

Computing Twin-Width via Treedepth and Vertex Integrity

Robert Ganian  

Algorithms and Complexity Group, TU Wien, Austria

Mathis Rocton  

Algorithms and Complexity Group, TU Wien, Austria

Abstract

Twin-width is a graph parameter that has become central to explaining the fixed-parameter tractability of first-order model checking across many graph classes. Despite its algorithmic importance, computing twin-width remains poorly understood: even recognizing graphs of twin-width at most four is NP-hard, and no fixed-parameter approximations parameterized by twin-width itself are known. A recent approach towards breaking this barrier focuses on first developing fixed-parameter algorithms for computing or approximating twin-width under parameterizations distinct from twin-width.

Our first result establishes that approximating twin-width is fixed-parameter tractable when parameterized by treedepth, thereby breaking the long-standing barrier that all previous tractable parameterizations were based on deletion distance. The proof proceeds via oriented twin-width, yielding the first constructive evidence that this variant may be easier to handle algorithmically. As our second main result, we show that computing twin-width exactly is fixed-parameter tractable with respect to vertex integrity. This constitutes the first non-trivial parameterized algorithm for computing optimal contraction sequences.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases twin-width, fixed-parameter algorithms, treedepth, vertex integrity

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.42

Funding *Robert Ganian*: Robert Ganian acknowledges support by the FWF and WWTF Science Funds (FWF projects 10.55776/Y1329 and 10.55776/COE12, WWTF project ICT22-029).

Mathis Rocton: Mathis Rocton acknowledges support by the European Union’s Horizon 2020 research and innovation COFUND programme (LogiCS@TUWien, grant agreement No 101034440), and the FWF Science Fund (FWF project Y1329).



1 Introduction

Twin-width is a relatively recent graph-theoretic parameter that has both emerged from and stimulated a series of breakthroughs in algorithmic model theory [10, 11, 12, 14, 15]. It holds the promise of offering a unifying explanation for why first-order logic model-checking is fixed-parameter tractable on a wide range of graph classes which, until recently, were viewed as isolated “islands of tractability.” Examples include graphs of bounded rank-width, proper minor-closed graphs, bounded-width posets [3], map graphs [16] as well as a number of other specialized graph classes [4, 25].

Although twin-width is related to other parameters such as path-width, clique-width and rank-width [15], it stands apart in one crucial respect: no efficient algorithms are known for computing it. In fact, already deciding whether a graph has twin-width 4 is NP-hard [6]. This poses a serious challenge, as essentially all known algorithms that exploit twin-width require access to a *contraction sequence*, which plays a role analogous to that of decompositions for classical parameters such as treewidth [39] and rank-width [36]. Intuitively, a contraction sequence of width t – which exists whenever G has twin-width at most t – is a sequence \mathcal{C}



of contractions of (not necessarily adjacent) vertex pairs such that, at each step of \mathcal{C} , every vertex v has at most t neighbors with an ancestor not adjacent to some ancestor of v ; see Section 2 for formal details.

The NP-hardness of recognizing graphs of twin-width at most 4 [6] rules out both fixed-parameter and XP algorithms for computing optimal contraction sequences when parameterized by twin-width itself. A natural workaround would be to design a fixed-parameter algorithm (parameterized by the twin-width t) that computes an *approximate* contraction sequence, i.e., one of width $f(t)$ for a computable function f . From a complexity-theoretic standpoint, such a result would be nearly as powerful as computing the exact twin-width as it would still enable fixed-parameter tractability of first-order model checking. In fact, two decades ago it was the discovery of an analogous approximate decomposition-computing result [37] which opened the door to the algorithmic applications of clique-width.

Unfortunately, finding such an algorithm for twin-width has proven to be highly challenging, and it remains unclear whether it is even possible. Indeed, determining whether twin-width admits any fixed-parameter approximation (for arbitrary computable f) is arguably the central open problem in current research on the parameter. In their recent work, Balabán, Ganian and Rocton [1] approached this question by relaxing the runtime requirements and asked whether one could obtain an $f(t)$ -approximation for twin-width via a fixed-parameter algorithm parameterized by a graph measure other than twin-width itself. As a first step, they developed a fixed-parameter algorithm that computes a contraction sequence of width at most $t + 1$, parameterized by the *feedback edge number* of the graph [1], i.e., by the edge deletion distance to a forest. In a follow-up work, the same authors obtained a fixed-parameter approximation algorithm which computes a contraction sequence of width at most $2t$ and is parameterized by the *vertex integrity* of the graph [2]. Vertex integrity can be roughly understood as the deletion distance to a collection of small components, and the aforementioned 2-approximation algorithm can be seen as a way of lifting the trivial fixed-parameter algorithm for computing optimal contraction sequences parameterized by the vertex cover number of the graph [1, 2].

Results. In this article, we push beyond the state of the art for computing twin-width in two directions. As our first result, we lift the fixed-parameter approximability of twin-width from the vertex integrity parameterization to treedepth – a well-studied parameter that has fundamental ties to the theory of graph sparsity [35]. In particular, we prove:

► **Main Result 1.** *Approximating twin-width is FPT w.r.t. the treedepth of the input graph.*

Even though the algorithm underlying Main Result 1 (formalized in Theorem 20) provides approximation guarantees only in terms of a superexponential function of the twin-width, we believe it to be a major step forward in our conceptual understanding. Indeed, while all previous parameterizations which supported efficient approximation algorithms for twin-width were based on deletion distance, twin-width itself – as well as major structural parameters such as treewidth and clique-width – are *decompositional* in nature. Our result thus marks the first time we are able to break the barrier towards decompositional parameters: having bounded treedepth is typically witnessed via a certain type of decomposition (which resembles a bounded-depth tree), but cannot be expressed via a constant number of deletion operations¹.

¹ We remark that both treedepth and treewidth admit characterizations that rely on vertex deletion operations – however, in both cases the number of such operations is not bounded by the parameter.

A further noteworthy property of Theorem 20 is that the proof does not target the computation of twin-width directly. Instead, it relies on reduction rules which maintain a 2-approximate bound on the so-called *oriented twin-width* – a variant of twin-width where contraction sequences only capture information about edges of the original graph in a one-sided way. Oriented twin-width has been proposed and studied already in the pioneering series of works that introduced twin-width [15]; it was known to be functionally equivalent to twin-width [15, Theorem 4.1], and in Section 3 we make this relationship both explicit and constructive. It was suspected that oriented twin-width could be easier to compute, and Theorem 20 yields concrete substance to that claim: the algorithm achieves a 2-approximation on the oriented twin-width of the input graph, but it is entirely unclear how to obtain any such bound for “vanilla” twin-width.

While our first result relies on allowing for a potentially large approximative error in the twin-width, the second one makes do without this entirely:

► **Main Result 2.** *Computing twin-width is FPT w.r.t. the vertex integrity of the input graph.*

Main Result 2 (formalized in Theorem 33) is the first non-trivial parameterized algorithm for computing optimal contraction sequences, and shows that the NP-hardness of the problem can be overcome at least under restrictive parameterizations. Moreover, while its running time guarantees are weak in practice, the underlying algorithm relies on the performance of reduction rules which are easy to implement and guaranteed to preserve the twin-width of the input graph. We remark that both algorithms obtained in this work are deterministic, constructive and rely exclusively on computable functions.

Technical Contributions. On a high level, the proof of Theorem 20 relies on establishing that every sufficiently large graph of bounded treedepth must have a “redundant” subgraph H – that is, a subgraph which we can find and safely delete from the instance. Recursively applying such a rule will eventually reduce the graph to a problem kernel [18], at which point one may compute a contraction sequence via brute force or any other known algorithm. It is well-known that deleting vertices cannot increase the twin-width of a graph, but for correctness it is also necessary to prove the converse: that by deleting H , we have not accidentally reduced the twin-width of G . In other words, a redundant subgraph must have the property that it can be reinserted into the graph without increasing the twin-width.

The main obstacle towards approximating twin-width when parameterized by treedepth is that – unlike many problems where the above recursive deletion technique has been successfully applied – reinserting H may require the contraction sequence to have a short subsequence where the twin-width is elevated. In particular, the *red degree* (see Section 2) can exceed the twin-width bound by a factor of at most 2, and this can happen both for vertices in H and vertices outside of H . For a single reinsertion step this would not be an issue, but the algorithm must be able to perform deletion multiple times, which would then lead to the error gradually snowballing to a factor of 4, 8, and so forth. The crucial insight here is that if we instead consider contraction sequences for oriented twin-width, the red degree exceeds the twin-width bound only for vertices inside H ; in turn, this allows us to isolate the approximation error in clearly delimited and pairwise disjoint parts of the contraction sequence.

