Deciding Twin-Width at Most 4 Is NP-Complete

Pierre Bergé
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Édouard Bonnet
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Hugues Déprés
Univ Lyon, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP UMR5668, France

Abstract

We show that determining if an n-vertex graph has twin-width at most 4 is NP-complete, and requires time $2^{\Omega(n/\log n)}$ unless the Exponential-Time Hypothesis fails. Along the way, we give an elementary proof that n-vertex graphs subdivided at least $2\log n$ times have twin-width at most 4. We also show how to encode trigraphs H (2-edge colored graphs involved in the definition of twin-width) into graphs G, in the sense that every d-sequence (sequence of vertex contractions witnessing that the twin-width is at most d) of G inevitably creates H as an induced subtrigraph, whereas there exists a partial d-sequence that actually goes from G to H. We believe that these facts and their proofs can be of independent interest.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Fixed parameter tractability

Keywords and phrases Twin-width, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.18

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2112.08953`

Funding This paper was supported by the ANR projects TWIN-WIDTH (ANR-21-CE48-0014-01) and Digraphs (ANR-19-CE48-0013-01).

Acknowledgements We wish to thank Eunjung Kim, Stéphan Thomassé, and Rémi Watrigant.

1 Introduction

A *trigraph* is a graph with some edges colored black, and some colored red. A (vertex) *contraction* consists of merging two (non-necessarily adjacent) vertices, say, u, v into a vertex w, and keeping every edge wz black if and only if uz and vz were previously black edges. The other edges incident to w become red (if not already), and the rest of the trigraph stays the same. A *contraction sequence* of an n-vertex graph G is a sequence of trigraphs $G = G_n, \ldots, G_1 = K_1$ such that G_i is obtained from G_{i+1} by performing one contraction. A *d-sequence* is a contraction sequence where all the trigraphs have red degree at most d. The *twin-width* of G, denoted by $tww(G)$, is then the minimum integer d such that G admits a d-sequence. See Figure 1 for an example of a graph admitting a 2-sequence. The *red graph* of a trigraph is obtained by simply deleting its black edges. A *partial d-sequence* is similar to a d-sequence but ends on any trigraph G_i, instead of on the 1-vertex (tri)graph G_1. Twin-width can be naturally extended to matrices over a finite alphabet (in an unordered [6], or an ordered setting [4]), and hence to any binary structure.

Surprisingly many classes turn out to be of bounded twin-width. Such is the case of graphs with bounded clique-width, H-minor free graphs for any fixed H, posets with antichains of bounded size, strict subclasses of permutation graphs, map graphs, bounded-degree string

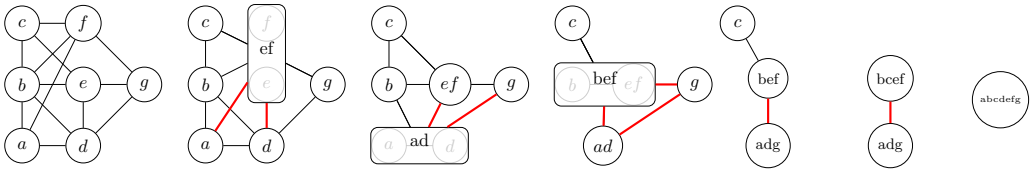

Figure 1 A 2-sequence witnessing that the initial graph has twin-width at most 2.

graphs [6], as well as $\Omega(\log n)$-subdivisions of n-vertex graphs, and some classes of cubic expanders [3]. One of the main algorithmic interests with twin-width is that first-order (FO) model checking, that is, deciding if a first-order sentence φ holds in a graph G, can be decided in fixed-parameter time (FPT) $f(|\varphi|, d) \cdot |V(G)|$ for some computable function f, when given a d-sequence of G [6]. As for most classes known to have bounded twin-width, one can compute $O(1)$-sequences in polynomial time for members of the class, the latter result unifies and extends several known results [14, 17, 18, 16, 21] for hereditary (but not necessarily monotone) classes.

For monotone (i.e., subgraph-closed) classes, the FPT algorithm of Grohe, Kreutzer, and Siebertz [20] for FO model checking on nowhere dense classes, is complemented by W[1]-hardness on classes that are somewhere dense (i.e., *not* nowhere dense) [11], and even AW[∗]-hardness on classes that are *effectively* somewhere dense [9]. The latter results mean that, for monotone classes, FO model checking is unlikely to be FPT beyond nowhere dense classes.

The missing piece for an FO model-checking algorithm in FPT time on any class of bounded twin-width is a polynomial-time algorithm and a computable function f, that given a constant integer bound c and a graph G, either finds an $f(c)$-sequence for G, or correctly reports that the twin-width of G is greater than c. The running time of the algorithm could be $n^{g(c)}$, for some function g. However to get an FPT algorithm in the combined parameter *size of the sentence + bound on the twin-width*, one would further require that the approximation algorithm takes FPT time in c (now seen as a parameter), i.e., $g(c)n^{O(1)}$. We know such an algorithm for instance on ordered graphs (more generally, ordered binary structures) [4], graphs of bounded clique-width, proper minor-closed classes [6], but not in general graphs.

On the other hand, prior to this paper, no algorithmic lower bound was known for computing the twin-width. Our main result rules out an (exact) XP algorithm to decide $tww(G) \leqslant k$, that is, an algorithm running in time $n^{f(k)}$ for some computable function f. Indeed we show that deciding if the twin-width of a graph is at most 4 is intractable. We refer the reader to Section 2 for some context on the Exponential-Time Hypothesis (ETH), which implies that n-variable 3-SAT cannot be solved in time $2^{o(n)}$.

▶ **Theorem 1.** *Deciding if a graph has twin-width at most 4 is NP-complete. Furthermore, no algorithm running in time $2^{o(n/\log n)}$ can decide if an n-vertex graph has twin-width at most 4, unless the ETH fails.*

As far as approximation algorithms are concerned, our result only rules out a ratio better than 5/4 for determining the twin-width. This still leaves plenty of room for an $f(\mathrm{OPT})$-approximation, which would be good enough for most of the (theoretical) algorithmic applications. Note that such algorithms exist for treewidth in polytime [12] and FPT time [27], for pathwidth [12], and for clique-width via rank-width [30].

Is Theorem 1 surprising? On the one hand, it had to be expected that deciding, given a graph G and an integer k, whether $tww(G) \leqslant k$ would be NP-complete. This is the case for example of treewidth [1], pathwidth [29, 26, 28], clique-width [13], rank-width [23],

mim-width [32], and bandwidth [19]. On the other hand, the parameterized complexity of these width parameters is more diverse and harder to predict. Famously, Bodlaender's algorithm is a linear FPT algorithm to exactly compute treewidth [2] (and a non-uniform FPT algorithm came from the Graph Minor series [31]). In contrast, while there is an XP algorithm to compute bandwidth [33], an FPT algorithm is highly unlikely [10]. It is a long-standing open whether an FPT or a mere XP algorithm exist for computing clique-width exactly, or even simply if one can recognize graphs of clique-width at most 4 in polynomial time (deciding clique-width at most 3 is indeed tractable [7]).

Theorem 1 almost completely resolves the parameterized complexity of exactly computing twin-width on general graphs. Two questions remain: can graphs of twin-width at most 2, respectively at most 3, be recognized in polynomial time. Graphs of twin-width 0 are cographs, which can be recognized in linear time [22], while it was recently shown that graphs of twin-width at most 1 can be recognized in polynomial time [5]. In the course of establishing Theorem 1 we show and generalize the following, where an $(\geqslant s)$-*subdivision* of a graph is obtained by subdividing each of its edges at least s times.

▶ **Theorem 2.** *Any $(\geqslant 2\log n)$-subdivision of an n-vertex graph has twin-width at most 4.*

That those graphs have bounded twin-width was known [3], but not with the explicit bound of 4. Another family of graphs with twin-width at most 4 is the set of grids, walls, their subgraphs, and subdivisions. Even if there is no proof of that fact, sufficiently large grids, walls, or their subdivisions likely have twin-width *at least* 4; it is actually surprising that the 6×8 grid still has twin-width 3 [34]. We also believe that long subdivisions of "sufficiently complicated" graphs have twin-width *at least* 4. That would make graphs of twin-width at most 3 considerably simpler than those of twin-width at most 4, especially among sparse graphs.

