Computing Twin-Width Parameterized by the Feedback Edge Number

Jakub Balabán
Faculty of Informatics, Masaryk University, Brno, Czech Republic

Robert Ganian
Algorithms and Complexity Group, TU Wien, Austria

Mathis Rocton
Algorithms and Complexity Group, TU Wien, Austria

Abstract

The problem of whether and how one can compute the twin-width of a graph – along with an accompanying contraction sequence – lies at the forefront of the area of algorithmic model theory. While significant effort has been aimed at obtaining a fixed-parameter approximation for the problem when parameterized by twin-width, here we approach the question from a different perspective and consider whether one can obtain (near-)optimal contraction sequences under a larger parameterization, notably the feedback edge number k. As our main contributions, under this parameterization we obtain (1) a linear bikernel for the problem of either computing a 2-contraction sequence or determining that none exists and (2) an approximate fixed-parameter algorithm which computes an ℓ-contraction sequence (for an arbitrary specified ℓ) or determines that the twin-width of the input graph is at least ℓ. These algorithmic results rely on newly obtained insights into the structure of optimal contraction sequences, and as a byproduct of these we also slightly tighten the bound on the twin-width of graphs with small feedback edge number.

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

Keywords and phrases twin-width, parameterized complexity, kernelization, feedback edge number

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.7

Related Version *Full Version*: `https://arxiv.org/abs/2310.08243`

Funding *Robert Ganian*: Robert Ganian acknowledges support by the FWF and WWTF Science Funds (FWF project Y1329 and 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

Since its introduction by Bonnet, Kim, Thomassé and Watrigant in 2020 [16], the notion of *twin-width* has made an astounding impact on the field of algorithmic model-checking [10, 11, 12, 13, 14]. Indeed, it promises a unified explanation of why model-checking first order logic is fixed-parameter tractable on a number of graph classes which were, up to then, considered to be separate islands of tractability for the model-checking problem, including proper minor-closed graphs, graphs of bounded rank-width, posets of bounded width and map graphs [16]; see also the recent works on other graph classes of bounded twin-width [1, 2, 21]. Beyond this, twin-width was shown to have fundamental connections to rank-width and path-width [14] as well as to matrix theory [13], and has by now been studied even in areas such as graph drawing [21] and SAT Solving [27, 34].

And yet, essentially all twin-width based algorithmic results known to date require a corresponding decomposition – a so-called *contraction sequence* – to be provided as part of the input. The fact that the inner workings of these algorithms rely on a contraction sequence is not surprising; after all, the same reliance on a suitable decomposition is present in essentially all graph algorithms parameterized by classical width measures such as treewidth [33] or rank-width [32, 24]. But while optimal decompositions for treewidth and rank-width can be computed in fixed-parameter time when parameterized by the respective width measure [6, 28] and even more efficient algorithms are known when aiming for decompositions that are only a constant-factor worse than optimal [30, 23], the situation is entirely different in the case of twin-width. In particular, it is known that already deciding whether a graph has twin-width at most 4, i.e., admits a 4-contraction sequence, is NP-hard [4] (ruling out fixed-parameter as well as XP algorithms for computing optimal contraction sequences). Moreover, whether one can at least compute approximately-optimal contraction sequences in fixed-parameter time is arguably the most prominent open question in contemporary research of twin-width.

Contribution. Given the difficulty of computing (near-)optimal contraction sequences when parameterized by twin-width itself, in this article we ask whether one can at least compute such contraction sequences under a larger parameterization, i.e., when using an auxiliary parameter which yields stronger restrictions on the input graph[1]. Algorithms obtained under such stronger restrictions are not intended to be used as a pre-computation step prior to using twin-width for model checking, but rather aim to further our understanding of the fundamental problem of computing (near-)optimal contraction sequences. In this sense, our work follows in the footsteps of previous work on, e.g., treedepth parameterized by the vertex cover number [29], MIM-width parameterized by the feedback edge number and other parameters [20], treewidth parameterized by the feedback vertex number [9] and the directed feedback vertex number parameterized by the (undirected) feedback vertex number [5].

As our two main contributions, we obtain the first non-trivial fixed-parameter algorithms for computing (near-)optimal contraction sequences.

▶ **Theorem 1.** *The problem of deciding whether the twin-width of an input graph is at most* 2 *admits a linear bikernel when parameterized by the feedback edge number k. Moreover, a* 2*-contraction sequence for G (if one exists) can be computed in time $2^{\mathcal{O}(k \cdot \log k)} + n^{\mathcal{O}(1)}$.*

We remark that Theorem 1 providing a *bikernel* [19] (instead of a kernel) is merely due to the output being a *trigraph* [16]. Our second result targets graphs of higher twin-width:

▶ **Theorem 2.** *There is an algorithm which takes as input an n-vertex graph G with feedback edge number k, runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ for a computable function f, and outputs a contraction sequence for G of width at most* $\text{tww}(G) + 1$.

We note that the graph parameter used in our results – the feedback edge number or equivalently the edge deletion distance to acyclicity – is highly restrictive and provides stronger structural guarantees on the input graph than not only twin-width itself, but also rank-width and treewidth. In a sense, it is one of the two most "restrictive" structural parameters used in the design of fixed-parameter algorithms [35, 3, 26, 25, 22], with the other being the vertex cover number, i.e., the minimum size of a vertex cover (see also Figure 1). But while there is a trivial fixed-parameter algorithm for computing optimal contraction

[1] When approximating width parameters, it is desirable to aim for approximation errors which depend only on the targeted width parameter.

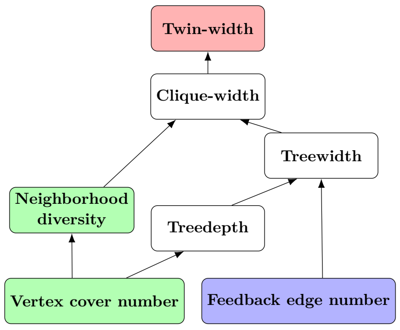

Figure 1 Complexity of computing twin-width with respect to notable structural parameters. A directed path from parameter x to parameter y indicates that y is upper-bounded by a function of x, i.e., that x is more restrictive than y. Green marks parameters where the problem is trivially fixed-parameter tractable; red and white signify para-NP-hardness and cases where the complexity is unknown, respectively; the parameter considered in this paper is highlighted in blue.

sequences w.r.t. the vertex cover number[2] and also a polynomial-time algorithm for solving the same problem on trees [16], lifting the latter to our more general setting parameterized by the feedback edge number is far from a trivial undertaking and goes hand in hand with the development of new insights into optimal contraction sequences of highly structured graphs.

Proof Overview and Techniques. At its core, both of our results are kernelization routines in the sense that they apply polynomial-time reduction rules in order to transform the input instance into an "equivalent" instance whose size is upper-bounded by a function of the parameter alone. The required reduction rules are in fact very simple – the difficulty lies in proving that they are safe. The used parameterization then allows us to argue that after exhaustive application of these results, we are guaranteed to obtain an instance whose size is upper-bounded by a function of the feedback edge number. Below, we provide a high-level overview of the proof; for brevity, we assume here that readers are familiar with the basic terminology associated with twin-width (as also introduced in Section 2).