For Theorem 33, we build on the very recently introduced Ramsey Pruning technique [19]. The core idea behind that technique is that instead of repeatedly arguing that a single redundant subgraph can be safely reinserted if it is deleted, we use Ramsey-type arguments to argue that every contraction sequence S for G must contain a carefully selected subgraph

Q which induces “guiding subsequence” S' – a contraction sequence for Q that is highly structured in a precisely defined way. Once defined, the properties of S' will allow us to argue that every solution on Q can be lifted to a contraction sequence of width t for an infinite set of supergraphs of Q which also includes G . Thus, while we do not directly prove the safeness of performing any single deletion operation, the proof technique guarantees that the existence of a solution (in this case, a contraction sequence of width t) is preserved between the first and final graph in the sequence of reduction rules.

The main difficulty that arises when trying to implement the above strategy in the setting of twin-width is that, essentially, the properties of S' one can guarantee by invoking the analogous Ramsey-type arguments as in the preceding work [19] are too weak. There, the core idea was to capture pairwise relationships between certain subgraphs of G via color-coded edges in an auxiliary graph, and then employ Ramsey theory to guarantee the existence of a monochromatic subclique. Unfortunately, pairwise relationships do not capture sufficient information to guarantee the ability to expand S' into a solution for G (or a supergraph thereof). We solve this by showing that maintaining information about interactions between triplets of certain subgraphs in G does suffice – a complication which necessitates the use of Ramsey hypergraph theory instead of the classical bounds used in [19].

Related Work. Beyond the computation of twin-width and its associated contraction sequences, a substantial body of work has focused on fixed-parameter algorithms for computing a structural graph parameter A when parameterized by graph parameters different from A . The overarching goal of this research direction is to deepen our understanding of the fundamental task of computing the target parameter A . Prominent examples include fixed-parameter algorithms for treedepth parameterized by the vertex cover number [32], treewidth parameterized by the feedback vertex number [9], MIM-width parameterized by the feedback edge number and other parameters [24], as well as directed feedback vertex number parameterized by the undirected feedback vertex number [7].

Treedepth and vertex integrity have also served as effective parameterizations for developing algorithms for a variety of hard problems [5, 29, 33, 8, 30, 27, 31]. Moreover, vertex integrity has been studied under several asymptotically equivalent notions in the literature, including the *fracture number* [23, 28] and *starwidth* [40].

2 Preliminaries

For integers i and j , we let $[i, j] := \{n \in \mathbb{N} \mid i \leq n \leq j\}$ and $[i] := [1, i]$. We assume familiarity with basic concepts in graph theory [20] and parameterized algorithmics [21, 18]. When H is an induced subgraph of G , we denote it by $H \subseteq G$. Given vertex sets X and U , we will use $G[X]$ to denote the graph induced on X and $G - U$ to denote the graph $G[V(G) \setminus U]$. A vertex v of a rooted tree T is a *descendant* of a vertex w if w occurs on the root-to- v path in T ; we then call w an *ancestor* of v . We recall that the number of non-isomorphic graphs of size up to n can be upper-bounded by $n \cdot 2^{n^2}$. To express some of our bounds, we will occasionally use the Knuth notation $\uparrow\uparrow$ where for an integer z , $2 \uparrow\uparrow z$ represents an exponential tower of 2's of height z .

Treedepth. A *treedepth decomposition* of a graph G is a pair (T, f) , where T is a rooted forest and $f : V(G) \rightarrow V(T)$ is a bijective function such that for each $\{u, v\} \in E(G)$, either $f(u)$ is a descendant of $f(v)$ or $f(v)$ is a descendant of $f(u)$. The *depth* of the treedepth decomposition is the number of vertices in the longest root-to-leaf path in T ; or ease of

presentation, w.l.o.g. we will assume that f is just an identity. The *treedepth* of a graph G , denoted by $\text{td}(G)$, is the minimum over the depths of all possible treedepth decompositions of G . When G is connected, T is a tree. For any vertex u of T , we denote by T_u the subtree of T rooted at u , and X_u the subgraph $G[V(T_u)]$.

A treedepth decomposition T is *nice* if, for each node $u \in V(T)$ and each child w of u in T , X_w forms a connected component of $X_u \setminus \{u\}$ – in particular, there is a one-to-one correspondence between the subtrees rooted at the children of u in T and the set Comp_u of connected components in $X_u \setminus \{u\}$. It is known that every treedepth decomposition can be transformed into a nice one with a smaller or equal depth via a simple rearrangement argument [38], and that a nice treedepth decomposition of depth $\text{td}(G)$ can be computed in time $2^{\mathcal{O}(\text{td}(G)^2)} \cdot |V(G)|$ [38], see also the recent work [34] on the topic.

For ease of presentation, we say that a subgraph H of G *appears in* T if there exist a *seed* vertex u in T such that $H = X_u$. Moreover, we call two subgraphs X_a, X_b appearing in T *siblings* if a and b are siblings in T , i.e., have the same parent.

Vertex Integrity. A graph G has *vertex integrity* $\text{vi}(G) = p$ if p is the smallest integer with the following property: G contains a vertex set S such that for each connected component H of $G - S$, $|V(H) \cup S| \leq p$. One may observe that the vertex integrity is upper-bounded by the size of a minimum vertex cover in the graph (i.e., the vertex cover number) plus one. The vertex integrity of an n -vertex graph along with a corresponding partition into S and $\mathcal{C} = G - S$ can be computed in time $\mathcal{O}(p^{p+1} \cdot n)$ [22].

Hypergraph Ramsey Bounds. We will employ a known generalization of Ramsey’s classical theorem to edge-colored hypergraphs in order to argue the existence of certain structures in our correctness proof. In particular, the following fact follows directly from the application of Erdős’ and Rado’s improved bound on the Ramsey numbers for multicolored hypergraphs [26, Theorem 1]; see also the recent work on the topic [17] for an overview.

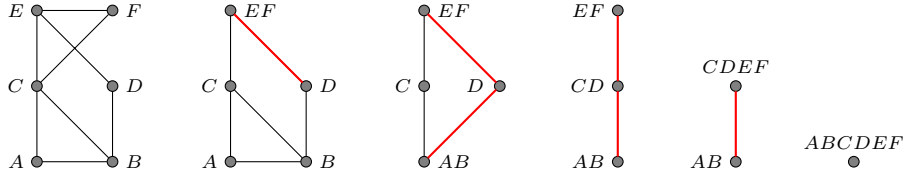
► **Fact 1.** *Let H be a complete hypergraph whose hyperedges all have size 3 and are mapped to single colors from $[L]$, and let t be a positive integer. If $|V(H)| \geq L^{L^{2L \cdot t}}$, then H contains an induced t -vertex subhypergraph H' such that all hyperedges fully contained in H' have the same color.*

Twin-Width. Our notation for twin-width follows the conventions introduced in the recent related works aimed at computing twin-width under more restrictive parameterizations [1, 2]. A *trigraph* G is a graph whose edge set is partitioned into a set of *black* and *red* edges. The set of red edges is denoted $R(G)$, and the set of black edges $E(G)$. The *black* (resp. *red*) *degree* of $u \in V(G)$ is the number of black (resp. red) edges incident to u in G . We extend graph-theoretic terminology to trigraphs by ignoring the colors of edges; for example, the degree of u in G is the sum of its black and red degrees.

Given a trigraph G , a *contraction* of two distinct vertices $u, v \in V(G)$ is the operation which produces a new trigraph by (1) removing u, v and adding a new vertex w , (2) adding a black edge wx for each $x \in V(G)$ such that $xu, xv \in E(G)$, and (3) adding a red edge wy for each $y \in V(G)$ such that $yu \in R(G)$, or $yv \in R(G)$, or y has a black edge to either v or u (but not both). For an integer p , a sequence $\mathcal{C} = (G = G_1, \dots, G_p)$ is a *partial contraction sequence* of G if it is a sequence of trigraphs such that for all $i \in [p - 1]$, G_{i+1} is obtained from G_i by contracting two vertices; if $p = |V(G)|$ then G_p is the single-vertex graph and we call \mathcal{C} a *contraction sequence*. We call a contiguous subsequence of \mathcal{C} a *segment*. The *width* of a (partial) contraction sequence \mathcal{C} is the maximum red degree over all vertices in

all trigraphs in \mathcal{C} . The *twin-width* of G , denoted $\text{tw}(G)$, is the minimum width of any contraction sequence of G , and a contraction sequence of width $\text{tw}(G)$ is called *optimal*. An example of a contraction sequence is provided in Figure 1.

► **Fact 2** (e.g., Observation 6 in [1]). *An optimal contraction sequence of an n -vertex graph can be computed in time $2^{\mathcal{O}(n \cdot \log n)}$.*



■ **Figure 1** A contraction sequence of width 2 for the leftmost graph, consisting of 6 trigraphs.