Contrary to the hardness proof for treewidth [1], which involves some structural characterizations by chordal completions, and the intermediate problems Minimum Cut Linear Arrangement, Max Cut, and Max 2-SAT [19], our reduction is "direct" from 3-SAT. This makes the proven hardness of twin-width more robust, and easier to extend to restricted classes of graphs, especially sparse ones. Theorem 1 holds for bounded-degree input graphs. For instance, performing our reduction from Planar 3-SAT produces subgraphs of constant powers of the planar grid (while admittedly weakening the ETH lower bound from $2^{\Omega(n/\log n)}$ to $2^{\Omega(\sqrt{n}/\log n)}$). Hence, while the complexity status of computing treewidth on planar graphs is a famous long-standing open question, one can probably extend the NP-hardness of *twin-width at most 4* to planar graphs, by tuning and/or replacing the few non-planar gadgets of our reduction.

Let us point out that, in contrast to subset problems, there is no $2^{O(n)}$-time algorithm known to compute twin-width. The exhaustive search takes time $n^{2n+O(1)}$ by considering all sequences of $n-1$ pairs of vertices. We leave as an open question whether the ETH lower bound of computing twin-width can be brought from $2^{\Omega(n/\log n)}$ to $2^{\Omega(n)}$, or even $2^{\Omega(n\log n)}$. The latter lower bound is known to hold for Subgraph Isomorphism [8] (precisely, given a graph H and an n-vertex graph G, deciding if H is isomorphic to a subgraph of G requires time $2^{\Omega(n\log n)}$, unless the ETH fails), or computing the Hadwiger number [15] (i.e., the size of the largest clique minor).

1.1 Outline of the proof of Theorem 1

We propose a quasilinear reduction from 3-SAT. Given an n-variable instance I of 3-SAT, we shall construct an $O(n\log n)$-vertex graph $G = G(I)$ which has twin-width at most 4 if and only if I is satisfiable.

Half of our task is to ensure that no 4-sequence will exist if I is unsatisfiable. This is challenging since many contraction strategies are to be considered and addressed. We make this task more tractable by attaching *fence gadgets* to some chosen vertex subsets. The effect of the fence *enclosing* S is that no contraction can involve vertices in S with vertices outside of S, while S is not contracted into a single vertex. The *maximal or outermost* fences (we may nest two or more fence gadgets) partition the rest of the vertices. This significantly tames the potential 4-sequences of G.

Our basic building block, the *vertical set*, consists of a pair of vertices (*vertical pair*) enclosed by a fence. It can be thought of as a bit set to 0 as long as the pair is not contracted, and to 1 when the pair gets contracted. It is easy to assemble vertical sets as prescribed by an auxiliary digraph D (of maximum degree 3), in such a way that, to contract (by a partial 4-sequence) the pair of a vertical set V, one first has to contract all the vertical sets that can reach V in D. This allows to propagate and duplicate a bit in a so-called *wire* (corresponding to an out-tree in D), and to perform the logical AND of two bits.

The bit propagation originates from a variable gadget (we naturally have one per variable appearing in I) that offers two alternatives. One can contract the "top half" of the gadget of variable x_i, which then lets one contract the vertical sets in the wire of literal x_i, or one can contract instead the "bottom half" of the gadget, as well as the vertical sets in the wire of literal $\neg x_i$. Concretely, these two contraction schemes represent the two possible assignments for variable x_i. A special "lock" on the variable gadget (called *half-guards*) prevents its complete contraction, and in particular, performing contractions in both the wires of a literal and its negation.

The leaves of the literal wires serve as inputs for 3-clause gadgets. One can contract the output (also a vertical set) of a clause gadget if and only if one of its input is previously contracted. We then progressively make the AND of the clauses via a "path" of binary AND gadgets fed by the clause outputs. We eventually get a vertical set, called *global output*, which can be contracted by a partial 4-sequence only if I is satisfiable. Indeed at this point, the variable gadgets are still locked so at most one of their literals can be propagated. This ticks one of our objective off. We should now ensure that a 4-sequence is possible from there, when I is satisfiable.

For that purpose, we add a wire from the global output back to the half-guards (or locks) of the variable gadgets. One can contract the vertical sets of that wire, and in particular the half-guards. Once the variable gadgets are "unlocked," they can be fully contracted. As a consequence, one can next contract the wires of literals set to false, and *all* the remaining vertical sets involved in clause gadgets.

At this point, the current trigraph H roughly has one vertex per outermost fence with red edges linking two adjacent gadgets (and no black edge). We will guarantee that the (red) degree of H is at most 4, its number of vertices of degree at least 3 is at most βn, for some constant β. Besides we will separate gadgets by degree-2 wires of length $2\log(\beta n)$ beforehand. This is crucial so that the red graph of H is a $(2\log n')$-subdivision of an n'-vertex graph. We indeed show that such trigraphs have twin-width at most 4. A complicated proof in [3] shows that $\Theta(\log n')$-subdivisions of n'-vertex graphs have bounded twin-width. Here we give an elementary proof of a similar fact with an explicit upper bound of 4.

This finishes to describe our overall plan for the reduction and its correctness. It happens that fence gadgets are easier to build as trigraphs, while the rest of the gadgetry can be directly encoded by graphs. We thus show how to encode trigraphs by graphs, as follows. For any trigraph J whose red graph has degree at most d, and component size at most h,

there is a graph G on at most $f(d,h)\cdot|V(J)|$ vertices such that J has twin-width at most $2d$ if and only if G has twin-width at most $2d$. This uses some local replacements and confluence properties of certain partial contraction sequences.

The proofs of the theorems and lemmas marked with a $\star$ can be found in the full version.

2 Preliminaries

For i and j two integers, we denote by $[i,j]$ the set of integers that are at least i and at most j. For every integer i, $[i]$ is a shorthand for $[1,i]$. We use the standard graph-theoretic notations: $V(G)$ denotes the vertex set of a graph G, $E(G)$ denotes its edge set, $G[S]$ denotes the subgraph of G induced by S, etc. An $(\geqslant s)$-subdivision (resp. s-subdivision) of a graph G is obtained by subdividing every edge of G at least s times (resp. exactly s times).

2.1 Definitions and notations related to twin-width

A *trigraph* G has vertex set $V(G)$, black edge set $E(G)$, red edge set $R(G)$. Its *red graph* $(V(G),R(G))$ may be denoted $\mathcal{R}(G)$. The *red degree* of a trigraph is the degree of its red graph. We say that $u\in V(G)$ is a *black neighbor* (respectively *red neighbor*) of $v\in V(G)$ when $(u,v)\in E(G)$ (respectively $(u,v)\in R(G)$). A trigraph G' is an *induced subtrigraph* of trigraph G if $V(G')\subseteq V(G)$, $E(G')=E(G)\cap\binom{V(G')}{2}$, and $R(G')=R(G)\cap\binom{V(G')}{2}$. Then we say that G is a supertrigraph of G', and we may also denote G' by $G[V(G')]$. A *(partial) d-sequence* of a (tri)graph G is a (partial) contraction sequence starting at G and admitting trigraphs of red degree at most d.

The *twin-width* of a graph, introduced in [6], can be defined in the following way. A *partition sequence* of an n-vertex graph G, is a sequence $\mathcal{P}_n,\dots,\mathcal{P}_1$ of partitions of its vertex set $V(G)$, such that $\mathcal{P}_n$ is the set of singletons $\{\{v\} : v\in V(G)\}$, $\mathcal{P}_1$ is the singleton set $\{V(G)\}$, and for every $2\leqslant i\leqslant n$, $\mathcal{P}_{i-1}$ is obtained from $\mathcal{P}_i$ by merging two of its parts into one. Two parts P,P' of a same partition $\mathcal{P}$ of $V(G)$ are said *homogeneous* if either every pair of vertices $u\in P, v\in P'$ are non-adjacent, or every pair of vertices $u\in P, v\in P'$ are adjacent. Finally the twin-width of G, denoted by $tww(G)$, is the least integer d such that there is partition sequence $\mathcal{P}_n,\dots,\mathcal{P}_1$ of G with each part P of each $\mathcal{P}_i$ $(1\leqslant i\leqslant n)$ being homogeneous to every part of $\mathcal{P}_i\setminus\{P\}$ but at most d.

The reason we gave two (equivalent) definitions of twin-width is that both viewpoints are incomparably useful and convenient. To navigate between these two worlds, we use the following notations and vocabulary.

Assume there is a partial contraction sequence from (tri)graph G to trigraph H. If u is a vertex of H, then $u(G)$ denotes the set of vertices eventually contracted into u in H. We denote by $\mathcal{P}(H)$ the partition $\{u(G) : u\in V(H)\}$ of $V(G)$. If G is clear from the context, we may refer to a *part* of H as any set in $\{u(G) : u\in V(H)\}$. We say that a contraction of two vertices $u,u'\in V(H)$ *involves* a vertex $v\in V(G)$ if $v\in u(G)$ or $v\in u'(G)$. A contraction *involves* a pair of vertices v,v' if $v\in u(G)$ and $v'\in u'(G)$ (or $v\in u'(G)$ and $v'\in u(G)$). A contraction *involves* a set S, if it involves a vertex of S, or a pair of sets S,T if it involves a pair in $S\times T$.