The first step towards both desired kernelization algorithms consists of a set of tree-pruning rules which allow us to "cut off" subtrees in the graph that are connected to the rest of the graph via a bridge, i.e., a single edge. Already this step (detailed in Section 3) requires some effort in the context of twin-width, and replaces the cut-off subtrees by one of two kinds of *stumps* which depend on the properties of the replaced subtree. After the exhaustive application of these general rules, in Section 4 we apply a further cleanup step which uses the structural properties guaranteed by our parameter to deal with the resulting stumps. This reduces the instance to an equivalent trigraph consisting of $\mathcal{O}(k)$-many vertices plus a set of "*dangling*" paths connecting these vertices – paths whose internal vertices have degree 2. We note that some of the reduction rules developed here, and in particular those which allow us to safely remove trees connected via a bridge, provide techniques that could be lifted to remove other kinds of dangling subgraphs and hence may be of general interest. In fact, we

[2] In particular, this follows by repeatedly deleting vertices which are twins (an operation which is known to preserve twin-width [16]) until one obtains a problem kernel.

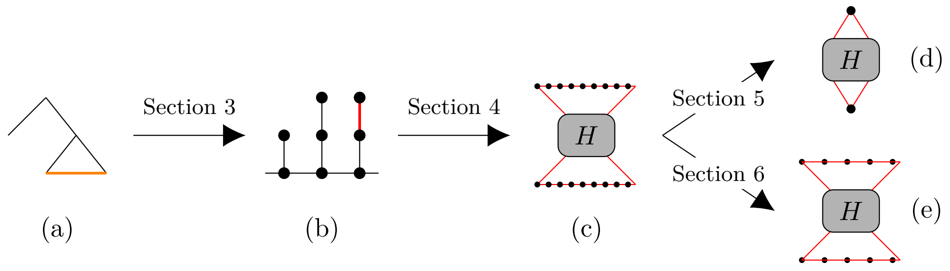

Figure 2 **(a)** The input graph – a tree with k extra edges. **(b)** Three kinds of stumps, which are obtained by cutting down dangling trees. **(c)** A small subtrigraph H and long dangling red paths. **(d)** If the target twin-width is 2, the paths can be shortened to single vertices. **(e)** Paths can be shortened to bounded length while retaining a guarantee on the twin-width. A detailed example depicting the first two steps is provided in Figure 4.

use our reduction rules to improve the previously known twin-width 2 upper bound from trees to graphs of feedback edge number 1 (Theorem 21), which additionally yields a slightly tighter relationship between twin-width and the feedback edge number (Corollary 22).

The structure of the trigraph at this point seems rather simple: it consists of a bounded-size part plus a small set of arbitrarily long dangling paths. Intuitively, one would like to obtain a bikernel by showing that each sufficiently long dangling path can be replaced by a path of length bounded by some constant without altering the twin-width. In Section 5, we implement this approach by guaranteeing the existence of "well-structured" 2-contraction sequences for graphs of twin-width 2, in turn allowing us to complete the proof of Theorem 1.

The situation becomes significantly more complicated when aiming for contraction sequences for graphs of higher twin-width. In particular, not only does the approach used in Section 5 not generalize, but we prove that there can exist no safe twin-width preserving rule to shorten dangling paths to a constant length, for any constant (see Proposition 9). Circumventing this issue – even when allowing for an additive error of one – in Section 6 forms the most challenging part of our results. The core idea used in the proof here is to partition a hypothetical optimal contraction sequence into a bounded number of stages (defined via so-called *blueprints*). Crucially, we show that the original contraction sequence can be transformed into a "nice" sequence where we retain control over the operations carried out in each stage, at the cost of allowing for a slightly higher width of the sequence. The structure in these nice sequences is defined by aggregating all the descendants of the dangling paths into so-called *centipedes*. Afterward, we use an iterative argument to show that such a well-behaved sequence can also be used to deal with a kernelized trigraph where all the long dangling paths are replaced by paths whose length is not constant, but depends on a function of k.

A mind map of our techniques and algorithmic results is provided in Figure 2.

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 [18] and parameterized algorithmics [19, 17].

The *length* of a path is the number of edges it contains. An edge set F in an n-vertex graph G is called a *feedback edge set* if $G - F$ is acyclic, and the *feedback edge number* of G is the size of a minimum feedback edge set in G. A *dangling path* in G is a path of vertices which all have degree 2 in G, and a *dangling tree* in G is an induced subtree in G which can be separated from the rest of G by a bridge (see, e.g., Figure 4 later).

Twin-Width. 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. We say a (sub)graph is *black (*resp. *red)* if all of its edges are black (resp. red); for example, P is a red path in G if it is a path containing only red edges. Without a color adjective, the path (or a different kind of subgraph) may contain edges of both colors. We use $G[Q]$ to denote the subtrigraph of G induced on $Q \subseteq V(G)$.

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 contains only a single black edge to either v or u. A sequence $C = (G = G_1, \ldots, G_n)$ is a *partial contraction sequence of* G if it is a sequence of trigraphs such that for all $i \in [n-1]$, G_{i+1} is obtained from G_i by contracting two vertices. A *contraction sequence* is a partial contraction sequence which ends with a single-vertex graph. The *width* of a (partial) contraction sequence C, denoted $w(C)$, is the maximum red degree over all vertices in all trigraphs in C; we also use α*-contraction sequence* as a shorthand for a contraction sequence of width at most α. The *twin-width* of G, denoted $\mathrm{tww}(G)$, is the minimum width of any contraction sequence of G, and a contraction sequence of width $\mathrm{tww}(G)$ is called *optimal*. An example of a contraction sequence is provided in Figure 3.

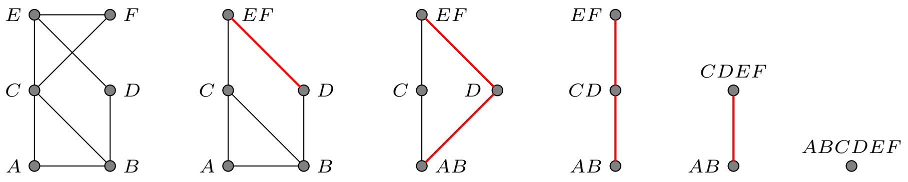

Figure 3 A 2-contraction sequence of the leftmost graph, consisting of 6 trigraphs.

Let us now fix a contraction sequence $C = (G = G_1, \ldots, G_n)$. 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.

Note that if a vertex u appears in multiple trigraphs in 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 *an ancestor* of v in G_i and v is *the descendant* of u in G_j (clearly, this descendant is unique). If H is an induced subtrigraph of G_i, then $u \in V(G_j)$ is a *descendant* of H if it is a descendant of at least one vertex of H, and we say that $u \in V(G_i)$ is *contracted to* H *in* G_j if u is an ancestor of a descendent of H in G_j. A contraction of $u, v \in V(G_j)$ into $uv \in V(G_{j+1})$ *involves* $w \in V(G_i)$ if w is an ancestor of uv.

The following definition provides terminology that allows us to partition a contraction sequence into "steps" based on contractions between certain vertices in the original graph.