Let us now fix a contraction sequence $\mathcal{C} = (G = G_1, \dots, G_n)$. For each $i \in [n]$, we associate each vertex $u \in V(G_i)$ with a set $\beta(u, i) \subseteq V(G)$, called the *bag* of u , which contains all vertices contracted into u . Formally, we define the bags as follows:

- for each $u \in V(G)$, $\beta(u, 1) := \{u\}$;
- for $i \in [n - 1]$, if w is the new vertex in G_{i+1} obtained by contracting u and v , then $\beta(w, i + 1) := \beta(u, i) \cup \beta(v, i)$; otherwise, $\beta(w, i + 1) := \beta(w, i)$.

Note that if a vertex u appears in multiple trigraphs in \mathcal{C} , then its bag is the same in all of them, and so we may denote the bag of u simply by $\beta(u)$. Let us fix $i, j \in [n]$, $i \leq j$. If $u \in V(G_i)$, $v \in V(G_j)$, and $\beta(u) \subseteq \beta(v)$, then we say that u is a \mathcal{C} -*ancestor* of v in G_i and v is the \mathcal{C} -*descendant* of u in G_j (clearly, the \mathcal{C} -descendant is unique). If H is an induced subtrigraph of G_i , then $u \in V(G_j)$ is a \mathcal{C} -*descendant* of H if it is a \mathcal{C} -descendant of at least one vertex of H . A contraction of $u, v \in V(G_j)$ into $w \in V(G_{j+1})$ *involves* $w \in V(G_i)$ if w is a \mathcal{C} -ancestor of w . Given a contraction sequence \mathcal{C} of G and vertex subset $Z \subseteq V(G)$, we let $\mathcal{C}[Z]$ denote the *restriction* of \mathcal{C} to Z , i.e., the contraction sequence obtained from \mathcal{C} by omitting each step that involves vertices outside of Z ; note that $\mathcal{C}[Z]$ is a contraction sequence of $G[Z]$ whose width is upper-bounded by the width of \mathcal{C} .

The twin-width of a graph with vertex integrity p is upper-bounded by p . Indeed, given a vertex set S such that for each connected component H of $G - S$, $|V(H) \cup S| \leq p$, we can construct a contraction sequence of width p by processing components in an arbitrary order as follows. We contract the first component H_1 into a single vertex v_1 (during which the red degree never exceeds p). Afterwards, for each unprocessed component i , $i \geq 2$, we contract H_i into a single vertex (during which the red degree never exceeds p either) and contract this vertex with v_{i-1} to produce v_i .

Oriented Twin-Width. The oriented variant of twin-width is defined using the same concept of contraction sequences as twin-width, and hence we opt to use primarily shared terminology (as opposed to the homogeneity-based notation of [15]). Intuitively, in oriented twin-width red edges only represent discrepancies between the corresponding bags in a one-directional manner; this will require the individual trigraphs in a contraction sequence to depend not only on the previous trigraph in the sequence but also on the original graph. We formalize below.

An *oriented trigraph* is a graph whose edge set is partitioned into a set of black undirected edges and red directed edges (arcs). Given an initial graph G and an oriented trigraph G_{i-1} each of whose vertices $z \in V(G_{i-1})$ is associated with a bag $\beta(z, i - 1) \subseteq V(G)$, an

oriented contraction of two distinct vertices $u, v \in V(G_{i-1})$ is the operation which produces a new oriented trigraph G_i as follows. First, we remove u, v and add a new vertex w where $\beta(w, i) = \beta(u, i-1) \cup \beta(v, i-1)$; for all other vertices, the bags remain the same and so do the edges and arcs not incident to w . For each $a \in V(G_i)$ we add a black edge aw if every vertex of $\beta(a, i)$ is adjacent to every vertex of $\beta(w, i)$ in G . For each $b \in V(G_i)$ we add a red arc (w, b) if there exist $w_1, w_2 \in \beta(w, i)$ and $b_0 \in \beta(b, i)$ such that $w_1 b_0 \in E(G)$ and $w_2 b_0 \notin E(G)$. Similarly, for each $c \in V(G_i)$ we add a red arc (c, w) if there exist $c_1, c_2 \in \beta(c, i)$ and $w_0 \in \beta(w, i)$ such that $c_1 w_0 \in E(G)$ and $c_2 w_0 \notin E(G)$. Notice that an oriented contraction cannot increase the red out-degree of any vertex in $V(G_i) \setminus \{w\}$.

An *oriented contraction sequence* is then defined analogously as a contraction sequence, but using oriented contractions. The *oriented width* of an oriented contraction sequence \mathcal{C} the maximum red out-degree over all vertices in all trigraphs in \mathcal{C} . While the oriented twin-width of a graph can be smaller than its twin-width, by comparing the definitions of the two notions of contraction sequences we immediately obtain:

► **Fact 3** ([15]). *The oriented twin-width of every oriented contraction sequence \mathcal{C} is upper-bounded by the width of the contraction sequence \mathcal{C}' obtained by the same sequence of vertex pair contractions.*

The *oriented twin-width* of a graph G , denoted $\text{otww}(G)$, is the minimum oriented width of any oriented contraction sequence of G .

► **Remark.** Since Section 4 will deal exclusively with oriented contraction sequences and oriented width, for brevity we omit the adjective “oriented” in that section.

3 Translating Twin-Width and Oriented Twin-Width

Bonnet, Kim, Reinald and Thomassé established the functional equivalence between twin-width and its oriented variant by showing:

► **Fact 4** (Proof of Theorem 4.1 in [15]). *For every graph G , $\text{otww}(G) \leq \text{tw}(G) \leq 2^{2^{\text{otww}(G)}}$.*

Unfortunately, the proof of the above statement is not constructive – it does not provide any algorithm for converting an oriented contraction sequence \mathcal{C} of oriented width α into a contraction sequence \mathcal{C}' of width bounded by a function of α . In order to construct the contraction sequences for Main Result 1, we need to revisit the relationship between the two parameters and provide an efficient conversion algorithm.

First, we provide some background for the terminology required solely in this section. Given a total order σ over $V(G)$, the *ordered adjacency matrix* $M(G, \sigma)$ is the adjacency matrix of G where the rows and columns both appear in the order σ . The mixed number is a numerical parameter of ordered adjacency matrices which is known to be related to twin-width, but we will not need its definition for our purposes. Similarly, a notion of *twin-width* has also been defined over ordered adjacency matrices but we will not need its definition for our purposes. We say that a (oriented) contraction sequence in \mathcal{C} of G *respects* σ if the vertices in every bag in every trigraph of \mathcal{C} occur consecutively in σ – in other words, for each pair of vertices a, b occurring in some bag $\beta(\cdot, \cdot)$, there cannot exist any $c \notin \beta(\cdot, \cdot)$ such that c occurs between a and b in σ . The following result can be obtained by adapting the proof of [15, Theorem 4.1].

► **Lemma 5.** *Given a graph G with an oriented contraction sequence \mathcal{C} of oriented width at most k , we can compute in polynomial time an ordering σ on $V(G)$ such that $(M(G, \sigma))$ has mixed number at most $2k + 3$.*

The first part of the following statement follows directly from the second part of the Grid Minor Theorem for Twin-Width [16, Theorem 10], see also [10, Theorem 2.2], when setting $t := 2k + 3$. The second part follows from [16, Theorem 5.8], which translates the aforementioned grid minor theorem to graphs. We note that the statement of [16, Theorem 5.8] omits the fact that the constructed contraction sequence respects σ , but this follows directly from the proof.

► **Fact 6.** *Every ordered matrix $(M(G), \sigma)$ of mixed number $2k + 3$ has twin-width at most $2^{2^{\mathcal{O}(k)}}$. Moreover, there exists a contraction sequence of G which respects σ and has width at most $2^{2^{\mathcal{O}(k)}}$.*

We remark that $100 \cdot 2^{2^{12k}}$ can serve as a non-tight but explicit bound for the above terms involving k . Next, we recall the fixed-parameter tractability of approximating contraction sequences with respect to a provided total order of the vertex set [13, Theorem 7].

► **Fact 7.** *There is a fixed-parameter algorithm that takes as input a graph G with a total order σ of its vertices, and outputs a contraction sequence of G which respects σ of width at most $2^{\mathcal{O}(p^4)}$. Here, the parameter p is the minimum width among all contraction sequences of G which respect σ .*

By chaining the above arguments, we obtain:

► **Lemma 8.** *There is a fixed-parameter algorithm that takes as input a graph G together with a contraction sequence \mathcal{C} of oriented width k (the parameter) and outputs a contraction sequence \mathcal{C}' of width at most $2^{2^{\mathcal{O}(k)}}$. Moreover, if $k \leq f(\text{otww}(G))$ for some computable non-decreasing function f , then \mathcal{C}' has width at most $2^{2^{\mathcal{O}(f(\text{otww}(G)))}}$.*

4 A Fixed-Parameter Algorithm Parameterized by Treedepth

In this section, we design an FPT 2-approximation algorithm for computing oriented twin-width when parameterized by the treedepth, see Theorem 19. In combination with Lemma 8, this will in turn imply Main Result 1 as formalized in Theorem 20.