2.2 Useful observations

The twin-width can only decrease when taking induced subgraphs or turning red into black.

▶ **Observation 3.** *Let G' be an induced subtrigraph of trigraph G. Then $tww(G')\leqslant tww(G)$.*

▶ **Observation 4.** *Let G be a trigraph and G' another trigraph obtained from G by turning some non-edges and some edges into red edges. Then $tww(G') \geqslant tww(G)$.*

Trees admit a simple 2-sequence, that gives a d-sequence on red trees of degree at most d.

▶ **Lemma 5** ([6]). *Every (black) tree has twin-width at most 2. Every red tree has twin-width at most its maximum degree.*

2.3 The Exponential-Time Hypothesis

The Exponential-Time Hypothesis (ETH) was proposed by Impagliazzo and Paturi [24] and asserts that there is no subexponential-time algorithm solving 3-SAT. More precisely, there is an $\varepsilon > 0$ such that n-variable 3-SAT cannot be solved in time $2^{\varepsilon n}$. A classic reduction [35] linear in the number of clauses, and the Sparsification Lemma [25] imply that:

▶ **Theorem 6** ([35, 25]). *The n-variable 3-SAT problem where each variable appears at most twice positively, and at most twice negatively, is NP-complete, and cannot be solved in time $2^{o(n)}$, unless the ETH fails.*

3 Long subdivisions have twin-width at most four

In [3], it is proved that the $\Omega(\log n)$-subdivision of any n-vertex graph has bounded twin-width. The proof is rather involved, relies on a characterization by *mixed minors* established in [6], and does not give an explicit constant bound. Here we give an elementary proof that any ($\geqslant 2\lceil \log n \rceil - 1$)-subdivision of an n-vertex graph has twin-width at most 4.

▶ **Theorem 7.** *Let G be a trigraph obtained by subdividing each edge of an n-vertex graph H at least $2\lceil \log n \rceil - 1$ times, and by turning red any subset of its edges as long as the red degree of G remains at most 4, and no vertex with red degree 4 has a black neighbor. Then $tww(G) \leqslant 4$.*

Proof. By no more than doubling the number of vertices of H, we can assume that n is a power of 2. Indeed, padding H with isolated vertices up to the next power of 2 does not change the quantity $\lceil \log |V(H)| \rceil$.

Let G' be a supertrigraph of G obtained by arbitrarily arranging the vertices of H (in G) at the leaves of a "virtual" full binary tree of height $\log n$. So that the red degree does not exceed 4, we so far omit the edges of the tree incident to a leaf (i.e., a vertex of H), while we

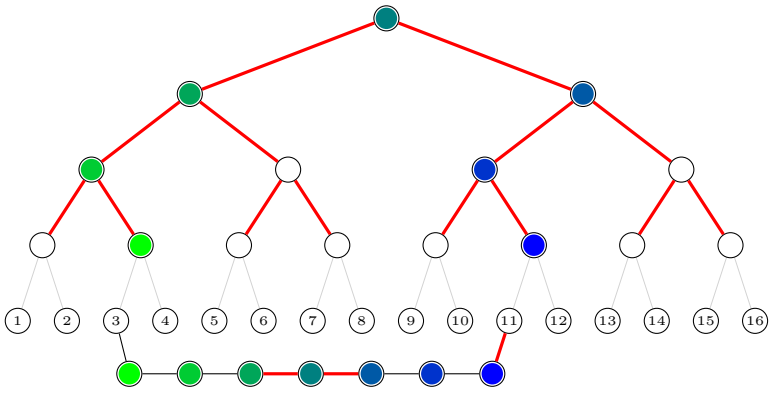

Figure 2 Contracting the pairs of vertices with the same color, from the greenest to the bluest, is a partial 4-sequence, which acts as a deletion of the subdivided edge $(3, 11)$.

put in red all the other edges of the tree. (The missing edges of the tree will naturally appear in red.) The internal nodes of the tree are all fresh vertices, not present in G. See Figures 2 and 3 for an illustration. We show that $tww(G') \leqslant 4$, hence by Observation 3, $tww(G) \leqslant 4$ since G is an induced subtrigraph of G'.

We label $1, 2, \ldots, n$ the vertices of H. If there is an edge $ij \in E(H)$, it is subdivided into a path, say, $i, s(ij,1), s(ij,2), \ldots, s(ij,z), j$ in G with $z \geqslant 2\log n - 1$. First, we repeatedly contract adjacent vertices in the middle of this path until it consists of exactly $2\log n$ edges. If $z > 2\log n - 1$, we had to contract at least one pair of adjacent vertices. Thus the vertex in the middle of the path necessarily has now two red edges incident to it. Note that the other edges of the path can be black or red indifferently. To avoid cumbersome notations, we rename the inner vertices of the path $s(ij,1), s(ij,2), \ldots, s(ij,z)$ with now $z = 2\log n - 1$.

▷ Claim 8. There is a partial 4-sequence from G' to $G' - \{s(ij,1), \ldots, s(ij,z)\}$.

Proof. Intuitively we "zip" the subdivision of ij with the walk made by the union of the path from leaf i to the root, and the path from the root to leaf j. Let $i, v_1, v_2, \ldots, v_z, j$ be the concatenation of the simple path from i to the root of the tree, and the simple path from the root to j. Its length is thus $2\log n = z + 1$. For h going from 1 to $z = 2\log n - 1$, we contract v_h and $s(ij,h)$ (see Figure 2). After each contraction, the newly formed vertex has red degree at most 4. The red degree of vertices that are neither the new vertex nor a leaf of the tree is either unchanged or at most 2. The red degree of a leaf ℓ of the tree may increase by 1. This may only happen the first time a neighbor of ℓ is involved in a contraction, and that contraction merges a black neighbor of ℓ with the parent of ℓ in the tree (like is the case for leaf 3 from Figure 2 to Figure 3). By assumption, this implies that ℓ had red degree at most 3, thus its red degree does not exceed 4. Thus, what we defined is indeed a partial 4-sequence. One can finally notice that after these z contractions, we indeed reach trigraph $G' - \{s(ij,1), \ldots, s(ij,z)\}$. ◁

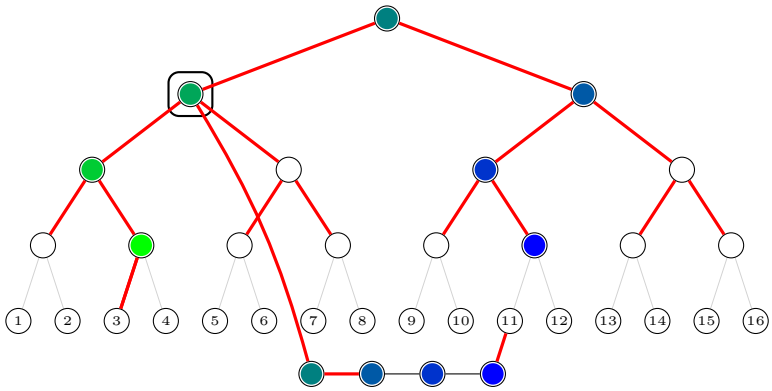

Figure 3 The picture after the first three contractions. The newly formed vertex has red degree 4.

We apply Claim 8 for each edge of H (or rather, subdivided edge in G). We are then left with a red full binary tree which admits a 3-sequence by Lemma 5. Hence there is a 4-sequence for G', and in particular, for G. ◀

We will only use the following consequence.

▶ **Lemma 9.** *Let G be a trigraph obtained by subdividing at least $2\lceil \log n \rceil - 1$ times each edge of an n-vertex graph H of degree at most 4, and by turning red all its edges. Then $tww(G) \leqslant 4$.*

4 Hardness of determining if the twin-width is at most four

Here we show the main result of the paper.

▶ **Theorem 1.** *Deciding if a graph has twin-width at most 4 is NP-complete. Furthermore, no algorithm running in time $2^{o(n/\log n)}$ can decide if an n-vertex graph has twin-width at most 4, unless the ETH fails.*

The membership to NP is ensured by the d-sequence: a polynomial-sized certificate that a graph has twin-width at most d, checkable in polynomial time. We thus focus on the hardness part of the statement, and design a quasilinear reduction from 3-SAT.