▶ **Definition 3.** *Let C be a contraction sequence of a trigraph G, and let H be an induced subgraph of G with $|V(H)| = m$. For $i \in [m-1]$, let $C\langle i\rangle_H$ be the trigraph in C obtained by the i-th contraction between two descendants of H, and let $C\langle 0\rangle_H = G$. For $i \in [m-1]$, let U_i and W_i be the bags of the vertices which are contracted into the new vertex of $C\langle i\rangle_H$.*

A contraction sequence $C[H] = (H = H_1, \dots, H_m)$ *is the* restriction of C to H *if for each* $i \in [m-1]$, H_{i+1} *is obtained from* H_i *by contracting the two vertices* $u, w \in V(H_i)$ *such that* $\beta(u) = U_i \cap V(H)$ *and* $\beta(w) = W_i \cap V(H)$.

Next, we introduce a notion that will be useful when dealing with reduction rules in the context of computing contraction sequences.

▶ **Definition 4.** *Let* G, G' *be trigraphs. We say that the twin-width of* G' *is* effectively *at most the twin-width of* G*, denoted* $\operatorname{tww}(G') \leq_e \operatorname{tww}(G)$*, if (1)* $\operatorname{tww}(G') \leq \operatorname{tww}(G)$ *and (2) given a contraction sequence* C *of* G*, a contraction sequence* C' *of* G' *of width at most* $w(C)$ *can be constructed in polynomial time. If* $\operatorname{tww}(G') \leq_e \operatorname{tww}(G)$ *and* $\operatorname{tww}(G) \leq_e \operatorname{tww}(G')$*, then we say that the two graphs have* effectively *the same twin-width,* $\operatorname{tww}(G') =_e \operatorname{tww}(G)$.

We say that G' is a *pseudoinduced* subtrigraph of G if G' is obtained from an induced subtrigraph of G by the removal of red edges or their replacement with black edges.

▶ **Observation 5.** *If* G' *is a pseudoinduced subtrigraph of* G*, then* $\operatorname{tww}(G') \leq_e \operatorname{tww}(G)$.

Preliminary Observations and Remarks. We begin by stating a simple brute-force algorithm for computing twin-width.

▶ **Observation 6.** *An optimal contraction sequence of an* n*-vertex graph can be computed in time* $2^{\mathcal{O}(n \cdot \log n)}$.

The following observation not only establishes the twin-width of trees – which form a baseline case for our algorithms – but also notes that the necessary contractions are almost entirely independent of the choice of the root.

▶ **Observation 7** ([16, Section 3]). *For any rooted tree* T *with root* r*, there is a contraction sequence* C *of* T *of width at most* 2 *such that the only contraction involving* r *is the very last contraction in* C.

Next, we recall that a 1-contraction sequence can be computed in polynomial time not only on graphs, but also on trigraphs with at most one red edge.

▶ **Theorem 8** ([15, Section 7]). *If* G *is a trigraph with at most one red edge, then it can be decided in polynomial time whether the twin-width of* G *is at most 1. In the positive case, the algorithm also returns an optimal contraction sequence of* G.

Finally, we formalize the claim made in Section 1 that there can be no "simple" twin-width preserving reduction rule for handling long dangling paths; in particular, any such rule for simplifying dangling paths cannot depend purely on the length of the path itself.

▶ **Proposition 9.** *For every integer* $c \geq 1$*, there exists a graph* G^c *with the following properties:* (1) G^c *contains a dangling path* P *of length* c, (2) $\operatorname{tww}(G^c) \geq 5$*, and* (3) *the graph obtained by subdividing one edge in* P *(i.e., replacing* P *with a path* P' *whose length is* $c+1$*) has twin-width at most* 4.

Finally, we remark that throughout the paper, we assume the input graph G to be connected; this is without loss of generality, since otherwise one can handle each of the graph's connected components separately.

3 Cutting Down a Forest

After computing a minimum feedback edge set of an input graph G, the first task on our route towards Theorems 1 and 2 is to devise reduction rules which can safely deal with dangling trees. While it is not possible to delete such trees entirely while preserving twin-width, we show that they can be safely "cut down"; in particular, depending on the structure of the tree it can be replaced by one of the two kinds of stumps defined below.

▶ **Definition 10.** *Let G be a trigraph, and let $u, v, w \in V(G)$. We say that u has a* half stump *v if $uv \in E(G)$, and the degree of v in G is 1. We say that u has a* red *(resp.* black*)* stump *vw if $uv \in E(G)$ and $vw \in R(G)$ (resp. $vw \in E(G)$), and the degrees of v and w are 2 and 1, respectively. Half, red, and black stumps are collectively called* stumps*. The stump vw (or v) then* belongs *to u.*

We begin by observing that special kinds of subtrees – specifically, stars – can be safely replaced with just a black stump.

▶ **Observation 11.** *Let G be a trigraph with a dangling tree T connected to the rest of the graph via the bridge $e = uv$ where $v \in V(T)$. Assume that T is a black star consisting of at least one vertex other than v. Then G has effectively the same twin-width as the trigraph G' obtained from $G - T$ by adding a black stump to u.*

Next, we show that all dangling trees not covered by Observation 11 can be safely cut down to a red stump, as long as the trigraph obtained by the cutting has twin-width at least 2 (a condition that will be handled later on). The proof of this case is significantly more difficult than the previous one.

▶ **Lemma 12.** *Let G be a trigraph with a bridge $e = uv$ such that the connected component T of $G - \{e\}$ containing v is a black dangling tree which contains a vertex at distance 2 from v and let G' be the trigraph obtained from $G - T$ by adding a red stump to u. If $\mathrm{tww}(G) \geq 2$ and $\mathrm{tww}(G') \geq 2$, then G and G' have effectively the same twin-width.*

Proof Sketch. We begin by establishing $\mathrm{tww}(G) \leq_e \mathrm{tww}(G')$. Let C' be a contraction sequence of G', and we will show how to construct a contraction sequence $C = (G = G_0, G_1, \dots)$ of G of width at most $w(C')$. C starts with contracting T, following the contraction sequence C_T given by Observation 7 (with v as the root), but stops before the first contraction involving v (which is the very last contraction in C_T). Let G_i be the obtained trigraph. By definition of C_T, no vertex of G_j, $j \in [i]$, exceeds the bound on its red degree since $w(C_T) \leq 2 \leq \mathrm{tww}(G')$. G_i is isomorphic to G' and so from here on, C follows C'.

Our task in the remainder of the proof will be to establish $\mathrm{tww}(G') \leq_e \mathrm{tww}(G)$. Let $C = (G, G_1, \dots)$ be a contraction sequence of G, and let $w, x \in V(T) \setminus \{v\}$ be such that $vw, wx \in E(T)$. We need to construct a contraction sequence of G' of width at most $w(C)$; note that there can be vertices with red degree up to 2 in this desired sequence since $w(C) \geq \mathrm{tww}(G) \geq 2$. Let $G^- := G[V(G - T) \cup \{v, w\}]$. Observe that the only difference between G' and G^- is the color of the edge vw (it is red in G', but black in G^-) and let $C^- = (G^- = G^-_0, G^-_1, \dots, G^-_m)$ be the restriction of C to G^-; hence $w(C^-) \leq w(C)$. Let us consider the contraction sequence $C' = (G' = G'_0, G'_1, \dots, G'_m)$ of G' which follows C^- in each step (i.e., $V(G^-_i) = V(G'_i)$ for all $i \in [m]$). To avoid any confusion, we explicitly note that it is not possible to rule out $w(C') > w(C^-)$. We complete the proof by performing a case distinction that will allow us to either guarantee $w(C') \leq w(C^-)$, or – in the most difficult case – construct a new contraction sequence C'' of G' such that $w(C'') \leq w(C)$. ◀

Let us now provide some intuition on how we aim to apply Lemma 12 (a formalization is provided in the proof of Theorem 17 at the end of this section). Assume without loss of generality that the input graph G has twin-width at least 2. The first time we want to apply Lemma 12 to go from G to G', we can verify that $\text{tww}(G') \geq 2$ by Theorem 8; if this check fails then we show how to construct a 2-contraction sequence of G, and otherwise we replace T with a red stump as per the lemma statement. For every subsequent application of Lemma 12, G' will have two red edges and hence we cannot rely on Theorem 8 anymore – but instead, we can guarantee that the condition on G' holds by establishing the following lemma.