4.1 Initial Setup and Overview

For the following, it will be useful to recall the definition of treedepth presented in Section 2.

Let G be an arbitrary graph with treedepth p and T a nice treedepth decomposition of G of depth p . Note that G and T have the same set of vertices. We assume without loss of generality that the input graph G is connected, as the oriented twin-width of a graph is the maximum oriented twin-width of its connected components; connectivity will then be preserved throughout our reduction rules. For a given vertex u in T , we now define a notion of “component-types” which intuitively captures the equivalence between components which exhibit the same outside connections and internal structure.

► **Definition 9.** *We say that two graphs $H_0, H_1 \in \text{Comp}_u$ appearing in T are u -twin-blocks, denoted $H_0 \sim_u H_1$, if there exists a canonical isomorphism α from H_0 to H_1 such that for each vertex $v \in V(H_0)$ and each $w \in V(G \setminus (H_0 + H_1))$, $vw \in E(G)$ if and only if $\alpha(v)w \in E(G)$. Clearly, \sim_u is an equivalence relation.*

When u is clear from context, we simply refer to H_0 and H_1 as *twin-blocks* ($H_0 \sim H_1$) and note that \sim can be tested in time $\mathcal{O}(|H_0|^{|H_0|} \cdot p)$ via isomorphism testing procedures. Moreover, we lift α to sets of vertices by simply setting $\alpha(S) = \{\alpha(s) \mid s \in S\}$.

The core of our approach lies in establishing a reduction rule which takes a graph G with its treedepth decomposition T and gradually deletes subgraphs from G until we obtain a subgraph whose size is upper-bounded by a function of $\text{td}(G)$.

► **Definition 10.** A vertex u at depth d in T is processed if: $T_u \setminus \{u\}$ contains only processed vertices, and $|\text{Comp}_u| \leq g(d, p)$, where we set $g(p, p) = 2^p$ and, for any $i \in [0, p - 1]$, $g(i, p) := 2^{g(i+1, p)^{op}}$.

► **Lemma 11.** If a subtree T_u of T is rooted at depth d and u is processed, it contains at most $b(d, p) := (g(d, p) + 1)^p$ vertices. In particular, if the root of T is processed, T contains at most $(g(1, p) + 1)^p = 2 \uparrow \uparrow \mathcal{O}(p)$ vertices.

► **Definition 12.** Let G be a graph, T a treedepth decomposition of G , $v \in V(G)$ and \mathcal{C} a contraction sequence for G . We say that a segment Y of \mathcal{C} is an X_v -merge if v has a sibling w in T with the following properties:

- $X_v \sim X_w$, i.e., they are twin-blocks in G .
- In the first trigraph G_i of Y :
 - the subgraph induced by $V(X_w)$ contains no black edges and all red arcs in it are symmetric, and
 - vertices of X_v are in bags only with other vertices of X_w , and
 - for each pair of canonically isomorphic vertices $z \in V(X_v)$ and $z' \in V(X_w)$, let $B(z)$ and $B(z')$ be the bags containing z and z' in G_i , respectively. Then $\alpha(\beta(B(z))) = \beta(B(z')) \cap V(X_w)$ – in particular, the bag contents match when restricted to the subtrees rooted at v and w .
- Each contraction in Y is a contraction between corresponding vertices of the subtrigraphs induced by $V(X_v)$ and $V(X_w)$ that merges pairwise-isomorphic vertices w.r.t. α .
- In the last trigraph of Y , each vertex of X_v has been contracted with a vertex from X_w .

Intuitively, an X_v -merge Y is a subsequence of \mathcal{C} such that at the beginning of the subsequence, X_v and X_w are in the “same state” if we ignore external vertices in the latter, and Y merely contracts the two subtrees into each other. The conditions on X_w moreover guarantee that Y acts as a deletion operation on X_v which will not result in new red edges in the resulting trigraph (even though new red edges may be created in the intermediate steps). In particular:

► **Lemma 13.** Let Y^{start} and Y^{end} be the first and last trigraphs in an X_v -merge Y , respectively. Then $Y^{\text{end}} = Y^{\text{start}} - \{B(x) \mid x \in V(X_v)\}$ and, for each trigraph Y^{mid} between Y^{start} and Y^{end} (included) in \mathcal{C} , $\text{otww}(Y^{\text{mid}}) \leq 2 \cdot \text{otww}(Y^{\text{start}})$.

For each X_v -merge Y , we define its *internal part* Y^{int} as the subsegment obtained by removing the first and last trigraph of Y . We now show that the above notion of X_v -merges has two useful properties: they are unique (in the sense of Lemma 14) and they do not interfere with each other (in the sense of Lemma 15).

► **Lemma 14.** Let G be a graph, T a treedepth decomposition of G and \mathcal{C} a contraction sequence for G . Each vertex $v \in V(G)$ admits at most a single X_v -merge in \mathcal{C} .

► **Lemma 15.** Let G be a graph, T a treedepth decomposition of G , \mathcal{C} a contraction sequence for G and $v, v' \in V(G)$. If v and v' admit an X_v -merge Y and an $X_{v'}$ -merge Y' , respectively, then Y_{int} and Y'_{int} do not intersect in \mathcal{C} , unless the following holds: $X_v \sim X_{v'}$ and $Y = Y'$.

42:10 Approximating Twin-Width Is FPT Parameterized by Treedepth

In particular, we note that the only distinction between the trigraph at the end of Y and the trigraph at the beginning of Y is that all vertices of $T(v)$ have been removed. This will allow us to use X_v merges as a viable reduction rule when parameterized by treedepth – however, care must be given to the fact that during the merge, the oriented twin-width may increase by a factor of at most 2. Towards formally treating this, we define the following refinement of oriented twin-width that will serve as an invariant during the recursive application of our reduction rule.

► **Definition 16.** *Let G be a graph, T a treedepth decomposition of G , and t a positive integer. We say that a (potentially partial) contraction sequence \mathcal{C} for (G, T) has $(t, 2t)$ -oriented twin-width if:*

- *the width of \mathcal{C} is at most $2t$, and*
- *each trigraph in \mathcal{C} of width larger than t lies in the internal part of an X_v -merge for some $v \in V(G)$.*

► **Reduction Rule 1.** *Let (G, T) be a graph with its nice treedepth decomposition of depth p . Let $u \in T$ be not yet processed such that $\forall v \in V(T_u) \setminus \{u\}$, v is processed. Let d be the depth of u . For a connected component H_0 in Comp_u such that the equivalence class of H_0 for \sim_u is strictly larger than $h(d, p) = 2^{2^{4p} \cdot b(d+1, p)}$, delete H from both the graph and the treedepth decomposition to obtain a pair (G', T') .*

Observe that by the choice of H , if G was connected then the graph G' obtained by Reduction Rule 1 remains connected. The cornerstone of our proof will be the following lemma, which states that Reduction Rule 1 is a safe reduction rule to use and that we prove in Section 4.2.

► **Lemma 17.** *Let $(G, T), (G', T')$ be the tuples before and after applying Reduction Rule 1, respectively. If (G', T') admits a $(t, 2t)$ -oriented twin-width contraction sequence \mathcal{C}' , then (G, T) does as well and we can construct the corresponding sequence \mathcal{C} from \mathcal{C}' in time $2 \uparrow \uparrow \mathcal{O}(p) \cdot |V(G)|^{\mathcal{O}(1)}$.*

For now, we proceed towards establishing our results conditioned on the correctness of Lemma 17. The following statement summarizes the outcome after the exhaustive application of Reduction Rule 1.

► **Lemma 18.** *Let (G, T) be a graph and its nice treedepth decomposition with depth p . Applying Reduction Rule 1 exhaustively can be done in polynomial time, and in the resulting pair (G^*, T^*) the graph G^* has size upper bounded by $(g(1, p) + 1)^p = 2 \uparrow \uparrow \mathcal{O}(p)$. Moreover, if (G^*, T^*) has a $(t, 2t)$ -oriented twin-width contraction sequence \mathcal{C}^* , (G, T) does as well and we can construct the corresponding sequence \mathcal{C} from \mathcal{C}^* in time at most $2 \uparrow \uparrow \mathcal{O}(p) \cdot |V(G)|^{\mathcal{O}(1)}$.*

Using Lemma 18 to guarantee the desired properties during the exhaustive application of Reduction Rule 1, we obtain the claimed tractability result for oriented twin-width:

► **Theorem 19.** *2-approximating oriented twin-width is FPT parameterized by treedepth. Specifically, given a graph G on n vertices and treedepth p , there exist a computable function λ and an algorithm running in time $\lambda(p) \cdot n^{\mathcal{O}(1)}$ which outputs a contraction sequence for G of oriented width at most $2\text{otww}(G)$.*

By combining Theorem 19 and Lemma 8, we obtain:

► **Theorem 20.** *There is an algorithm which takes as input an n -vertex graph G of twin-width t and treedepth p , runs in time at most $f(p) \cdot n^{\mathcal{O}(1)}$ and outputs a contraction sequence of width at most $q(t)$, for some computable functions q, f .*

4.2 Proof of Lemma 17

The main idea to prove Lemma 17 is that we will integrate the deleted subgraph H into \mathcal{C}' in a very local manner. The high-level ideas underlying this integration are inspired by those previously used for approximating twin-width via vertex integrity [2], but with significant distinctions caused by the difference in both the employed parameter (treedepth) and the computed measure (oriented twin-width).