4.1 Foreword to the reduction

Let us start by a construction allowing to encode trigraphs into (plain) graphs. Given a trigraph H with red degree at most d, we can produce a graph G such that H admits a $2d$-sequence iff G admits a $2d$-sequence.

▶ **Lemma 10** ($\star$)**.** *Given any trigraph H whose red graph has degree at most d and connected components of size at most h, one can compute in time $O_{d,h}(|V(H)|)$ a graph G on $O_{d,h}(|V(H)|)$ vertices such that H has a $2d$-sequence if and only if G has a $2d$-sequence.*

In what follows, we only need the following scaled-down version.

▶ **Lemma 11.** *Given any trigraph H whose red graph is a disjoint union of 12-vertex paths and isolated vertices, one can compute in polynomial time a graph G on $O(|V(H)|)$ vertices such that H has twin-width at most 4 if and only if G has twin-width at most 4.*

Our task is now slightly simpler. Given a 3-SAT instance I, we may design a *trigraph* satisfying the requirements of Lemma 11 with twin-width at most 4 if and only if I is satisfiable.

We will motivate all the gadgets along the way, by exhibiting key properties that they impose on a potential 4-sequence. These properties readily lead to a satisfying assignment for I. We also describe partial 4-sequences to reduce most of the gadgets. However some preconditions (specifying the context in which a particular gadget stands) tend to be technical, and make more sense after the construction of G. In those cases, to avoid unnecessarily lengthy lemmas, we only give an informal strategy, and postpone the adequate contraction sequence.

4.2 Fence gadget

The vertex set of a fence gadget is $A \cup B$ with $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ and $B = \{b_1, b_2, b_3, b_4, b_5, b_6\}$. Its black edge set consists of 13 edges: the cycles $a_1a_2a_3a_4a_5a_6a_1$ and $b_1b_2b_3b_4b_5b_6b_1$, plus the edge b_1a_6. Its red edge set consists of 11 edges: a_ib_i for each $i \in [6]$, and a_ib_{i+1} for each $i \in [5]$. A fence gadget is *attached* to a vertex subset S by making A fully adjacent to S. See Figure 4 for an illustration.

We will later nest fence gadgets. Thus we have to tolerate that F has other neighbors than S in G. Actually we even allow $V(F)$ to have neighbors outside of S and the fence gadgets surrounding F. We however always observe the following rule.

▶ **Definition 12** (Attachment rule)**.** *A fence gadget F with vertex bipartition (A, B), and attached to S, satisfies the* attachment rule *in a trigraph H if F is a connected component of $\mathcal{R}(H)$, and there is a set $X \subseteq V(H) \setminus (A \cup B \cup S)$ such that:*

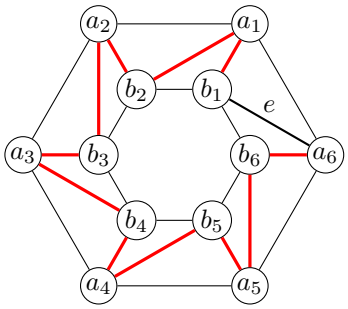

Figure 4 The fence gadget F, with $A = \{a_i \mid 1 \leqslant i \leqslant 6\}$ and $B = \{B_i \mid 1 \leqslant i \leqslant 6\}$.

- $\forall x \in A$, $N(x) \setminus V(F) = X \cup S$,
- $\forall x \in B$, $N(x) \setminus V(F) = X$, *and*
- $\forall x \in X$, $S \subset N(x)$.

Initially in G, we make sure that all the fence gadgets satisfy the attachment rule. This will remain so until we decide to contract them.

For each fence gadget F satisfying the attachment rule, there is only one adequate set X, it is $X = N(F) \setminus (V(F) \cup S)$. We denote by Y the set $V(G) \setminus (V(F) \cup S \cup X)$. We make the following observations on the three possible neighborhoods that vertices outside of F have within $V(F)$.

▶ **Observation 13.** *The fence gadget definition and the attachment rule implies:*

- $\forall x \in S$, *it holds* $N(x) \cap V(F) = A$,
- $\forall x \in X$, *it holds* $N(x) \cap V(F) = V(F) = A \cup B$, *and*
- $\forall x \in Y$, *it holds* $N(x) \cap V(F) = \emptyset$.

Henceforth we will represent every fence gadget as a brown rectangle surrounding the set S it is attached to. The vertices of X are linked to the brown rectangle, as they are fully adjacent to $S \cup V(F)$. See Figure 5 for an illustration of the attachment rule, and a compact representation of fence gadgets.

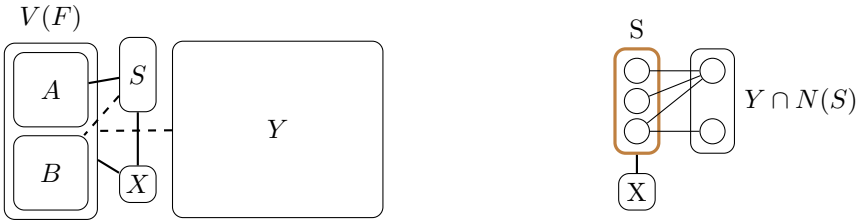

Figure 5 Left: The forced adjacencies (solid lines, all edges between the two sets) and non-adjacencies (dashed lines, no edge between the two sets), as specified by the attachment rule. Right: Symbolic representation of the fence gadget attached to S by a brown rectangle.

Constraints of the fence gadget on a 4-sequence. The following lemmas are preparatory steps for the milestone that no part in a 4-sequence of G can overlap S (that is, intersects S without containing it).

▶ **Lemma 14** ($\star$). *The first contraction involving two vertices of F results in a vertex of red degree at least 5, except if it is a contraction of some $a_i \in A$ with some $b_j \in B$.*

▶ **Lemma 15** (⋆). *If the first contraction involving two vertices of F is of some $a_i \in A$ with some $b_j \in B$, the red degree within F of the created vertex is at least 3.*

The last preparatory step is this easy lemma.

▶ **Lemma 16.** *Before a contraction involves two vertices of $V(F)$, the following holds in a partial 4-sequence:*
- *no part intersects both X and S,*
- *no part intersects both Y and S, and*
- *no part intersects both X and Y.*

Proof. By Observation 13, such a part would have red degree $|B| = 6$, $|A| = 6$, and $|A \cup B| = 12$, respectively. ◀

As a consequence we obtain the following.

▶ **Lemma 17.** *In a partial 4-sequence of G, the first contraction involving a vertex in $V(F)$ and a vertex in $V(F) \cup S$ has to be done after S is contracted into a single vertex.*

Proof. We consider the first time a vertex $u \in V(F)$ is involved in a contraction with a vertex of $V(F) \cup S$. Either (case 1) the part of u, P_u, is contracted with a part P_v containing $v \in V(F)$, or (case 2) P_u is contracted with a part P intersecting S but not $V(F)$.

In case 1, by Lemma 14, u and v hit both A and B. Thus, by Lemma 15, the red degree within F of the resulting vertex z is at least 3. Moreover z is linked by a red edge to every part within S, since S is fully adjacent to A, and fully non-adjacent to B. Thus S should at this point consist of a single part.

We now argue that case 2 is impossible in a partial 4-sequence. By Lemma 16, part P cannot intersect $X \cup Y$ (nor $V(F)$, by construction). Thus $P \subseteq S$. If $u \in A$, then the contraction of P_u and P has incident red edges toward at least 5 vertices: three vertices of A non-adjacent to u and two private neighbors of u (in the total graph) within B. If instead $u \in B$, the red degree of the contracted part is at least 6, as witnessed by two neighbors of u in B, and four non-neighbors of u in A. ◀

We can now establish the main lemma on how a fence gadget constrains a 4-sequence. Lemmas 16 and 17 have the following announced consequence: While S is not contracted into a single vertex, no part within S can be contracted with a part outside of S, and similarly vertices of X cannot be contracted with vertices of Y.

▶ **Lemma 18** (⋆). *In a partial 4-sequence, while S is not contracted to a single vertex,*
- *(i) no part intersects both S and $V(G) \setminus S$, nor*
- *(ii) both X and Y.*

Contracting the fence gadget. The previous lemmas establish some constraints that the fence gadget imposes on a supposed (partial) 4-sequence. We now see how a partial 4-sequence actually contracts a fence gadget.

Every time we are about to contract a fence gadget F attached to S, we will ensure that the following properties hold:
- no prior contraction has involved a vertex of $V(F)$,
- no red edge has one endpoint in $V(F)$ and one endpoint outside $V(F)$, and
- S is contracted into a single vertex with red degree at most 3.

In particular, the fence gadget F still satisfies the attachment rule.