▶ **Lemma 13.** *Let G be a connected trigraph with two red stumps. Then* $\text{tww}(G) \geq 2$.

With Observation 11 and Lemmas 12-13, we can effectively preprocess (tri)graphs by "cutting down" all dangling trees (i.e., replacing them with stumps). However, we will also need to deal with the fact that a vertex could now be connected to many distinct stumps. For half stumps this is not an issue, as multiple half stumps can be assumed to be contracted into a single half stump due to the vertices in them being twins. For all pairs of other kinds of stumps (except for a pair consisting of a half and a black stump), we show that it is sufficient to replace these with a single stump instead.

▶ **Observation 14.** *Let G' be a trigraph such that* $\text{tww}(G') \geq 2$*, let $u \in V(G')$ be a vertex with a red stump S in G', and let G be a trigraph obtained from G' by adding an additional stump to u. Then G' and G have effectively the same twin-width.*

▶ **Lemma 15.** *Let G' be a trigraph, let $u \in V(G')$ be a vertex with a red stump S in G', and let G be a trigraph obtained from G' by removing S and adding two black stumps to u. If* $\text{tww}(G) \geq 2$ *and* $\text{tww}(G') \geq 2$*, then G' and G have effectively the same twin-width.*

We conclude this section by formalizing the trigraph that can be obtained through the exhaustive application of the reduction rules arising from Observations 11, 14 and Lemmas 12, 15. We say that an induced subtrigraph in G is a *dangling pseudo-path* if it can be obtained from a dangling path P in G by adding, to each of the vertices in P, either (a) one red stump or (b) at most one black stump and at most one half stump.

▶ **Definition 16.** *A connected trigraph G with* $\text{tww}(G) \geq 2$ *is an $(H, \mathcal{P})$-graph if $\mathcal{P}$ is a set of dangling pseudo-paths in G, and there are two disjoint induced subtrigraphs of G, H and $\sqcup\mathcal{P}$ (the disjoint union of all paths in $\mathcal{P}$), such that each vertex of G belongs to one of them.*

We proceed with some related terminology that will be used extensively in the subsequent sections. A vertex $u \in V(H)$ is a *connector in G* if u is adjacent to a vertex of $\sqcup\mathcal{P}$ in G. We say that $P \in \mathcal{P}$ is *original* if all edges in P that do not belong to a stump are black and the edges connecting the endpoints of P to H are also black. Later, we will also deal with *tidy* paths in $\mathcal{P}$, where $P \in \mathcal{P}$ is tidy if P is a dangling red path (i.e., contains no stumps) which additionally satisfies the following three technical conditions for each connector $u \in V(H)$ adjacent to an endpoint v of P:

1. u has black degree 0;
2. v is the only neighbor of u in $\sqcup\mathcal{P}$; and
3. u has a unique neighbor u' in $V(H)$ and u' has positive black degree.

We say that an $(H, \mathcal{P})$-graph is *original* (*tidy*) if all paths $P \in \mathcal{P}$ are original (tidy, respectively). An illustration of these notions is provided in Figure 4, which also showcases the outcome of applying the culmination of this section – Theorem 17 – on an input graph.

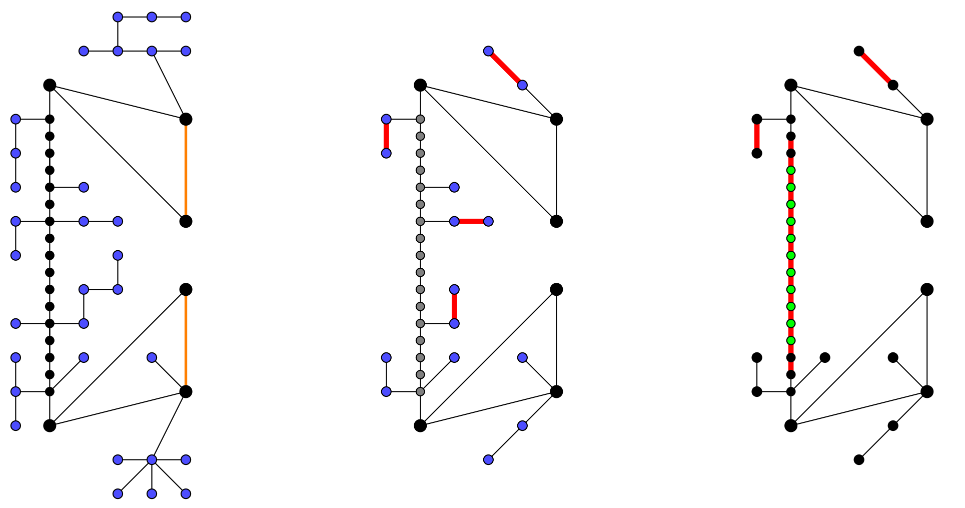

Figure 4 Left: A graph G with feedback edge number two. Feedback edges are orange, vertices in dangling trees are blue. **Middle:** The original $(H, \mathcal{P})$-graph obtained from G after all dangling trees have been cut down (i.e., the outcome of Theorem 17). A dangling pseudo-path is depicted via the grey vertices. **Right:** The tidy $(H, \mathcal{P})$-graph that will later be obtained from the original $(H, \mathcal{P})$-graph by applying Corollary 20 at the end of Section 4. Here, $\mathcal{P}$ contains a single tidy dangling path which is colored green, and all other vertices lie in H.

▶ **Theorem 17.** *There is a polynomial-time procedure which takes as input a graph G with feedback edge number k and either outputs an optimal contraction sequence of G of width at most 2, or an original $(H, \mathcal{P})$-graph G' with effectively the same twin-width as G such that $|V(H)| \leq 16k$ and $|\mathcal{P}| \leq 4k$.*

4 Cleaning the Paths

In the second phase of our proof, our aim is to simplify the instance even further after Theorem 17 in order to reach a trigraph which is "clean" enough to support the path reduction rules developed in the next sections. In particular, we will show that all the dangling pseudo-paths arising from Theorem 17 can be safely transformed into red dangling paths, resulting in an $(H, \mathcal{P})$-graph that is tidy as per the definition in the previous section.

We first show how to deal with stumps belonging to a single vertex.

▶ **Observation 18.** *Let G be a trigraph, let $u \in V(G)$ be a vertex with a single stump or a black and a half stump, and let G' be the trigraph obtained from G by deleting the stumps belonging to u and making all edges incident to u red. There is a partial contraction sequence from G to G' of width $\max\{d_1 + 1, d_2\}$, where d_1 is the red degree of u in G and d_2 is the maximum red degree of any vertex in G'.*

Next, we establish that if a dangling pseudo-path has two consecutive vertices without stumps, the edge between them can be turned red. We remark that this lemma will be applied to subtrigraphs of the considered trigraph, and hence we cannot assume that the trigraph has twin-width at least 2.

