In particular, in Theorem 13, we proved that it is unlikely that there is a $g(k)$-approximation algorithm for the tree-independence number with running time $f(k)n^{o(k)}$, for any computable function g. This shows that the $n^{\Omega(k)}$-factor in the running time is unavoidable up to some reasonable complexity assumptions. For exact computation of the tree-independence number, we proved in Theorem 2 that it is NP-complete to decide whether $\mathsf{tree}\text{-}\alpha(G) \leq k$ for every constant $k \geq 4$. Since $\mathsf{tree}\text{-}\alpha(G) = 1$ if and only if G is a chordal graph, Theorem 2 leads to the question about the complexity of deciding whether the tree-independence number is at most k for $k = 2$ and 3. In Theorem 14, we demonstrated that for every fixed integer $k \geq 3$, deciding if two given vertices of a graph can be separated by removing a set of vertices that induces a graph with independence number at most k is NP-complete. This result indicates that it may be already NP-complete to decide whether $\mathsf{tree}\text{-}\alpha(G) \leq 3$. We hesitate to state any conjecture for the case $k = 2$.

The final question is about the place of computing the tree-independence number in the polynomial hierarchy. For a fixed k, deciding whether $\mathsf{tree}\text{-}\alpha(G) \leq k$ is in NP. However, when k is a part of the input, the problem is naturally placed in the class Σ_2^P on the second level of the polynomial hierarchy. Is the problem Σ_2^P-complete?

References

1 Tara Abrishami, Bogdan Alecu, Maria Chudnovsky, Sepehr Hajebi, Sophie Spirkl, and Kristina Vušković. Tree independence number i. (even hole, diamond, pyramid)-free graphs, 2024. arXiv:2305.16258.

2 Isolde Adler. *Width functions for hypertree decompositions.* PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2006.

3 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41(1):153–180, 1994.

4 M. Bíró, M. Hujter, and Zs. Tuza. Precoloring extension. I. Interval graphs. *Discrete Math.*, 100(1-3):267–279, 1992. doi:10.1016/0012-365X(92)90646-W.

5 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.

6 Hans L. Bodlaender, Jens Gustedt, and Jan Arne Telle. Linear-time register allocation for a fixed number of registers. In Howard J. Karloff, editor, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*, pages 574–583. ACM/SIAM, 1998. URL: http://dl.acm.org/citation.cfm?id=314613.314994.

7 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From gap-exponential time hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more. *SIAM J. Comput.*, 49(4):772–810, 2020. doi:10.1137/18M1166869.

8 Steven Chaplick, Fedor V. Fomin, Petr A. Golovach, Dušan Knop, and Peter Zeman. Kernelization of graph Hamiltonicity: proper H-graphs. *SIAM J. Discrete Math.*, 35(2):840–892, 2021. doi:10.1137/19M1299001.

9 Steven Chaplick, Martin Töpfer, Jan Voborník, and Peter Zeman. On H-topological intersection graphs. *Algorithmica*, 83(11):3281–3318, 2021. doi:10.1007/s00453-021-00846-3.

10 Steven Chaplick and Peter Zeman. Combinatorial problems on H-graphs. *Electron. Notes Discret. Math.*, 61:223–229, 2017. doi:10.1016/j.endm.2017.06.042.

11 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discret. Math.*, 86(1-3):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

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

13 Clément Dallard, Fedor V. Fomin, Petr A. Golovach, Tuukka Korhonen, and Martin Milanič. Computing tree decompositions with small independence number. *CoRR*, abs/2207.09993, 2022. arXiv:2207.09993.

14 Clément Dallard, Martin Milanič, and Kenny Štorgel. Treewidth versus clique number in graph classes with a forbidden structure. In Isolde Adler and Haiko Müller, editors, *Graph-Theoretic Concepts in Computer Science – 46th International Workshop, WG 2020, Leeds, UK, June 24-26, 2020, Revised Selected Papers*, volume 12301 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2020. doi:10.1007/978-3-030-60440-0_8.

15 Clément Dallard, Martin Milanič, and Kenny Štorgel. Treewidth versus clique number. II. tree-independence number. *J. Comb. Theory, Ser. B*, 164:404–442, 2024. doi:10.1016/J.JCTB.2023.10.006.