▶ **Lemma 19** (⋆). *Let H be a trigraph containing a fence gadget F attached to a single vertex s of red degree at most 3. We assume that F satisfies the attachment rule in H.*

Then there is a partial 4-sequence from H to H', where H' is the trigraph obtained from H by contracting $V(F)$ into a single vertex.

4.3 Propagation, wire, and long chain

A *vertical set* V consists of two vertices x, y combined with a fence gadget F attached to $\{x, y\}$. Thus $V = \{x, y\} \cup V(F)$. We call *vertical pair* the vertices x and y. We will usually add a superscript to identify the different copies of vertical sets: Every vertex whose label is of the form u^j belongs to the vertical set V^j.

The *propagation gadget* from V^j to V^k puts all the edges between V^j and x^k (and no other edge). We call these edges an *arc from V^j to V^k*. We also say that the vertical set V^k is *guarded* by V^j. The pair V^j, V^k is said *adjacent*. Here, singleton $\{x^k\}$ plays the role of X (Definition 12) for the attachment rule of vertical set V_j.

The *propagation digraph of G*, denoted by $\mathcal{D}(G)$, has one vertex per vertical set and an arc between two vertical sets linked by an arc (in the previous sense). A (maximal) *wire* W is an induced subgraph of G corresponding in $\mathcal{D}(G)$ to a (maximal) out-tree on at least two vertices. See Figure 6 for an illustration of a wire made by simply concatenating propagation gadgets. Eventually $\mathcal{D}(G)$ will have out-degree at most 2, in-degree at most 2, and total degree at most 3.

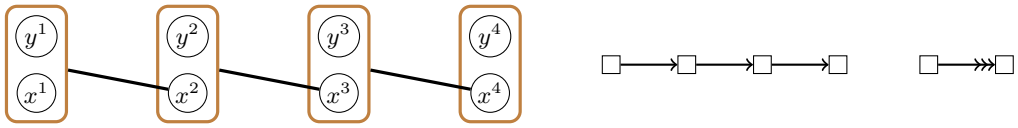

Figure 6 Left: A non-branching wire made of 4 vertical sets and 3 propagation gadgets. Every vertical set is guarded by the vertical set just to its left. Center: A more compact representation, which corresponds to the propagation digraph. Right: Symbolic representation of the long chain, that is, of the represented wire if $L = 4$.

The *children* of a vertical set V are the vertical sets that V guards. The *root of wire W* is the unique vertical set of in-degree 0 in $\mathcal{D}(W)$. The *leaves of wire W* are the vertical sets of out-degree 0 in $\mathcal{D}(W)$. A wire is said *primed* when the vertical pair of its root has been contracted.

A wire W is *non-branching* if every vertex of $\mathcal{D}(W)$ has out-degree at most 1; hence, $\mathcal{D}(W)$ is a directed path. A *long chain* is a wire W such that $\mathcal{D}(W)$ is a directed path on L vertices, where integer L will be specified later (and can be thought as logarithmic in the total number of fences which do not belong to vertical sets). Otherwise, if $\mathcal{D}(W)$ has at least one vertex with out-degree at least 2, wire W is said *branching*. A vertical set with two children is also said *branching*.

Constraints of the propagation gadget on a 4-sequence. We provide the proof that a contraction in a vertical set V is only possible when the vertical pair of all the vertical sets V' with a directed path to V in $\mathcal{D}(G)$ has been contracted.

▶ **Lemma 20.** *Let V^j and V^k be two vertical sets with an arc from V^j to V^k. In a partial 4-sequence from G, any contraction involving two vertices of V^k has to be preceded by the contraction of x^j and y^j.*

Proof. We recall the notations of Section 4.2. Let F be the fence gadget attached to $S = \{x^j, y^j\}$, X the neighborhood of $V(F)$ outside of $S \cup V(F)$, and Y the vertices that are not in $S \cup V(F) \cup X$. (We always assume that the attachment rule is satisfied.) We have $x^{j'} \in X$ and $y^k \in Y$, therefore by the second item of Lemma 18, their contraction has to be preceded by the contraction of x^j and y^j. Now applying the first item of Lemma 18 to the fence gadget F' attached to $S' = \{x^k, y^k\}$, and Lemma 17, any contraction involving a pair of V^k distinct from S' has to be preceded by the contraction of x^k and y^k. ◀

Henceforth, when we say that a contraction is *preceded* by another contraction, it includes the case that the two contractions are in fact the same. By a straightforward induction, we obtain the following from Lemma 20 (and Lemma 17).

▶ **Lemma 21.** *In a partial 4-sequence from G, any contraction involving a pair of vertices in a vertical set V has to be preceded by the contraction of the vertical pair of every vertical set V' such that there is a directed path from V' to V in $\mathcal{D}(G)$.*

Contracting wires. As the roots and leaves of wires will be connected to other gadgets, we postpone the description of how to contract wires until after building the overall construction G. Intuitively though, contracting a wire (in the vacuum) consists of contracting the vertical pair of its root, then its fence gadget by applying Lemma 19, and finally recursively contracting the subtrees rooted at its children. Since $\mathcal{D}(G)$ has total degree at most 3, every vertex has red degree at most 4 (for the at most 3 adjacent vertical sets, plus the pendant vertex of the fence gadget).

4.4 Binary AND gate

The *binary* AND *gate* (AND gadget, for short) simply consists of three vertical sets V^1, V^2, V^3 with an arc from V^1 to V^3, and an arc from V^2 to V^3. As usual, the vertical pairs of V^1, V^2, and V^3, are $\{x^1, y^1\}$, $\{x^2, y^2\}$, and $\{x^3, y^3\}$, respectively. We call the vertical sets V^1, V^2 the *inputs* of the AND gadget, and the vertical set V^3 the *output* of the AND gadget.

Constraint of the AND gadget on a 4-sequence. By Lemma 20, we readily derive:

▶ **Lemma 22.** *Assume G contains an AND gadget with inputs V^1, V^2, and output V^3. In a partial 4-sequence from G, any contraction involving two vertices of V^3 has to be preceded by the contraction of x^1 and y^1, and the contraction of x^2 and y^2.*

Contraction of an AND gadget. Once V^1 and V^2 are contracted into single vertices, one can contract the vertical pair x^3, y^3. This results in a vertex of red degree 2. Thus one can contract the fence gadget of V^3 by applying Lemma 19.

4.5 Binary OR gate

The *binary* OR *gate* (OR *gadget*) is connected to three vertical sets: two *inputs* V^1, V^2, and one *output* V^3. We start by building two vertical sets V, V' whose vertical pairs are $\{a, b\}$ and $\{c, d\}$, respectively. The edges ac and bd are added, as well as a vertex e adjacent to a and to c. Finally a fence is attached to $\{e\} \cup V \cup V'$.

The OR gadget is connected to its inputs and output, in the following way. Vertex a is made adjacent to x^1 and to y^1 (but not to their fence gadget). Similarly vertex b is linked to x^2 and y^2. Finally x^3 is adjacent to all the vertices of the OR gadget, that is, $\{e\} \cup V \cup V'$ plus the vertices of the outermost fence. See Figure 7 for a representation of the OR gadget.

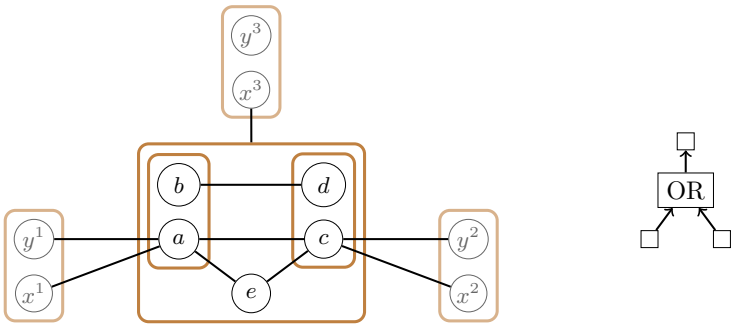

Figure 7 An OR gadget attached to inputs V^1, V^2 and output V^3, and its symbolic representation. These vertical sets are technically not part of the OR gate, so we represent them slightly dimmer.

Constraint of the OR gadget on a 4-sequence. By design, one can only start contracting the OR gadget after the vertical pair of at least one of its two inputs has been contracted. This implies that no contraction can involve V^3 before at least one the vertical pairs $\{x^1, y^1\}$ and $\{x^2, y^2\}$ is contracted.