▶ **Lemma 19.** *Let G be a trigraph, let (u_1, u_2, u_3, u_4) be an induced path in G in this order, and assume that the degree of u_2 and u_3 is 2 in G and that the degree of $u \in \{u_1, u_4\}$ in G is 3 if u has a single stump, 4 if u has a half stump and a black stump, and 2 otherwise. Let G' be the trigraph obtained from G by changing the color of the edge u_2u_3 to red. Given a contraction sequence C of G, a contraction sequence C' of G' such that $w(C') = \max\{2, w(C)\}$ can be constructed in polynomial time.*

With Lemma 19, we show that a dangling pseudo-path can either be safely transformed into a "real" dangling path, or (if the path is too short) absorbed into H, which in turn allows us to prove:

▶ **Corollary 20.** *There is a polynomial-time algorihm which transforms an original $(H, \mathcal{P})$-graph into a tidy $(H', \mathcal{P}')$-graph with effectively the same twin-width such that $|V(H')| \leq |V(H)| + 24 \cdot |\mathcal{P}|$ and $|\mathcal{P}'| \leq |\mathcal{P}|$.*

Before we proceed towards establishing our main algorithmic theorems, we remark that Theorem 17 and Corollary 20 allow us to bound the twin-width of graphs with feedback edge number 1, generalizing the earlier result of Bonnet et al. [16, Section 3] for trees.

▶ **Theorem 21.** *Every graph with feedback edge number 1 has twin-width at most 2.*

As an immediate corollary, we can obtain an even more general statement:

▶ **Corollary 22.** *Every graph with feedback edge number $\ell \geq 1$ has twin-width at most $1 + \ell$.*

5 Establishing Theorem 1: Recognizing Twin-width 2

Our aim now is to make the step from Corollary 20 towards a proof of Theorem 1. Towards this, let us fix a tidy n-vertex $(H, \mathcal{P})$-graph G. When dealing with a contraction sequence, we will use G_i to denote the i-th trigraph obtained from G, and let H_i be the subtrigraph of G_i induced by the descendants of H. We say that $u \in V(G_i)$ is *an outer vertex in G_i* if $u \notin V(H_i)$, and we lift the previous definition of connectors by saying that u is a *connector in G_i* if $u \in V(H_i)$ and u is adjacent to an outer vertex in G_i.

We begin with a simple observation which will be useful throughout the rest of the section.

▶ **Observation 23.** *If G_i is a trigraph obtained from a tidy $(H, \mathcal{P})$-graph G by a sequence of contractions, then all outer vertices and all connectors in G_i have black degree 0 in G_i.*

Our proof of Theorem 1 relies on establishing that if a tidy $(H, \mathcal{P})$-graph G has twin-width 2, then it also admits a contraction sequence which is, in a sense, "well-behaved". The proof of this fact is based on induction, and hence being "well-behaved" (formalized under the notion of *regularity* below) is defined not only for entire sequences but also for prefixes.

▶ **Definition 24.** *Let $C = (G_1 = G, G_2, \ldots, G_n)$ be a contraction sequence. For $P \in \mathcal{P}$, let us denote by P_i the subtrigraph of G_i induced by the descendants of P which are not in H_i. We say that a prefix $(G_1, \ldots, G_i)$ of C is* regular *if:*

- *for all $j \in [i]$, $\{P_j \mid P \in \mathcal{P}\}$ is a set of disjoint red paths, and the endpoints of these paths are adjacent to connectors; and*
- *for all $j \in [i - 1]$:*
  1. *G_{j+1} is obtained by a contraction inside H_j, or*
  2. *there is $P \in \mathcal{P}$ such that if you shorten P_j by one vertex in G_j, you obtain G_{j+1}, or*
  3. *there is $P \in \mathcal{P}$ such that $|V(P_j)| = 1$ and $|V(P_{j+1})| = 0$.*

Let $reg(C)$ denote the length of the longest regular prefix of C. Below, we show that unless we reach a degenerate trigraph (a case which is handled in the proof of Proposition 26 later), every contraction sequence of width 2 can be made "more regular" until it is entirely regular.

▶ **Lemma 25.** *Let $C = (G_1 = G, G_2, \dots, G_n)$ be an optimal contraction sequence of G of width 2, and let $i := reg(C)$. If $i < n$ and G_i is not a red cycle of length 4, then there is an optimal contraction sequence C' of G such that $reg(C') > i$.*

We can now use Lemma 25 to show that contracting all the tidy dangling paths in G into singletons cannot increase the twin-width of G.

▶ **Proposition 26.** *Let G' be the trigraph obtained from a tidy $(H, \mathcal{P})$-graph G of twin-width 2 by shortening each path in $\mathcal{P}$ to a single vertex. Then* $\operatorname{tww}(G') = 2$.

With Proposition 26 in hand, we can complete the proof of Theorem 1.

▶ **Theorem 1.** *The problem of deciding whether the twin-width of an input graph is at most 2 admits a linear bikernel when parameterized by the feedback edge number k. Moreover, a 2-contraction sequence for G (if one exists) can be computed in time $2^{\mathcal{O}(k \cdot \log k)} + n^{\mathcal{O}(1)}$.*

Proof. First we use Theorem 17: if it returns an optimal contraction sequence of the input graph G_0, we immediately know its twin-width. Otherwise, we obtain an original $(H, \mathcal{P})$-graph G with effectively the same twin-width as G_0 such that $|V(H)| \leq 16k$ and $|\mathcal{P}| \leq 4k$. Now we use Corollary 20 to transform G into a tidy $(H', \mathcal{P}')$-graph G' that has effectively the same twin-width as G and that satisfies $|V(H')| \leq 112k$ and $|\mathcal{P}'| \leq 4k$. By transitivity of $=_e$, we obtain $\operatorname{tww}(G_0) =_e \operatorname{tww}(G')$. Finally, let G'' be the trigraph obtained from G' by shortening each path in $\mathcal{P}'$ to a single vertex.

By Proposition 26, $\operatorname{tww}(G') = 2$ implies $\operatorname{tww}(G'') = 2$. Conversely, given a contraction sequence C'' of G'', we can construct a contraction sequence C' of G' of width at most $w(C'')$ by first shortening each path in $\mathcal{P}'$ to a single vertex via progressive contractions of consecutive vertices, and then following C''; thus, $\operatorname{tww}(G') \leq_e \operatorname{tww}(G'')$. Since $\operatorname{tww}(G') \geq 2$, we obtain that $\operatorname{tww}(G'') = 2$ implies $\operatorname{tww}(G') = 2$. By combining these two implications with $\operatorname{tww}(G_0) = \operatorname{tww}(G')$, we obtain that $\operatorname{tww}(G_0) = 2$ if and only if $\operatorname{tww}(G'') = 2$. Since all operations required to construct G'' from G_0 can be performed in polynomial time and $|V(G'')| \leq 116k$, G'' is indeed a linear bikernel for the considered problem.