Intuitively, we will first follow the sequence of contractions \mathcal{C}' without interacting with H , until at some point we put \mathcal{C}' on hold, identify a special twin-block $H' \sim H$, and *insert* H using H' . Namely, we will do contractions within H to fit the current state of H' , and then use a H -merge to identify vertices of H with vertices of H' . After this insertion, we can continue with the contractions of \mathcal{C}' for the rest of the contraction. The main difficulties of this subsection are to identify the point where we stop following \mathcal{C}' to insert H , and prove the existence of a twin-block H' such that the obtained sequence satisfies our requirements.

Let $(G, T), (G', T')$ be the graph-decomposition pairs that occur, respectively, before and after applying Reduction Rule 1, and H the subgraph of G appearing in T which was deleted to obtain (G', T') . Let $\mathcal{C}' = (G'_0, G'_1, \dots)$ be a contraction sequence for (G', T') .

► **Definition 21.** For a given trigraph G'_i in \mathcal{C}' , the extension $\text{Ext}(G'_i, H)$ is the trigraph obtained by applying the partitioning of vertices of $V(G') \cup V(H)$ following on $V(G')$ the partitioning corresponding to G'_i , and leaving all vertices in $V(H)$ in bags as singletons. This intuitively corresponds to the trigraph obtained from G following the same contractions leading from G' to G'_i . We extend this notion to partial contraction sequences: $\text{Ext}(\mathcal{C}', H) = (\text{Ext}(G'_0, H), \text{Ext}(G'_1, H), \dots)$.

Observe that $\text{Ext}(\mathcal{C}', H)$ and \mathcal{C}' have the same length, and at the end of $\text{Ext}(\mathcal{C}', H)$, all vertices of G' are merged into one bag, but the vertices of H are still each in a separate bag.

► **Definition 22.** Let the pair (G, T) (resp. (G', T')) contain the graph and the associated decomposition before (resp. after) applying Reduction Rule 1, H be the subgraph of G appearing in T which was deleted to obtain (G', T') , and $\text{Ext}(\mathcal{C}', H) = (G = G_0, G_1, G_2, \dots)$ the extension of \mathcal{C}' to G .

- We say that a trigraph G_i in $\text{Ext}(\mathcal{C}', H)$ is H -indifferent if there is no red arc from the bags containing vertices of G' to the vertices of H .
- We say that a trigraph G_i in $\text{Ext}(\mathcal{C}', H)$ is H -safe if: there exist H', H'' twin-blocks to H which are merged by \mathcal{C}' in G_i – in particular, for each $u \in V(H')$, there is a vertex $v \in V(G_i)$ such that $u, \alpha(u) \in \beta(v)$.

► **Observation 23.** $\text{Ext}(\mathcal{C}', H)$ contains H -indifferent trigraphs as well as trigraphs which are not H -indifferent. Moreover, the property is monotone: along $\text{Ext}(\mathcal{C}', H)$ if a trigraph is H -indifferent then so is every trigraph preceding it in $\text{Ext}(\mathcal{C}', H)$.

The next lemma guarantees an H -safe trigraph sufficiently early in $\text{Ext}(\mathcal{C}', H)$.

► **Lemma 24.** Let the pair (G, T) (resp. (G', T')) contain the graph and the associated decomposition that occurs before (resp. after) applying Reduction Rule 1, H be the subgraph of G appearing in T which was deleted to obtain (G', T') , and $\text{Ext}(\mathcal{C}', H) = (G = G_0, G_1, G_2, \dots)$ the extension of \mathcal{C}' to G . The first trigraph in $\text{Ext}(\mathcal{C}', H)$ which is not H -indifferent is H -safe.

In fact, we need a slightly stronger result: for our purposes, it will be necessary to have a graph which is both H -indifferent and H -safe.

42:12 Approximating Twin-Width Is FPT Parameterized by Treedepth

► **Lemma 25.** *Let the pair (G, T) (resp. (G', T')) contain the graph and the associated decomposition before (resp. after) applying Reduction Rule 1, H the subgraph of G appearing in T which was deleted to obtain (G', T') , and $\text{Ext}(\mathcal{C}', H) = (G = G_0, G_1, G_2, \dots)$ the extension of \mathcal{C}' to G .*

The last H -indifferent trigraph G^ in $\text{Ext}(\mathcal{C}', H)$ is H -safe.*

The following lemma has a crucial role in controlling the approximation factor of the final algorithm.

► **Lemma 26.** *Let the pair (G, T) (resp. (G', T')) contain the graph and the associated decomposition before (resp. after) applying Reduction Rule 1, H the subgraph of G appearing in T which was deleted to obtain (G', T') , and $\text{Ext}(\mathcal{C}', H) = (G = G_0, G_1, G_2, \dots)$ the extension of \mathcal{C}' to G . The last H -indifferent trigraph G^* in $\text{Ext}(\mathcal{C}', H)$ does correspond in \mathcal{C}' to a trigraph G'_{i-1} which is not in the internal part of any X_u -merge segment.*

With this final intermediate step done, we proceed to the proof of Lemma 17.

Proof Sketch for Lemma 17. We first construct $\text{Ext}(\mathcal{C}', H) = (G = G_0, G_1, \dots)$, i.e., the extension of \mathcal{C}' to G . We moreover identify the index i of the first trigraph in $\text{Ext}(\mathcal{C}', H)$ which is not H -indifferent. Let z be the seed vertex of H , i.e., $H = X_z$. By Definition 22 and Lemma 25, there are $H', H'' \in \mathcal{H}$ that are merged in G_{i-1} . Towards computing these, we first test, for each pair of siblings a, b of z in T , whether X_a and X_b are twin-blocks (i.e., belong to the same equivalence class of \sim). If they are, we have a canonical isomorphism $\alpha : H' \rightarrow H''$ witnessing the equivalence, and can then test whether each vertex v of H' lies in the same bag as $\alpha(v)$ in G_{i-1} . Once again, by Lemma 25 we are guaranteed that this test will eventually succeed.

Let $\mathcal{C}'_{\text{indifferent}} := (\mathcal{C}')_{[0, i-1]}$. Let $\mathcal{C}'_{H'}$ be the restriction of $\mathcal{C}'_{\text{indifferent}}$ to H' , i.e., $\mathcal{C}'_{H'} = \mathcal{C}'_{\text{indifferent}}[H']$. Using the canonical isomorphism α from H' to H , we define \mathcal{C}_H as the result of applying α on each vertex (or bag) of each trigraph in $\mathcal{C}'_{H'}$. It is easy to observe that \mathcal{C}_H is a valid partial contraction of H . Let us now define 4 partial contraction sequences:

- $\mathcal{C}_A := \text{Ext}(\mathcal{C}'_{\text{indifferent}}, H)$,
- $\mathcal{C}_B := \text{Ext}(\mathcal{C}_H, G_{i-1})$,
- \mathcal{C}_C is an arbitrary H -merge of H into H' starting from the last trigraph of \mathcal{C}_B ,
- $\mathcal{C}_D := (\mathcal{C}')_{\geq i-1}$ the end of the contraction sequence \mathcal{C}' .

We now combine these by appending them in order. To complete the proof, it remains to show that after removing duplicates, the resulting sequence of trigraphs \mathcal{C} is a $(t, 2t)$ -oriented twin-width contraction sequence of G . ◀

5 An Exact Algorithm Based on Vertex Integrity

Let S be a vertex set satisfying that for each connected component H of $G - S$, $|V(H) \cup S| \leq \text{vi}(G)$, and recall that we may compute such a set S using known results (see Section 2). Let \mathcal{D}_S denote the set of connected components of $G - S$; we drop the S when the set is clear from context. For the rest of the section, we assume to have computed and fixed a specific choice of S and that $p = \text{vi}(G)$. Throughout the section, we assume that $p \geq 2$ since instances where $p \leq 1$ are trivial.

5.1 Setting Up the Framework

In this subsection, we adapt the preparatory steps for using the framework [19] to our setting. Our aim here is to define and compute a “reduced graph” whose size is bounded by a function of $\text{vi}(G)$ (Lemma 31). The main novel contributions then lie in Subsection 5.2, which shows that a contraction sequence of the reduced graph can be lifted to a contraction sequence of the original graph.

We first define a basic notion of equivalence between components and restate a known result about computing these.