▶ **Lemma 23.** *Assume G contains an OR gadget attached to inputs V^1 and V^2. In a partial 4-sequence from G, the contractions of a, b and of c, d have to be preceded by the contraction of x^1, y^1 or the contraction of x^2, y^2.*

Proof. Assume none of the pairs $\{a, b\}, \{c, d\}, \{x^1, y^1\}, \{x^2, y^2\}$ have been contracted. Because of the fences, by Lemma 18, all the vertices $x^1, y^1, a, b, c, d, x^2, y^2$ and e are in distinct parts. Therefore contracting a and b would create a vertex of red degree at least 5, considering the (singleton) parts of x^1, y^1, c, d, e. Symmetrically contracting c and d yields at least five red neighbors, considering the (singleton) parts of x^2, y^2, a, b, e. ◀

From Lemma 23 we get the following.

▶ **Lemma 24.** *Assume G contains an OR gadget attached to inputs V^1 and V^2. In a partial 4-sequence from G, no contraction involving a vertex of V^3 can happen before either the pair x^1, y^1 or the pair x^2, y^2 is contracted.*

Proof. Suppose neither x^1, y^1 nor x^2, y^2 is contracted. By Lemma 23, the pairs $\{a, b\}$ and $\{c, d\}$ cannot be contracted. By the first item of Lemma 18, no contraction can involve a vertex of $\{a, b, c, d, e\}$. As x^3 is adjacent to all these vertices but not y^3, one cannot contract the vertical pair $\{x^3, y^3\}$. Hence by Lemma 17 no contraction can involve a vertex of V^3. ◀

Contraction of the OR gadget. We now show how to contract the OR gate when the vertical pair of one of its inputs has been contracted.

▶ **Lemma 25.** *Assume that x^1 and y^1 have been contracted into z^1, and that z^1, x^2, and y^2 all have red degree at most 3. Then there is a partial 4-sequence that contracts the whole OR gadget to a single vertex with only three red neighbors: z^1, x^2, and y^2. (The same holds symmetrically if x^2 and y^2 have been contracted into a single vertex.)*

Proof. First contract a and b into vertex α of red degree 4. At this point the fence of $\{a, b\}$ cannot be contracted yet, as this would make the red degree of α go above 4. Hence we next contract c and d into γ, decreasing the red degree of α to 3. Note that γ has only 4 red neighbors: α, e, x^2, y^2.

By Lemma 19, we can now contract the fence gadget of $\{a, b\}$ to a single vertex. Next we contract this latter vertex with α, and the resulting vertex with t; we call α' the obtained vertex. Now γ has only red degree 3, so we can contract the fence gadget of $\{c, d\}$ to a single vertex that we further contract with γ; we call γ' the obtained vertex. We contract α' and γ' in a vertex ε of red degree 3; its three red neighbors are z^1, x^2, and y^2. Again by Lemma 19, the outermost fence of the OR gadget can be contracted into a single vertex, that we finally contract with ε. This results in a vertex with three red neighbors: z^1, x^2, and y^2.

Throughout this process the red degree of z^1, x^2, and y^2 never goes above 4. Indeed the red degree of these vertices is initially at most 3, while they have exactly one black neighbor in the entire OR gadget (so at most one part to be in conflict with). ◀

4.6 Variable gadget

The variable gadget is represented in Figure 8 (see long version for a textual description).

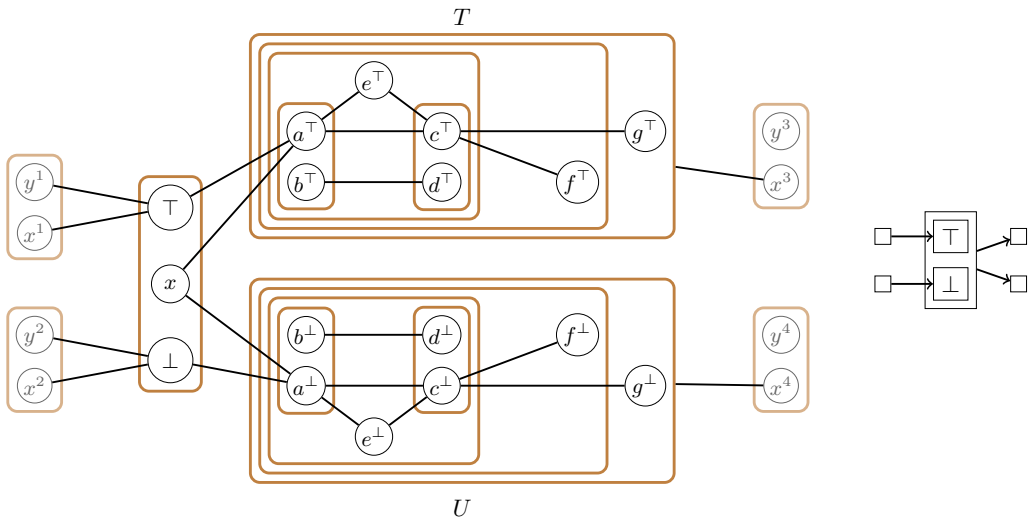

Figure 8 A variable gadget half-guarded by V^1, V^2, and with outputs V^3, V^4, and its compact representation (right).

Constraints of the variable gadget on a 4-sequence. Because of the fence gadget attached to $\{\top, x, \bot\}$, one has at some point to contract $\top$ and $\bot$ (be it with or without x). A first observation is that this has to wait that the vertical pair of V^1 or V^2 is contracted.

▶ **Lemma 26.** *Assume G has a variable gadget half-guarded by vertical sets V^1 and V^2. In a partial 4-sequence from G, the contraction of $\top$ and $\bot$ has to be preceded by the contraction of the pair x^1, y^1 or of the pair x^2, y^2.*

Proof. The pairs $\{x^1, y^1\}$, $\{x^2, y^2\}$, and $\{\top, \bot\}$ are contained in three disjoint sets S^1, S^2, S, respectively, to which a fence is attached. Thus before any of these pairs are contracted, by Lemma 18, a vertex outside $S^1 \cup S^2 \cup S$, like $a^\bot$, is in a different part than the six vertices $x^1, y^1, x^2, y^2, \top, \bot$. Therefore contracting $\top$ and $\bot$ would create a vertex with five red neighbors, considering the parts of $x^1, y^1, x^2, y^2, a^\bot$. ◀

We next show that no contraction is possible in U (resp. in T), while x and $\bot$ (resp. x and $\top$) are not contracted.

▶ **Lemma 27.** *In a partial 4-sequence, the contractions of $a^\perp$ and $b^\perp$ (resp. $a^\top$ and $b^\top$) and of $c^\perp$ and $d^\perp$ (resp. $c^\top$ and $d^\top$) have to be preceded by the contraction of x and $\perp$ (resp. x and $\top$). Therefore no contraction is possible in U (resp. T) before x and $\perp$ (resp. x and $\top$) are contracted.*

Proof. Since the two statements are symmetric, we only show it with $\perp$. Assume none of the pairs $\{x, \perp\}$, $\{a^\perp, b^\perp\}$, $\{c^\perp, d^\perp\}$ are contracted. Because of the fence gadgets, by the first item of Lemma 18, the vertices $x, \perp, a^\perp, b^\perp, c^\perp, d^\perp, e^\perp, f^\perp$ are pairwise in distinct parts. Therefore contracting $a^\perp$ and $b^\perp$ or $c^\perp$ and $d^\perp$ would create a vertex of red degree at least 5. The structure of the fence gadgets in U thus prevents any contraction. ◀

We deduce that priming the wire of $\neg x$ (resp. $+x$) can only be done after x and $\perp$ (resp. x and $\top$) are contracted.

▶ **Lemma 28.** *Assume that G has a variable gadget with outputs the vertical sets V^3 (root of the wire of $+x$) and V_4 (root of the wire of $\neg x$). In a partial 4-sequence from G, the contraction of x^4 and y^4 (resp. x^3 and y^3) has to be preceded by the contraction of x and $\perp$ (resp. x and $\top$).*

Proof. By the second item of Lemma 18 applied to the fence attached to U, the pair x^4, y^4 cannot be contracted until U is not contracted to a single vertex. Thus by Lemma 27, the pair x^4, y^4 can only be contracted after the pair $x, \perp$ is contracted. The other statement is obtained symmetrically. ◀

Contraction of the variable gadget. We show two options to contract a "half" of the variable gadget, either T and its fence, or U and its fence, into a single vertex.