Finally, if $\operatorname{tww}(G_0) \leq 2$, an optimal contraction sequence of G_0 can be computed in the desired time: either it is given by Theorem 17, or we construct G'' in polynomial time and compute an optimal contraction sequence of G'' in time $2^{\mathcal{O}(k \cdot \log k)}$ as per Observation 6, and then the result follows by the effectiveness in $\operatorname{tww}(G_0) =_e \operatorname{tww}(G') \leq_e \operatorname{tww}(G'')$. ◀

6 Establishing Theorem 2: Almost-Optimal Contraction Sequence

We now move on to the most involved part of the paper: the final step towards proving Theorem 2, which we will outline in the next few paragraphs. Recall that after applying Corollary 20, we obtain a tidy $(H, \mathcal{P})$-graph G with effectively the same twin-width as the input graph. Now we "only" need to show that the dangling paths in $\mathcal{P}$ can be shortened to length bounded by the input parameter without increasing the twin-width too much. As we noted earlier, there is no "local" way of shortening a dangling path (see Proposition 9).

Instead, our approach is based on establishing the existence of a $(\operatorname{tww}(G)+1)$-contraction sequence C^* for the trigraph G^* obtained from G by shortening its long paths; C^* is obtained by non-trivially repurposing a hypothetical optimal contraction sequence $C = (G_1 =$

$G, G_2, \ldots, G_n$) of G. Once we do that, we will have proven that our algorithm can produce a near-optimal contraction sequence for G by first shortening the paths (by contracting neighbors) and then, when the paths are as short as in G^*, by applying Observation 6. In other words, the complex machinery devised in this section is required for the correctness proof, while the algorithm itself is fairly simple.

Intuitively, the reason why it is difficult to go from C to C^* is that C has too much "freedom": it can perform arbitrary contractions between vertices of the dangling paths, whereas the only limitation is that the red degrees cannot grow too high (this was not an issue in Section 5, since having twin-width 2 places strong restrictions on how the dangling paths may interact). We circumvent this issue by not following C too closely when constructing C^*. Instead, we only look at a bounded number of special trigraphs in C, called *checkpoints* – forming the *big-step* contraction sequence defined later – and show that we can completely ignore what happened in C between two checkpoints when constructing C^*. Moreover, while checkpoints may be large and complicated, we identify for each checkpoint a small set of characteristics which will be sufficient to carry out our construction; we call this set the *blueprint* of the checkpoint, and it also includes the induced subtrigraph H_i of all descendants of H in a checkpoint G_i (the so-called *core*).

Our aim is to simulate the transition from one checkpoint to another via a "controlled" contraction sequence. For this purpose, we define *representatives* – trigraphs in C^* which match the blueprints of checkpoints in C – and show how to construct a partial contraction sequence from one representative to another in Subsection 6.3. In the end, these partial contraction sequences will be concatenated to create C^*.

A crucial gadget needed to define these representatives are *centipedes*; these are well-defined and precisely structured objects which simulate the (possibly highly opaque) connections between red components in the core, and are illustrated in Figure 5 later. Subsection 6.2 is dedicated to establishing the operations required to alter the structure and placement of centipedes between individual representatives. These operations rely on the fact that a vertex is allowed to have red degree 4; this is one reason why Theorem 2 produces sequences whose width may be one larger than the optimum (the other reason is that the possibility of supporting an additional red edge provides more flexibility when moving and altering the centipedes between checkpoints).

One final issue we need to deal with is that the paths in $\mathcal{P}$ must be sufficiently long in order to support the creation of the centipedes at the beginning of C^*. Fortunately, there is a simple way of resolving this: paths in $\mathcal{P}$ which are not long enough can be moved into H. However, the cost of this is that each time we add such a path into H, the size of H – and hence also the bound on the length of the paths in G^* – can increase by an exponential factor. For this reason, unlike in the previous section, the bikernel we obtain is not polynomial and not even elementary; its size will be bounded by a tower of exponents whose height is linear in the parameter.

6.1 Initial Setup

Recall that at this point, we are dealing with a tidy $(H, \mathcal{P})$-graph G. Let $C = (G_1 = G, G_2, \ldots, G_n)$ be a contraction sequence of G, and recall that H_i denotes the subtrigraph of G_i induced on the descendants of H and that we call vertices not in H_i *outer*.

We say that G_i is *decisive* if $H_i \neq H_{i-1}$ or $i = 1$. We define *the big-step contraction sequence* C_{BS} as a subsequence of C which contains G_i if and only if G_i or G_{i+1} is decisive. We call the trigraphs in C_{BS} *checkpoints*. Furthermore, we define $f_H : \mathbb{N} \to \mathbb{N}$ as follows:

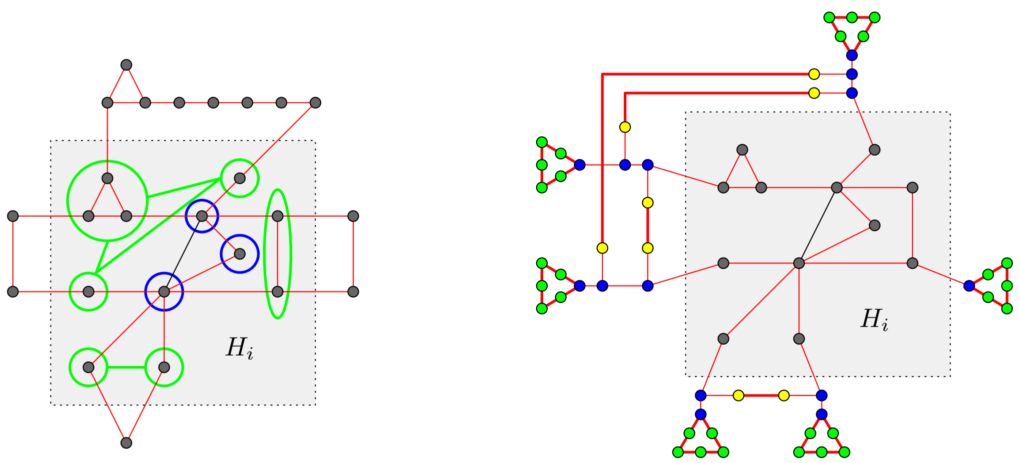

Figure 5 Left: A possible trigraph G_i. The components of H_i^R (i.e., vertices of B_i) are marked by green or blue circles: blue means isolated and green means non-isolated. The edges of B_i are depicted as green lines. **Right:** a representative for G_i. Blue vertices lie in the body of a centipede, yellow vertices form legs of centipedes, and green vertices lie in the tail of a centipede. Note that the tails and the leg paths, which are drawn as thick red lines, contain many vertices.

$f_H(\ell) = (3^{|V(H)|+4} \cdot |V(H)|^2)^\ell$. Informally, this function describes how big the centipedes will need to be in the ℓ-th trigraph in C_{BS}, counted from the end (the centipedes need to be the largest at the beginning, as they shrink during each transition between checkpoints).

Now we define the blueprints; these capture the information we need about the checkpoints.