► **Definition 27.** We say two graphs $H_0, H_1 \in \mathcal{D}$ are twins, denoted $H_0 \sim H_1$, if there exists a canonical isomorphism α from H_0 to H_1 such that for each vertex $u \in V(H_0)$ and each $v \in S$, $uv \in E(G)$ if and only if $\alpha(u)v \in E(G)$.

► **Fact 28** ([19]). Each graph $H \in \mathcal{D}$ has at most p vertices, \sim is an equivalence relation and the number of equivalence classes in $[\sim]$ is upper-bounded by $p \cdot 2^{2p^2}$. Moreover, a partition of connected components into $[\sim]$ can be computed in time at most $\mathcal{O}(p \cdot 2^{2p^2} \cdot n)$.

A crucial feature of the Ramsey Pruning technique is that it requires every connected component in \mathcal{D}_S to occur sufficiently many times. These components will then later be placed into groups, as we define below.

► **Definition 29.** Let k be a positive integer, an equivalence class $[H]$ of \sim is said to be k -large if $|[H]| \geq k$. Further a vertex set $L \subseteq V(G)$ is called a k -large group if the induced subgraph $G[L]$ is a disjoint union of exactly one graph from each k -large equivalence class of \sim .

With this, we can proceed to a formalization of our reduced graphs. Intuitively, this is the graph obtained by first recursively adding all equivalence classes that are “too small” for our purposes into S , until we form a deletion set $S \cup S'$. Note that adding components to the deletion set in such a way does not impact the remaining components, nor the twin relation between them. After that, we place a parameter-bounded number of representatives of “large” equivalence classes into groups and delete all the remaining connected components in $\mathcal{D}_{S \cup S'}$. Towards formalizing this, we fix two computable functions: $g(p) = 2 \uparrow \uparrow (p \cdot 2^{2p^2} \cdot 6 + 6)^p$ upper-bounds the size of the deletion set after we expand it to contain all small equivalence classes, and $f(p, x) = 2^{2^{2^{2^{2^{x+1}}}}}$ specifies a safe bound for how large the groups need to be.

► **Definition 30.** An induced subgraph G' of G is said to be a reduced graph of G if there exists a positive integer $x \leq g(p)$ and a partition of $V(G') = S \uplus S' \uplus Y$ satisfying:

1. $|S \cup S'| = x$ and S' is the set of all vertices in graphs in \mathcal{D} that do not belong to an $f(p, x)$ -large equivalence class of \sim .
2. If there are no $f(p, x)$ -large equivalence classes of \sim , $Y = \emptyset$ and $V(G') = S \uplus S' = V(G)$. Otherwise, it holds that $Y = L_1 \uplus \dots \uplus L_{f(p, x)}$ where L_i is an $f(p, x)$ -large group for each $i \in \{1, \dots, f(p, x)\}$, and each pair of L_i and L_j , $i \neq j$, is vertex-disjoint.

We now prove that a reduced graph can be computed efficiently by essentially following the intuitive description in the previous paragraph. One technical challenge that is treated in the proof is that when we increase the size of the set $S \cup S'$, we also increase the lower bound for our large equivalence classes.

► **Lemma 31.** There exists a reduced graph G' of G , and given S and \sim such a graph can be computed in polynomial time. Further, the number of vertices in G' is upper-bounded by $2 \uparrow \uparrow ((p \cdot 2^{2p^2} \cdot 6 + 6)^p + 10)$.

42:14 Approximating Twin-Width Is FPT Parameterized by Treedepth

The central lemma we will be proving in the next subsection is that every contraction sequence of a reduced graph G' can be lifted to a contraction sequence of G with the same width. We formalize this statement below.

► **Lemma 32.** *Let G' be a reduced graph of G . Then $\text{tw}(G) = \text{tw}(G')$. Moreover, given a partition of G' into $S \uplus S' \uplus L_1 \uplus \dots \uplus L_{f(p,x)}$ and a contraction sequence \mathcal{C}' of G' with width k , we can compute in time $\mathcal{O}((g(p) + p \cdot p \cdot 2^{2p^2})! \cdot |G|^2)$ a contraction sequence \mathcal{C} of G with width k .*

Before proceeding towards the proof of Lemma 32 (which will be our aim in the next two subsections), we show that establishing the lemma would allow us to prove Main Result 2.

► **Theorem 33.** *An optimal contraction sequence of an input graph G can be computed in time $f(p) \cdot n$, where p is the vertex integrity of G and f is a computable function.*

5.2 Towards Lemma 32: The Ramsey Machinery

Recalling Lemma 32, let us consider a reduced graph G' of the input graph G such that $V(G') \neq V(G)$ (as otherwise the lemma is trivial). Hence, let $V(G') = S \uplus S' \uplus L_1 \uplus \dots \uplus L_{f(p,|S \cup S'|)}$ be a partition witnessing that G' is a reduced graph. Let $\mathcal{L} := \{L_1, \dots, L_{f(p,|S \cup S'|)}\}$. For brevity, we will hereinafter use *large* as shorthand for $f(p,|S \cup S'|)$ -large. Note that each of the vertex sets L_i , $i \in [f(p,|S \cup S'|)]$ forms a large group.

For identification and tie-breaking purposes, it will be useful to fix an arbitrary total order $<^\#$ over $V(G)$. Among others, this immediately yields a total order of the equivalence classes $[\sim]$ of \sim and a total order \prec of the large groups in $L_1 \uplus \dots \uplus L_{f(p,|S \cup S'|)}$ (both can be defined, e.g., by the order in which each vertex set is seen when following $<^\#$, however the specific way we induce these total orders does not matter, only their existence). It will also be useful to introduce a notion of “canonical representative” for each of the large equivalence classes in $[\sim]$; intuitively, one could imagine that these canonical representatives all occur in the same large group, but this is neither important nor necessary.

► **Definition 34.** *Let k be the number of large equivalence classes in \sim and let R_i be an arbitrarily chosen but fixed canonical representative of the i^{th} large equivalence class of \sim . Let $R := V(R_1) \uplus \dots \uplus V(R_k)$.*

The set R above will essentially be used as a sort of blank “canvas” for our future statements. In particular, its sole purpose is to have a natural isomorphism to R that allows us to map consistently between different large groups and identify vertices between groups:

► **Definition 35.** *For a large group X with $G[X] = H_1 \uplus \dots \uplus H_k$, $H_i \in [R_i]$ for each $i \in [k]$, let α_X be an isomorphism from $G[X]$ to $G[R] = R_1 \uplus \dots \uplus R_k$ such that for each $i \in [k]$ and vertex $u \in H_i$, $\alpha_X(u) \in V(R_i)$, and for each $v \in S$, $uv \in E(G)$ if and only if $\alpha_X(u)v \in E(G)$.*

For any large group A and for each $u \in R$ we will sometimes use u_A as shorthand for $\alpha_A^{-1}(u)$. From now, say $L' \prec L^\circ \prec L^*$, in \mathcal{L} . The role of these will be to serve as an extended “canvas” for comparing triples of large groups.

► **Definition 36.** *For $X \prec Y \prec Z \in \mathcal{L} \setminus \{L', L^\circ, L^*\}$, we define $\phi_{X,Y,Z} : S \cup S' \cup X \cup Y \cup Z \rightarrow S \cup S' \cup L' \cup L^\circ \cup L^*$ as follows:*

- $\phi_{X,Y,Z}(s) = s$ for each $s \in S \cup S'$
- $\phi_{X,Y,Z}(x) = \alpha_{L'}^{-1}(\alpha_X(x))$ for each $x \in X$

- $\phi_{X,Y,Z}(y) = \alpha_{L^\circ}^{-1}(\alpha_Y(y))$ for each $y \in Y$
- $\phi_{X,Y,Z}(z) = \alpha_{L^*}^{-1}(\alpha_Z(z))$ for each $z \in Z$

Note that $\phi_{X,Y,Z}$ is an isomorphism from $G[S \cup S' \cup X \cup Y \cup Z]$ to $G[S \cup S' \cup L' \cup L^\circ \cup L'']$.

Now, let us consider an arbitrary hypothetical “solution” to our problem on G – that is, a contraction sequence \mathcal{C}' of G with minimum width. Given such \mathcal{C}' , we can characterize the behavior of a triple of large groups via a signature of size that is bounded solely by our parameter. We formalize this below (for a choice of \mathcal{C}'):

► **Definition 37.** For $X \prec Y \prec Z \in \mathcal{L} \setminus \{L', L^\circ, L^*\}$, $\text{info}_{\mathcal{C}'}(X, Y, Z)$ is the contraction sequence on $G[S \cup S' \cup L', L^\circ, L^*]$ obtained from $\mathcal{C}'[S \cup S' \cup X \cup Y \cup Z]$ by applying $\phi_{X,Y,Z}$ on every vertex in every trigraph in $\mathcal{C}'[S \cup S' \cup X \cup Y \cup Z]$.