▶ **Lemma 29.** *There is a partial 4-sequence that contracts x and $\top$ together, and $T \cup F^\top$ into a single vertex. Symmetrically there is a partial 4-sequence that contracts x and $\perp$ together, and $U \cup F^\perp$ into a single vertex.*

Proof. We first contract x and $\top$ into a vertex $+x$. Observe that $+x$ has exactly three red neighbors: x^1, y^1, and $a^\perp$. Thus $\{a^\top, b^\top, c^\top, d^\top, e^\top\}$ and their three fences can be contracted exactly like an OR gadget. So by Lemma 25, there is a partial 4-sequence that contracts all these vertices to a single vertex u, with three red neighbors ($+x$, $f^\top$, and $g^\top$). We can now contract u and $f^\top$ into u', followed by contracting their fence gadget $F'^\top$ into a single vertex, by Lemma 19. That pendant vertex can be contracted to u', and the result to $g^\top$, forming vertex v. Finally, again by Lemma 19, the fence $F^\top$ attached to T can be contracted into a single vertex, which can be contracted with v.

The other sequence is the symmetric. ◀

The second "half" of the variable gadget can be contracted once the vertical pairs of the *half-guards* V^1, V^2 have been contracted.

▶ **Lemma 30.** *Assume $T \cup F^\top$ (resp. $U \cup F^\perp$) has been contracted into a single vertex u, and that the pairs $\{\top, x\}$ (resp. $\{\perp, x\}$), $\{x^1, y^1\}$, and $\{x^2, y^2\}$ have been contracted into $+x$ (resp. $\neg x$), z^1, and z^2, respectively. We further assume that the red degree of z^2 (resp. z^1) is at most 3. Then there is partial 4-sequence that contracts $\top, x, \perp$ and their fence into a single vertex, and $U \cup F^\perp$ (resp. $T \cup F^\top$) into a single vertex.*

Proof. We contract $\perp$ with $+x$ into v of red degree 4. This increases the red degree of z^2 by one, which remains at most 4. We then contract $U \cup F^\perp$ into a single vertex w, like in Lemma 29. We contract u and w into y, a vertex of red degree at most 3. Now v has

degree 3. So we can contract its fence gadget by Lemma 19. We further contract v with its pendant neighbor, and finally with y. What results is a unique vertex with four red neighbors.

The other partial sequence is symmetric. ◀

4.7 Clause gadget

A *3-clause gadget* (or simply *clause gadget*) has for *inputs* three vertical sets V^1, V^2, V^3, and for *output* one vertical set V^4. It consists of combining two OR gadgets, and using long chains to make the OR gadgets *distant enough*. We add an OR gadget with input V^1 and V^2, and output V. We then add a long chain from V to V', and an OR gadget with input V' and V^3, and output V^4. The clause gadget is depicted in Figure 9. We call *first OR gadget* of the clause gadget, the one with output V, and *second OR gadget*, the one with output V^4.

Figure 9 Left: A 3-clause gadget. Right: A shorthand for the gadget.

Constraint of the clause gadget on a 4-sequence. As a consequence of Lemmas 21 and 24, we get that once a contraction involves the output, at least one of the vertical pairs of the inputs has been contracted.

▶ **Lemma 31** (⋆). *Assume G contains a clause gadget with inputs V^1, V^2, V^3 and output V^4. In a partial 4-sequence from G, any contraction involving a vertex of V^4 is preceded by the contraction of the vertical pair of one of V^1, V^2, or V^3.*

Contraction of the clause gadget. The OR gates of the clause gadgets will be contracted as specified in Lemma 25, while we wait the overall construction to describe the contraction of the wires.

4.8 Overall construction and correctness

Let $I = (C_1, \ldots, C_m)$ be an instance of 3-SAT, that is, a collection of m 3-clauses over the variables $x_1, \ldots, x_n$. We further assume that each variable appears at most twice positively, and at most twice negatively in I. The 3-SAT problem remains NP-complete with that restriction, and without $2^{o(n)}$-time algorithm unless the ETH fails; see Theorem 6. We build a trigraph H that has twin-width at most 4 if and only if I is satisfiable. As trigraph H will satisfy the condition of Lemma 11, it can be replaced by a graph G on $O(|V(H)|)$ vertices. We set L the length of the long chain to $2\lceil \log(5n+3m) \rceil = O(\log n)$, so that G has $O(n \log n)$ vertices. We now piece the gadgets described in the previous sections together.

Variable to clause gadgets. For every variable x_i, we add a variable gadget half-guarded by V_i^1, V_i^2, and with outputs V_i^3, V_i^4. We add a long chain starting at vertical set V_i^3 (resp. V_i^4), and ending at a branching vertical set from which starts two long chains stopping at vertical sets $V_i^{\top,1}$ and $V_i^{\top,2}$ (resp. $V_i^{\bot,1}$ and $V_i^{\bot,2}$). Vertical set $V_i^{\top,1}$ (resp. $V_i^{\bot,1}$) serves as the

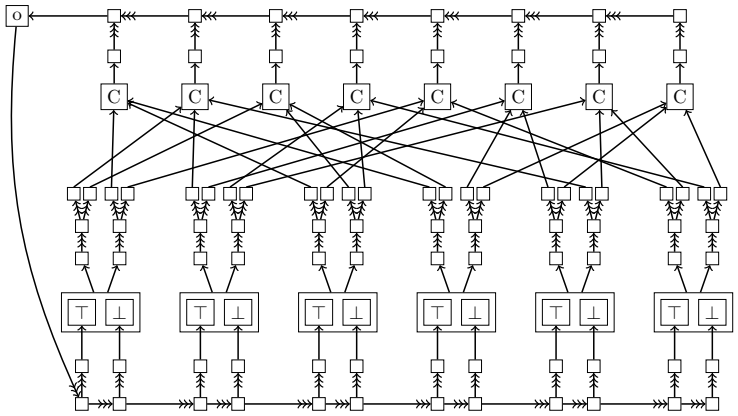

Figure 10 An example of the overall construction G on a 3-SAT instance with 6 variables and 8 clauses. The first two clauses are $\neg x_1 \vee x_3 \vee x_4$ and $x_1 \vee x_2 \vee \neg x_5$.

input of the first clause gadget in which x_i appears positively (resp. negatively), while $V_i^{\top,2}$ (resp. $V_i^{\bot,2}$) becomes the input of the second clause gadget in which x_i appears positively (resp. negatively). If a literal has only one occurrence, then we omit the corresponding vertical set, and the long chain leading to it. We nevertheless assume that each literal has at least one occurrence, otherwise the corresponding variable could be safely assigned.

Clause gadgets to global output. For every $j \in [m]$, we add a long chain from the output of the clause gadget of C_j, to a vertical set, denoted by V_j^c. For every $j \in [2, m]$, we then add a long chain starting at V_j^c and ending at V_{j-1}^c. We add an arc from V_1^c to a new vertical set V^o, which we call the *global output*.

Global output back to half-guarding the variable gadgets. For every $i \in [n]$, we add two vertical sets $V_i^{1,r}, V_i^{2,r}$, and puts a long chain starting at $V_i^{1,r}$ and ending at $V_i^{2,r}$. We add a long chain from V^o to $V_1^{1,r}$. We also add a long chain from $V_i^{2,r}$ to $V_{i+1}^{2,r}$, for every $i \in [n-1]$. Finally we add a long chain from $V_i^{a,r}$ to V_i^a for every $a \in \{1, 2\}$ and every $i \in [n]$. Recall that V_i^1 and V_i^2 are half-guarding the variable gadget of x_i.

This finishes the construction of the graph $G = G(I)$. See Figure 10 for an illustration.

If G has twin-width at most 4, then I is satisfiable. Let us consider the trigraph H obtained after the vertical pair of the global output V^o is contracted. This has to happen in a 4-sequence by the first item of Lemma 18 applied to the fence of V^o. By Lemma 21, no contraction involving vertices of the vertical sets V_i^1, V_i^2 can have happened (for any $i \in [n]$). This is because there is a directed path in the propagation digraph $\mathcal{D}(G)$ from V^o to V_i^1 and V_i^2.

Thus by Lemmas 26 and 28, for every variable x_i ($i \in [n]$), at most one of the wire of $+x_i$ and the wire of $\neg x_i$ has been primed. We can define the corresponding truth assignment $\mathcal{A}$: x_i is set to true if the wire of $\neg x_i$ is *not* primed, and to false if instead the wire of $+x_i$ is *not* primed. Besides, by Lemma 21 applied to the contraction in V^o, every vertical pair of a clause-gadget output has been contracted. Then Lemma 31 implies that the vertical pair of at least one input of each clause gadget has been contracted. But such a vertical pair can be contracted only if it corresponds to a literal set to true by $\mathcal{A}$. For otherwise, the root of the wire of that literal cannot be contracted. This implies that $\mathcal{A}$ is a satisfying assignment.