▶ **Definition 27.** *The* blueprint *of a trigraph G_i in C, denoted $\mathcal{B}_i$, is the tuple (H_i, B_i) where B_i is the vertex-labeled graph constructed in the following way:*

- *let H_i^R be the subgraph obtained from H_i by removing every edge that is black or incident to a black edge;*
- *$V(B_i)$ is the set of connected components of H_i^R (the* fully-red components*);*
- *$U \in V(B_i)$ is labeled* isolated *if for all $u \in U$, we have $N_{G_i}(u) \subseteq V(H_i)$, and otherwise it is* non-isolated*;*
- *$UU' \in E$ if there is a (red) path between some vertices $u \in U$ and $u' \in U'$ in G_i which contains no vertex of $V(H_i)$ except for u and u'. $E(B_i)$ is then the transitive closure of E. Observe that isolated vertices have degree 0 in B_i but a non-isolated vertex may have degree 0, too; see Figure 5 for an illustration.*

A reader may wonder why we define the edge relation of B_i to be transitive. The reason is that with this definition, a connection can disappear only when H_i changes, see Lemma 32 below. We are now ready to define centipedes – the technical gadget underlying our entire construction.

▶ **Definition 28.** *Let $d \geq 0$ and $\ell \geq 1$. The* centipede *$cen(d, \ell)$ is the following graph:*

- *the vertex set consists of three disjoint sets:* the body *$\{u_i \mid i \in [d+1]\}$,* the legs *$\{v_i \mid i \in [d]\}$, and* the tail *$\{w_i \mid i \in [\ell]\}$;*
- *the edge set is $\{u_iu_{i+1}, u_iv_i \mid i \in [d]\} \cup \{w_iw_{i+1} \mid i \in [\ell - 1]\} \cup \{u_{d+1}w_1, u_{d+1}w_\ell\}$.*

We say that u_1 is the head of cen(d, ℓ)*. Let G' be a supergraph of $cen(d, \ell)$. We say that a leg v of $cen(d, \ell)$ is* free in G' *if its degree is 1 in G'. For $x \in V(G')$, we say that $cen(d, \ell)$ is* attached to x *if x is adjacent to the head of $cen(d, \ell)$.*

Now we define the representatives; trigraphs representing the blueprints in C'. Note that ℓ specifies how big the centipedes in the representative need to be.

▶ **Definition 29.** *Let $\mathcal{B}_i = (H_i, B_i)$ be a blueprint and ℓ be an integer. Now a* representative for $\mathcal{B}_i$ of order ℓ*, denoted R_i^ℓ, is a trigraph that can be built as follows:*

1. *start with H_i; let $\mathcal{U} \subseteq V(B_i)$ be the set of non-isolated vertices of B_i;*
2. *for each $U \in \mathcal{U}$, add a red centipede $\Psi_U \cong cen(\deg_{B_i}(U), f_H(\ell))$;*
3. *for all $U \in \mathcal{U}$, add a red edge between the head of Ψ_U and some vertex of U which has at least one neighbor outside of H_i in G_i (there must be at least one such vertex because U is non-isolated);*
4. *for each edge $UU' \in E(B_i)$, do the following:*
   - *let w (resp. w') be a leg of Ψ_U (resp. $\Psi_{U'}$) free in the current trigraph. Add a red path of length $f_H(\ell)$ connecting w and w'. We call this path, including the two legs,* the leg path connecting Ψ_U and $\Psi_{U'}$.

Notice that the construction in Definition 29 works by a simple inductive argument because the number of legs of Ψ_U is $\deg_{B_i}(U)$ by construction; an illustration is provided in Figure 5. Moreover, observe that a representative is not uniquely determined by i and ℓ; the order of leg paths, as well as the vertices which the centipedes are attached to, can differ.

6.2 Moving Centipedes Around

In this subsection, we describe several operations, i.e., partial contraction sequences, which will later be used to obtain a partial contraction sequence which transitions from a representative of one checkpoint in C_{BS} to a representative of the next checkpoint. These operations will focus on the centipedes introduced in the previous subsection, and may result in trigraphs which are not necessarily a representative for any graph in C; we refer to these obtained trigraphs as *intermediate graphs* and note that their structure can be precisely formalized.

We start with a few simple operations on generalized intermediate graphs:

▶ **Observation 30** (Shortening a centipede)**.** *We can* shorten the tail *of a centipede Ψ, by contracting two neighboring vertices belonging to the tail, to any length. Similarly, we can* shorten a leg path *to any non-zero length.*

▶ **Observation 31** (Destroying a centipede)**.** *Let G' be an intermediate graph, let $u \in V(H')$, and let $\Psi = cen(0, \ell)$ be a centipede attached to u. We can* destroy Ψ*, i.e., there is a partial contraction sequence of width at most* $\operatorname{tww}(G) + 1$ *from G' to $G' - V(\Psi)$.*

The remaining operations allow us to

1. reorder the legs of a centipede,
2. move a centipede to a different vertex of the core,
3. connect two centipedes by a new leg path,
4. merge two centipedes into a single centipede, and
5. split a centipede into two new centipedes.

Each of these operations can be formalized by carefully prescribing the initial and final intermediate graph, the impact on the length of the involved centipedes, and a proof ensuring that the contraction subsequence between the initial and final intermediate graph has width at most $\operatorname{tww}(G) + 1$.

6.3 Contraction Sequences for Representatives

In this subsection, we will construct partial contraction sequences between the representatives of consecutive checkpoints by making use of the operations with the centipedes defined in Subsection 6.2. We distinguish between two cases depending on whether the core changes between the checkpoints or not, see Lemmas 33 and 34 below.

We start by observing that if the core does not change between two consecutive checkpoints, then the blueprint of the latter checkpoint contains all edges present in the blueprint of the former checkpoint.

▶ **Lemma 32.** *Let G_i and G_j be two consecutive trigraphs in C_{BS} such that $H_i = H_j$. This means that $V(B_i) = V(B_j)$, see Definition 27. It holds that $E(B_i) \subseteq E(B_j)$.*

We are now ready to define the partial contraction sequence between two representatives in the case when the core does not change.

▶ **Lemma 33.** *Let G_i, G_j $(i < j)$ be two consecutive trigraphs in C_{BS}, such that $H_i = H_j$. For any $\ell \in \mathbb{N}$, there is a partial contraction sequence from $R_i^{\ell+1}$ to R_j^{ℓ} whose width is at most* $\operatorname{tww}(G) + 1$.

Proof Sketch. Even though the number of contraction happening between G_i and G_j in C can be huge, the effect on the blueprints (and thus the representatives) is somewhat limited. After checking what could differ between the two representatives, we present a sequence of operations on the centipedes – namely moving, creating, connecting, shortening and destroying centipedes – which is sufficient to obtain R_j^{ℓ} from $R_i^{\ell+1}$.

To prove that this sequence of operation on centipedes is feasible, we verify that the tails and leg paths are sufficiently longer in $R_i^{\ell+1}$ than in R_j^{ℓ} to sustain the operations without becoming too short. Moreover, the control we have over the operations enables us to make sure that no vertex has a red degree higher than $tww(G) + 1$ at any point in the created contraction sequence. ◀

Next, we define the partial contraction sequence between two representatives in the second and final case, namely the case when the core does change. Note that in this case, we allow the sequence to terminate in a slightly different trigraph (which will be handled by Lemma 35).