16 Clément Dallard, Martin Milanič, and Kenny Štorgel. Treewidth versus clique number. III. Tree-independence number of graphs with a forbidden structure. *Journal of Combinatorial Theory, Series B*, 167:338–391, 2024. doi:10.1016/j.jctb.2024.03.005.

17 Clément Dallard, Martin Milanič, and Kenny Štorgel. Treewidth versus clique number. I. Graph classes with a forbidden structure. *SIAM J. Discrete Math.*, 35(4):2618–2646, 2021. doi:10.1137/20M1352119.

18 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for exponential-time-hypothesis-tight algorithms and lower bounds in geometric intersection graphs. *SIAM J. Comput.*, 49(6):1291–1331, 2020. doi:10.1137/20M1320870.

19 Erik D. Demaine, Fedor V. Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.

20 Irit Dinur. Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover. *Electron. Colloquium Comput. Complex.*, TR16-128, 2016. URL: https://eccc.weizmann.ac.il/report/2016/128, arXiv:TR16-128.

21 Fedor V. Fomin and Petr A. Golovach. Subexponential parameterized algorithms and kernelization on almost chordal graphs. *Algorithmica*, 83(7):2170–2214, 2021. doi:10.1007/s00453-021-00822-x.

22 Fedor V. Fomin, Petr A. Golovach, and Jean-Florent Raymond. On the tractability of optimization problems on H-graphs. *Algorithmica*, 82(9):2432–2473, 2020. doi:10.1007/s00453-020-00692-9.

23 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Excluded grid minors and efficient polynomial-time approximation schemes. *J. ACM*, 65(2):10:1–10:44, 2018. doi:10.1145/3154833.

24 Esther Galby, Andrea Munaro, and Shizhou Yang. Polynomial-time approximation schemes for independent packing problems on fractionally tree-independence-number-fragile graphs. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.SoCG.2023.34.

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

26 Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23(4):613–632, 2003. doi:10.1007/s00493-003-0037-9.

27 Petr Hlinený and Jan Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discret. Math.*, 229(1-3):101–124, 2001. doi:10.1016/S0012-365X(00)00204-1.

28 Ashwin Jacob, Fahad Panolan, Venkatesh Raman, and Vibha Sahlot. Structural parameterizations with modulator oblivion. *Algorithmica*, 84(8):2335–2357, 2022. doi:10.1007/s00453-022-00971-7.

29 Paloma T. Lima, Martin Milanič, Peter Muršič, Karolina Okrasa, Paweł Rzążewski, and Kenny Štorgel. Tree decompositions meet induced matchings: beyond Max Weight Independent Set. arXiv:2402.15834, 2024. doi:10.48550/arXiv.2402.15834.

30 Bingkai Lin. Constant approximating k-clique is W[1]-hard. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1749–1756. ACM, 2021. doi:10.1145/3406325.3451016.

31 Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. Subexponential parameterized algorithms on disk graphs (extended abstract). In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2031. SIAM, 2022. doi:10.1137/1.9781611977073.80.

32 Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense CSPs. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 78:1–78:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICS.ICALP.2017.78.

33 Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):Art. 42, 51, 2013. doi:10.1145/2535926.

34 Michal Pilipczuk. Computing tree decompositions. 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 189–213. Springer, 2020. doi:10.1007/978-3-030-42071-0_14.

35 Vijay Raghavan and Jeremy P. Spinrad. Robust algorithms for restricted domains. *J. Algorithms*, 48(1):160–172, 2003. doi:10.1016/S0196-6774(03)00048-8.

36 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.

37 Neil Robertson and Paul 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.

38 Konstantin Skodinis. Efficient analysis of graphs with small minimal separators. In Peter Widmayer, Gabriele Neyer, and Stephan J. Eidenbenz, editors, *Graph-Theoretic Concepts in Computer Science, 25th International Workshop, WG '99, Ascona, Switzerland, June 17-19, 1999, Proceedings*, volume 1665 of *Lecture Notes in Computer Science*, pages 155–166. Springer, 1999. doi:10.1007/3-540-46784-X_16.

39 Robert E. Tarjan. Decomposition by clique separators. *Discrete Math.*, 55(2):221–232, 1985. doi:10.1016/0012-365X(85)90051-2.

40 Nikola Yolov. Minor-matching hypertree width. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 219–233. SIAM, 2018. doi:10.1137/1.9781611975031.16.