Essentially, $\text{info}_{\mathcal{C}'}(X, Y, Z)$ is $\mathcal{C}'[S \cup S' \cup X \cup Y \cup Z]$ but translated into the shared canvas of $G[S \cup S' \cup X \cup Y \cup Z]$ – as a consequence, for two distinct triples X, Y, Z and X', Y', Z' of large groups it may happen that $\text{info}_{\mathcal{C}'}(X, Y, Z) = \text{info}_{\mathcal{C}'}(X', Y', Z')$. The above considerations mean that for any set of three large groups, we can apply \prec to obtain an ordering $X \prec Y \prec Z$ and then view $\text{info}_{\mathcal{C}'}(X, Y, Z)$ as a color from a set of a parameter-bounded number of colors. We can then invoke Fact 1 to show that the reduced graph contains a subset of $3p + 3$ “uniformly behaved” large groups w.r.t. \mathcal{C}' , regardless of what \mathcal{C}' is.

► **Lemma 38.** Let G' be a reduced graph of G and \mathcal{C}' be an arbitrary contraction sequence of G' of width w . There exist distinct large groups $H_1, \dots, H_{3p+3} \in \mathcal{L} \setminus \{L', L^\circ, L^*\}$ such that for each $1 \leq a < b < c \leq 3p + 3$ and each $1 \leq a' < b' < c' \leq 3p + 3$, $\text{info}_{\mathcal{C}'}(H_a, H_b, H_c) = \text{info}_{\mathcal{C}'}(H_{a'}, H_{b'}, H_{c'})$.

We call a set of $3p + 3$ large groups satisfying the conditions of Lemma 38 *uniform* (w.r.t. \mathcal{C}'). Below, we show that while Lemma 38 only stipulates that a uniform set satisfies a ternary form of uniformity, this in fact implies also uniformity of pairs and singletons from the set. Many of our arguments in the next subsection in fact only rely on these simpler forms of uniformity. Towards formalizing these, we define $\text{info}_{\mathcal{C}'}(X, Y)$ and $\text{info}_{\mathcal{C}'}(X)$ analogously as $\text{info}_{\mathcal{C}'}(X, Y, Z)$ but where the vertices are only mapped to L', L° and L^* , respectively.

► **Lemma 39.** Let H_1, \dots, H_{3p+3} be a uniform set of large groups. Then for each $1 \leq i < j \leq 3p + 3$ and each $1 \leq k < \ell \leq 3p + 3$, $\text{info}_{\mathcal{C}'}(H_i, H_j) = \text{info}_{\mathcal{C}'}(H_k, H_\ell)$ and $\text{info}_{\mathcal{C}'}(H_i) = \text{info}_{\mathcal{C}'}(H_\ell)$.

5.3 The Proof of Lemma 32

Let $\mathcal{F} \subseteq \mathcal{L} \setminus \{L', L^\circ, L^*\}$ be the family of distinct large groups whose existence is given by Lemma 38 with $\mathcal{F} := (B_1, \dots, B_\omega)$, where $B_i \prec B_j$ if $i < j$. Let \mathcal{C}^* be the restriction of \mathcal{C}' to $V(\bigcup_i B_i \cup S \cup S')$.

► **Definition 40.** We say that a contraction c in \mathcal{C}^* is:

- unique if it contracts bags intersecting $S \cup S'$ together; otherwise,
- special if it contracts a bag intersecting a given B_i with a bag intersecting $S \cup S'$; otherwise
- vertical if it contracts two bags intersecting a given B_i ; otherwise
- horizontal if it contracts a bag x with a bag y such that $\exists v_0 \in \beta(x)$ and $\exists w_0 \in \beta(y)$ such that $\alpha(v_0) = w_0$; and
- diagonal if none of the above cases apply.

42:16 Approximating Twin-Width Is FPT Parameterized by Treedepth

Intuitively, *unique* contractions will form a small number of “milestones” in our considered contraction sequence, *special* contractions merge vertices from large groups with $S \cup S'$, *vertical* contractions merge vertices from the same large group and *horizontal* contractions merge corresponding vertices across large groups. We now prove several useful properties of such contractions, including the fact that diagonal contractions can be disregarded; however, before that it will be useful to introduce some additional terminology.

► **Definition 41.** We say that a contraction c of the bags b_1 and b_2 contracts a pair (x, y) if $x \in \beta(b_1)$ and $y \in \beta(b_2)$.

For any $u \in V(R)$, $x \in S \cup S'$ and $i, j \in [\omega]$, we say that (x, u_{B_i}) and (x, u_{B_j}) are corresponding pairs. For any (not necessarily distinct) $u, v \in V(R)$ and $i < k, j < \ell \in [\omega]$, we say that (u_{B_i}, v_{B_k}) and (u_{B_j}, v_{B_ℓ}) are corresponding pairs.

The proofs of the next three lemmas rely heavily on the Ramsey machinery of Subsection 5.2. In particular, the reason one needs to consider the Ramsey hypergraph theory is to obtain Lemma 44.

► **Lemma 42.** Let c be any non-unique contraction and (x, y) be a pair contracted by c . A pair (x', y') corresponding to (x, y) and intersecting B_1 or B_ω is either contracted by c or had already been contracted by an earlier contraction in the sequence.

To provide intuition for Lemma 42, we present the corresponding pairs in each case:

- for $(x, y) = (x_{B_i}, s \in S \cup S')$: $(x', y') \in \{(x_{B_1}, s), (x_{B_\omega}, s)\}$,
- for $(x, y) = (x_{B_i}, y_{B_j})$ where $i < j$: $(x', y') \in \{(x_{B_1}, y_{B_\ell})_{1 < \ell \leq \omega}, (x_{B_k}, y_{B_\omega})_{1 \leq k < \omega}\}$, and
- for $(x, y) = (x_{B_i}, y_{B_i})$: $(x', y') \in \{(x_{B_1}, y_{B_1}), (x_{B_\omega}, y_{B_\omega})\}$.

► **Lemma 43.** For $i \neq j$, let c contract the pair (x_{B_i}, y_{B_j}) – possibly $x = y$. One of the pairs (x_1, y_2) and $(x_{\omega-1}, y_\omega)$ is either contracted by c or had already been contracted.

► **Lemma 44.** No diagonal contraction can be present in \mathcal{C}^* .

The next definition allows us to identify certain milestones that in turn leads to the crucial notion of *blocks* of \mathcal{C}^* .

► **Definition 45.** We say that two pairs $(A, B \neq A)$ and $(C, D \neq C)$ of large groups are synchronized at a trigraph G_s in \mathcal{C}^* if the trigraphs induced on $V(S \cup S' \cup A \cup B)$ and $V(S \cup S' \cup C \cup D)$ are identical up to renaming via α .

Let H_1, \dots, H_{3p+3} be a uniform set of large groups w.r.t. the contraction sequence \mathcal{C}^* . We say that a segment of \mathcal{C}^* is a *block* if it is a minimal segment of size at least 2 – i.e., one that contains at least one contraction – such that in its first and last trigraphs, the following holds: Every two pairs $(H_i, H_j), (H_k, H_\ell)$ for $i, j, k, \ell \in [3p+3]$, $i \neq j, k \neq \ell$ are synchronized.

For brevity, we say that two large groups $A \neq C$ are *synchronized* at a trigraph G_s in \mathcal{C}^* if the trigraphs induced on $V(S \cup S' \cup A)$ and $V(S \cup S' \cup C)$ are identical up to renaming via α . We note that this notion of synchronization also occurs at the beginning and end of blocks as it is directly implied by the synchronization of pairs.

It is easy to see that, by definition, all pairs are synchronized on G^* and K_1 , thus \mathcal{C}^* contains at least one block. Moreover, blocks form a “near-partitioning” of \mathcal{C}^* where each pair of blocks can only intersect in its first or last trigraph.

We then proceed to prove increasingly restrictive structural properties on the blocks of \mathcal{C}^* , until we reach the following key lemma:

► **Lemma 46.** *A block containing no unique contraction is of the form:*

$C_{\sigma(1)}, C_{\sigma(1),\sigma(2)}, C_{\sigma(2)}, C_{\sigma(2),\sigma(3)}, \dots, \dots, C_{\sigma(\omega-1)}, C_{\sigma(\omega-1),\sigma(\omega)}, C_{\sigma(\omega)}$ where each $C_{\sigma(i)}$ contains only vertical and special contractions of $B_{\sigma(i)}$, C_i and C_j are equivalent w.r.t. α , $C_{\sigma(i),\sigma(i+1)}$ contains exclusively horizontal contractions between $B_{\sigma(i)}$ and $B_{\sigma(i+1)}$, $C_{i,i+1}$ and $C_{j,j+1}$ are equivalent w.r.t. α , and σ is either the identity or $\omega + 1$ minus the identity (reversing the order defined by \prec).