▶ **Lemma 34.** *Let G_i, G_j $(i < j)$ be two consecutive trigraphs in C_{BS} such that $H_i \neq H_j$. For any $\ell \in \mathbb{N}$, there is a partial contraction sequence C^p from $R_i^{\ell+1}$ to a trigraph that is a pseudoinduced subtrigraph of R_j^{ℓ} such that $w(C^p) \leq \operatorname{tww}(G) + 1$.*

We now combine the previous two lemmas to obtain a contraction sequence of the representative for G_1 – the first trigraph in C_{BS} as well as in C. More generally:

▶ **Lemma 35.** *Let G_i be a trigraph in C_{BS} such that there are ℓ trigraphs after G_i in C_{BS}. There is a contraction sequence of R_i^{ℓ} whose width is at most* $\operatorname{tww}(G) + 1$.

One more thing we need to do is initialization: the trigraph we are interested in, i.e., the trigraph obtained from G by shortening all dangling paths to bounded length, does not contain any centipedes. This is handled by (the proof of) Theorem 36 below, which also summarizes the outcome of this subsection.

▶ **Theorem 36.** *Let G be a tidy $(H, \mathcal{P})$-graph such that* $\operatorname{tww}(G) \geq 3$ *and all paths in $\mathcal{P}$ have length at least $3 \cdot f_H(|2V(H)|^2) + 9$ and let $G' = (H, \mathcal{P}')$ be any trigraph obtained from G by shortening paths in $\mathcal{P}$ to arbitrary lengths no shorter than $3 \cdot f_H(2|V(H)|^2) + 9$. Then* $\operatorname{tww}(G') \leq \operatorname{tww}(G) + 1$.

6.4 Putting Everything Together

We are now ready to prove Theorem 2.

▶ **Theorem 2.** *There is an algorithm which takes as input an n-vertex graph G with feedback edge number k, runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ for a computable function f, and outputs a contraction sequence for G of width at most* $\text{tww}(G) + 1$*.*

Proof Sketch. We begin by handling the case where $\text{tww}(G) \leq 2$ by invoking Theorems 8 and 1. For the rest of the proof, we assume $\text{tww}(G) \geq 3$. Here, we first use Theorem 17 to get in $n^{\mathcal{O}(1)}$ time an original $(H, \mathcal{P})$-graph such that $|V(H)| \leq 16k$ and $|\mathcal{P}| \leq 4k$. Recall that this $(H, \mathcal{P})$-graph has effectively the same twin-width as G, so any optimal contraction sequence for it can be lifted to an optimal one for G. Using Corollary 20, we obtain – also in $n^{\mathcal{O}(1)}$ time – a tidy $(H', \mathcal{P}')$-graph G' such that $|V(H')| \leq 112k$, $|\mathcal{P}'| \leq 4k$, and G' still has effectively the same twin-width as G.

At this point, we check the length of each path in $\mathcal{P}'$, whereas if we identify a path $P \in \mathcal{P}'$ whose length is below the bound required by Theorem 36 (w.r.t. the current size of H'), we add P into H' and update our choices of H' and $\mathcal{P}'$ accordingly. After exhaustively completing the above check, we are guaranteed to have satisfied the conditions of Theorem 36. We now begin constructing our contraction sequence for G' as follows. First, we iteratively contract the paths which remain in $\mathcal{P}'$ until they have length precisely $3 \cdot f_{H'}(2|V(H')|^2) + 9$; recall that by Theorem 36, we are guaranteed that the resulting graph G^* has twin-width at most one larger than G (and also G'). Moreover, the number of vertices in G^* can be upper-bounded by a non-elementary function of our parameter, specifically $2^{2^{\cdot^{\cdot^{\cdot^{2^{\mathcal{O}(log(k))}}}}}}$ where the height of the tower of exponents is upper-bounded by $4k + 3$. At this point, we apply Observation 6 to construct an optimal contraction sequence of G^* and append it after the trivial sequence of contractions which produced G^*. The proof now follows by the fact that G' has effectively the same twin-width as G. ◀

7 Concluding Remarks

While the feedback edge number parameterization employed by our algorithms is highly restrictive, we believe Theorems 1 and 2 represent a tangible and important first step towards more general algorithms for computing near-optimal contraction sequences, with the "holy grail" being a fixed-parameter algorithm for computing near-optimal contraction sequences parameterized by twin-width itself. The natural next goals in this line of research would be to obtain fixed-parameter algorithms for the problem when parameterized by treedepth [31] and then by treewidth [33].

Towards this direction, we note that it is not at all obvious how one could apply classical tools such as *typical sequences* [7, 20, 8] in the context of computing contraction sequences. At least for treedepth, it may be possible to employ the general approach developed in Section 6 – in particular, establishing the existence of a near-optimal but "well-structured" contraction sequence and using that to identify safe reduction rules – but the details and challenges arising there seem to differ significantly from the ones handled in this article.

Last but not least, we remark that the algorithms developed here rely on reduction rules which are provably safe, simple to implement, and run in polynomial time; we believe these may potentially be of interest for heuristic and empirical purposes. We also believe that the additive error of 1 incurred by Theorem 2 is avoidable, albeit this may perhaps be seen as a less pressing question than settling the approximability of twin-width under the parameterizations outlined in the previous paragraph.

References

1 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.

2 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.

3 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.

4 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.

5 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.

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

7 Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995. doi:10.1006/jagm.1995.1009.

8 Hans L. Bodlaender, Lars Jaffke, and Jan Arne Telle. Typical sequences revisited - computing width parameters of graphs. *Theory Comput. Syst.*, 67(1):52–88, 2023. doi:10.1007/s00224-021-10030-3.

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, 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.

14 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.

15 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant. Twin-width and polynomial kernels. *Algorithmica*, 84(11):3300–3337, 2022. doi:10.1007/s00453-022-00965-5.

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 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.

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

19 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.

20 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.

21 David Eppstein. The widths of strict outerconfluent graphs. *CoRR*, abs/2308.03967, 2023. arXiv:2308.03967.

22 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.

23 Fedor V. Fomin and Tuukka Korhonen. Fast fpt-approximation of branchwidth. 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 886–899. ACM, 2022. doi:10.1145/3519935.3519996.

24 Robert Ganian and Petr Hlinený. On parse trees and myhill-nerode-type tools for handling graphs of bounded rank-width. *Discret. Appl. Math.*, 158(7):851–867, 2010. doi:10.1016/j.dam.2009.10.018.

25 Robert Ganian and Viktoriia Korchemna. The complexity of bayesian network learning: Revisiting the superstructure. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 430–442, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/040a99f23e8960763e680041c601acab-Abstract.html.

26 Robert Ganian and Sebastian Ordyniak. The power of cut-based parameters for computing edge-disjoint paths. *Algorithmica*, 83(2):726–752, 2021. doi:10.1007/s00453-020-00772-w.

27 Robert Ganian, Filip Pokrývka, André Schidler, Kirill Simonov, and Stefan Szeider. Weighted model counting with twin-width. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPIcs*, pages 15:1–15:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SAT.2022.15.

28 Petr Hlinený and Sang-il Oum. Finding branch-decompositions and rank-decompositions. *SIAM J. Comput.*, 38(3):1012–1032, 2008. doi:10.1137/070685920.

29 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.

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

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

32 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`.

33 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`.

34 André Schidler and Stefan Szeider. Computing twin-width with SAT and branch & bound. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 2013–2021. ijcai.org, 2023. `doi:10.24963/ijcai.2023/224`.

35 Johannes Uhlmann and Mathias Weller. Two-layer planarization parameterized by feedback edge set. *Theor. Comput. Sci.*, 494:99–111, 2013. `doi:10.1016/j.tcs.2013.01.029`.