The next lemma is crucial: it allows us to lift “well-behaved” contraction sequences towards G (or a supergraph thereof).

► **Lemma 47.** *Given $k \in \mathbb{N}$, \mathcal{C}^* a contraction sequence of width t for G^* with uniformity, and G_{super} the graph obtained from G^* by adding k large groups. We can construct in linear time a contraction sequence $\mathcal{C}_{\text{super}}$ of width t for G_{super} .*

We can now prove Lemma 32, which was the last missing piece required for Theorem 33.

Proof of Lemma 32. Let G' be a reduced graph of G , a partition $S \uplus S' \uplus L_1 \uplus \dots \uplus L_{f(p,x)}$ of its vertices. Using Lemma 38, we know that for any hypothetical solution of width k for G' , there exists a uniform set of large groups of size $3p + 3$. Thus, we can define G_{core} as the subgraph of G' (and of G) containing only S , S' and an arbitrary set of $3p + 3$ large groups, and brute-force a contraction sequence $\mathcal{C}_{\text{core}}$ of width k for G_{core} , such that the group of large groups in G_{core} is uniform for $\mathcal{C}_{\text{core}}$. This brute-force computation takes time at most $\mathcal{O}((g(p) + p \cdot p \cdot 2^{2p^2})!)$. Let us set m the maximum size of a \sim equivalence class in G . We define G_{super} the supergraph of G such that all equivalence classes of \sim which are not in S' are of size m . We use Lemma 47 to create a contraction sequence $\mathcal{C}_{\text{super}}$ for G_{super} , since it can be obtained from G_{core} by adding $m - k^*$ large groups, and $\mathcal{C}_{\text{core}}$ has the structure argued in Lemma 46. This takes at most linear time in $|\mathcal{C}_{\text{super}}|$ which is $\mathcal{O}(|G|^2)$, and we can in the same amount of time take a restriction of $\mathcal{C}_{\text{super}}$ to G , obtaining the sought-after contraction sequence of width k for G . ◀

References

- 1 Jakub Balabán, Robert Ganian, and Mathis Rocton. Computing twin-width parameterized by the feedback edge number. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France*, volume 289 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.STACS.2024.7.
- 2 Jakub Balabán, Robert Ganian, and Mathis Rocton. Twin-width meets feedback edges and vertex integrity. In Édouard Bonnet and Pawel Rzazewski, editors, *19th International Symposium on Parameterized and Exact Computation, IPEC 2024, September 4-6, 2024, Royal Holloway, University of London, Egham, United Kingdom*, volume 321 of *LIPIcs*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.IPEC.2024.3.
- 3 Jakub Balabán and Petr Hlinený. Twin-width is linear in the poset width. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 6:1–6:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.6.
- 4 Jakub Balabán, Petr Hlinený, and Jan Jedelský. Twin-width and transductions of proper k -mixed-thin graphs. In Michael A. Bekos and Michael Kaufmann, editors, *Graph-Theoretic Concepts in Computer Science - 48th International Workshop, WG 2022, Tübingen, Germany, June 22-24, 2022, Revised Selected Papers*, volume 13453 of *Lecture Notes in Computer Science*, pages 43–55. Springer, 2022. doi:10.1007/978-3-031-15914-5_4.

- 5 Michael J. Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *J. Graph Algorithms Appl.*, 22(1):23–49, 2018. doi:10.7155/jgaa.00457.
- 6 Pierre Bergé, Édouard Bonnet, and Hugues Déprés. Deciding twin-width at most 4 is np-complete. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 18:1–18:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICs.ICALP.2022.18.
- 7 Benjamin Bergougnoux, Eduard Eiben, Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Towards a polynomial kernel for directed feedback vertex set. *Algorithmica*, 83(5):1201–1221, 2021. doi:10.1007/s00453-020-00777-5.
- 8 Sujoy Bhore, Robert Ganian, Fabrizio Montecchiani, and Martin Nöllenburg. Parameterized algorithms for queue layouts. *J. Graph Algorithms Appl.*, 26(3):335–352, 2022. doi:10.7155/JGAA.00597.
- 9 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. *SIAM J. Discret. Math.*, 27(4):2108–2142, 2013. doi:10.1137/120903518.
- 10 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1977–1996. SIAM, 2021. doi:10.1137/1.9781611976465.118.
- 11 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width III: max independent set, min dominating set, and coloring. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPICs*, pages 35:1–35:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICs.ICALP.2021.35.
- 12 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 924–937. ACM, 2022. doi:10.1145/3519935.3520037.
- 13 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Torunczyk. Twin-width IV: ordered graphs and matrices. *J. ACM*, 71(3):21, 2024. doi:10.1145/3651151.
- 14 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, and Stéphan Thomassé. Twin-width V: linear minors, modular counting, and matrix multiplication. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPICs*, pages 15:1–15:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.STACS.2023.15.
- 15 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, and Stéphan Thomassé. Twin-width VI: the lens of contraction sequences. 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 1036–1056. SIAM, 2022. doi:10.1137/1.9781611977073.45.
- 16 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. *J. ACM*, 69(1):3:1–3:46, 2022. doi:10.1145/3486655.
- 17 Domagoj Bradač, Jacob Fox, and Benny Sudakov. The growth rate of multicolor ramsey numbers of 3-graphs. *Research in the Mathematical Sciences*, 11(3):52, 2024.
- 18 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.

- 19 Thomas Depian, Simon D. Fink, Robert Ganian, and Vaishali Surianarayanan. Linear layouts revisited: Stacks, queues, and exact algorithms. In *33rd Annual European Symposium on Algorithms (ESA 2025)*, LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.15.
- 20 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- 21 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.
- 22 Pål Grønås Drange, Markus S. Dregi, and Pim van 't Hof. On the computational complexity of vertex integrity and component order connectivity. *Algorithmica*, 76(4):1181–1202, 2016. doi:10.1007/S00453-016-0127-X.
- 23 Pavel Dvorák, Eduard Eiben, Robert Ganian, Dusan Knop, and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP: programs with few global variables and constraints. *Artif. Intell.*, 300:103561, 2021. doi:10.1016/J.ARTINT.2021.103561.
- 24 Eduard Eiben, Robert Ganian, Thekla Hamm, Lars Jaffke, and O-joung Kwon. A unifying framework for characterizing and computing width measures. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 63:1–63:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.63.
- 25 David Eppstein. The widths of strict outerconfluent graphs. *CoRR*, abs/2308.03967, 2023. arXiv:2308.03967.
- 26 Paul Erdos and Richard Rado. Combinatorial theorems on classifications of subsets of a given set. *Proceedings of the London mathematical Society*, 3(1):417–439, 1952.
- 27 Johannes Klaus Fichte, Robert Ganian, Markus Hecher, Friedrich Slivovsky, and Sebastian Ordyniak. Structure-aware lower bounds and broadening the horizon of tractability for QBF. In *LICS*, pages 1–14, 2023. doi:10.1109/LICS56636.2023.10175675.
- 28 Robert Ganian, Fabian Klute, and Sebastian Ordyniak. On structural parameterizations of the bounded-degree vertex deletion problem. *Algorithmica*, 83(1):297–336, 2021. doi:10.1007/S00453-020-00758-8.
- 29 Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.*, 257:61–71, 2018. doi:10.1016/J.ARTINT.2017.12.006.
- 30 Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. *Theor. Comput. Sci.*, 918:60–76, 2022. doi:10.1016/J.TCS.2022.03.021.
- 31 Tatsuya Gima and Yota Otachi. Extended MSO model checking via small vertex integrity. *Algorithmica*, 86(1):147–170, 2024. doi:10.1007/S00453-023-01161-9.
- 32 Yasuaki Kobayashi and Hisao Tamaki. Treedepth parameterized by vertex cover number. 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 18:1–18:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.18.
- 33 Michael Lampis and Valia Mitsou. Fine-grained meta-theorems for vertex integrity. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd International Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021, Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ISAAC.2021.34.
- 34 Wojciech Nadara, Michal Pilipczuk, and Marcin Smulewicz. Computing treedepth in polynomial space and linear FPT time. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 79:1–79:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.79.

- 35 Jaroslav Nešetřil 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.
- 36 Sang-il Oum. Rank-width and vertex-minors. *J. Comb. Theory, Ser. B*, 95(1):79–100, 2005. doi:10.1016/j.jctb.2005.03.003.
- 37 Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory B*, 96(4):514–528, 2006. doi:10.1016/J.JCTB.2005.10.006.
- 38 Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014. doi:10.1007/978-3-662-43948-7_77.
- 39 Neil Robertson and Paul D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory B*, 35(1):39–61, 1983. doi:10.1016/0095-8956(83)90079-5.
- 40 Martijn van Ee. Some notes on bounded starwidth graphs. *Inf. Process. Lett.*, 125:9–14, 2017. doi:10.1016/J.IPL.2017.04.011.