If I is satisfiable, then G has twin-width at most 4. In what follows, when we write that we *can* contract a vertex, a set, or make a sequence of contractions, we mean that there is a partial 4-sequence that does the job. Let $\mathcal{A}$ be a satisfying assignment of I. We start by contracting "half" of the variable gadget of each x_i. We add a subscript matching the variable index to the vertex and set labels of Figure 8. For every $i \in [n]$, we contract vertices x_i and $\top_i$ together, and $T_i \cup F_i^\top$ to a single vertex v_i, if $\mathcal{A}$ sets variable x_i to true, and x_i and $\bot_i$ together, and $U_i \cup F_i^\bot$ to a single vertex v_i, if instead $\mathcal{A}$ sets x_i to false. By Lemma 29, this can be done by a partial 4-sequence.

Next we contract the wire of $+x_i$ if $\mathcal{A}(x_i)$ is true, or the wire of $\neg x_i$ if $\mathcal{A}(x_i)$ is false. This is done as described in the end of Section 4.3. We contract the vertical pair of the root of the wire into z_i (the red degree of v_i goes from 1 to 2). We then contract its fence by Lemma 19. We can now contract the resulting vertex with z_i. Inductively, we contract the children of the current vertical set to single vertices, in a similar fashion. As the propagation digraph has degree at most 3, this never creates a vertex of red degree more than 4.

At the leaves of the wire (the vertical sets $V_i^{\top,1}, V_i^{\top,2}$ or $V_i^{\bot,1}, V_i^{\bot,2}$), we make an exception, and only contract the vertical pair. We then contract, by Lemma 25, all the (non-already reduced) OR gadgets (involved in clause gadgets) at least one input of which has its vertical pair contracted. After that, applying Lemma 19, we finish the contraction of the OR inputs whose vertical pairs was contracted. Next we contract each output of a contracted OR gadget into a single vertex.

In those clauses where the third literal is not satisfied by $\mathcal{A}$, the output of the clause gadget is not contracted at that point. However, as $\mathcal{A}$ is a satisfying assignment, the output of the first OR gate of the gadget is contracted. We then contract each vertical set of the long chain leading to the input of the second OR gate. We only contract the vertical pair of that input, and contract the incident OR gadget, by Lemma 25. We finally proceed by contracting the input vertical set into a single vertex, and the output vertical set into a single vertex.

At this point, each output of the clause gadgets is contracted into a single vertex. The (*not* strongly) connected component C of the propagation digraph $\mathcal{D}(G)$ containing the global output V_o is acyclic and has total degree at most 3. All the vertical sets of C with in-degree 0 (ant out-degree 1) in $\mathcal{D}(G)$ are the clause outputs, which have been contracted to single vertices. Thus each vertical set of C can be contracted to single vertices, by repeated use of Lemma 19, followed by contracting the pendant vertex (resulting from the fence contraction) with its unique (red) neighbor. Note that this process terminates by contracting the half-guards V_i^1, V_i^2 (for every $i \in [n]$). The conditions of Lemma 30 are now satisfied, so we can finish contracting each variable gadget into two vertices that we further contract together. This results in a vertex of red degree 4.

We can then contract the wire of the literal that was set to false by $\mathcal{A}$. Again, we only contract the vertical pair of the inputs of OR gates that are not contracted yet. Then we contract those OR gadgets, and finish by fully contracting the vertical set of those inputs. We next contract each output of the newly contracted OR gates. We eventually contract into a single vertex each vertical set of the long chain in clause gadgets where this was not already done.

At this point, the current trigraph H has only red edges. Thus it can be interpreted as a graph, and we write *degree* instead or *red degree*. H has $4n + 3m$ vertices of degree 3, n vertices of degree 4, and the rest of its vertices have degree 2. Because we added long chains to separate what now corresponds to vertices of degree at least 3, H is an $(\geqslant L)$-subdivision of a graph on $5n + 3m$ vertices. Since $L = 2\lceil \log(5n + 3m) \rceil$, by Lemma 9, H finally admits a 4-sequence.

References

1 Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

2 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.

3 Édouard Bonnet, Colin Geniet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width II: small classes. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1977–1996, 2021. doi:10.1137/1.9781611976465.118.

4 Édouard Bonnet, Ugo Giocanti, Patrice Ossona de Mendez, Pierre Simon, Stéphan Thomassé, and Szymon Toruńczyk. Twin-width IV: ordered graphs and matrices. *CoRR*, abs/2102.03117, 2021. arXiv:2102.03117.

5 Édouard Bonnet, Eun Jung Kim, Amadeus Reinald, Stéphan Thomassé, and Rémi Watrigant. Twin-width and polynomial kernels. 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 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.10.

6 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 601–612. IEEE, 2020. doi:10.1109/FOCS46700.2020.00062.

7 Derek G. Corneil, Michel Habib, Jean-Marc Lanlignel, Bruce A. Reed, and Udi Rotics. Polynomial-time recognition of clique-width $\leqslant 3$ graphs. *Discret. Appl. Math.*, 160(6):834–865, 2012. doi:10.1016/j.dam.2011.03.020.

8 Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socala. Tight lower bounds on graph embedding problems. *J. ACM*, 64(3):18:1–18:22, 2017. doi:10.1145/3051094.

9 Anuj Dawar and Stephan Kreutzer. Parameterized complexity of first-order logic. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:131, 2009. URL: http://eccc.hpi-web.de/report/2009/131.

10 Markus Sortland Dregi and Daniel Lokshtanov. Parameterized complexity of bandwidth on trees. 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 405–416. Springer, 2014. doi:10.1007/978-3-662-43948-7_34.

11 Zdenek Dvorák, Daniel Král, and Robin Thomas. Testing first-order properties for subclasses of sparse graphs. *J. ACM*, 60(5):36:1–36:24, 2013. doi:10.1145/2499483.

12 Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008. doi:10.1137/05064299X.

13 Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-width is NP-complete. *SIAM J. Discret. Math.*, 23(2):909–939, 2009. doi:10.1137/070687256.

14 Jörg Flum and Martin Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM J. Comput.*, 31(1):113–145, 2001. doi:10.1137/S0097539799360768.

15 Fedor V. Fomin, Daniel Lokshtanov, Ivan Mihajlin, Saket Saurabh, and Meirav Zehavi. Computation of hadwiger number and related contraction problems: Tight lower bounds. *ACM Trans. Comput. Theory*, 13(2):10:1–10:25, 2021. doi:10.1145/3448639.

16 Jakub Gajarský, Petr Hlinený, Daniel Lokshtanov, Jan Obdrzálek, Sebastian Ordyniak, M. S. Ramanujan, and Saket Saurabh. FO model checking on posets of bounded width. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 963–974, 2015. doi:10.1109/FOCS.2015.63.

17 Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, and Sebastian Ordyniak. Faster existential FO model checking on posets. *Logical Methods in Computer Science*, 11(4), 2015. doi:10.2168/LMCS-11(4:8)2015.

18 Robert Ganian, Petr Hlinený, Daniel Král, Jan Obdrzálek, Jarett Schwartz, and Jakub Teska. FO model checking of interval graphs. *Logical Methods in Computer Science*, 11(4), 2015. doi:10.2168/LMCS-11(4:11)2015.

19 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

20 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *J. ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

21 Sylvain Guillemot and Dániel Marx. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 82–101, 2014. doi:10.1137/1.9781611973402.7.

22 Michel Habib and Christophe Paul. A simple linear time algorithm for cograph recognition. *Discret. Appl. Math.*, 145(2):183–197, 2005. doi:10.1016/j.dam.2004.01.011.

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

24 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

25 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

26 Toshinobu Kashiwabara. NP-completeness of the problem of finding a minimal-clique number interval graph containing a given graph as a subgraph. In *Proc. 1979 Int. Symp. Circuit Syst*, pages 657–660, 1979.

27 Tuukka Korhonen. Single-exponential time 2-approximation algorithm for treewidth. *CoRR*, abs/2104.07463, 2021. arXiv:2104.07463.

28 Thomas Lengauer. Black-white pebbles and graph separation. *Acta Informatica*, 16:465–475, 1981. doi:10.1007/BF00264496.

29 Tatsuo Ohtsuki, Hajimu Mori, Ernest S. Kuh, Toshinobu Kashiwabara, and Toshio Fujisawa. One-dimensional logic gate assignment and interval graphs. In *The IEEE Computer Society's Third International Computer Software and Applications Conference, COMPSAC 1979, 6-8 November, 1979, Chicago, Illinois, USA*, pages 101–106. IEEE, 1979. doi:10.1109/CMPSAC.1979.762474.

30 Sang-il Oum. Approximating rank-width and clique-width quickly. *ACM Trans. Algorithms*, 5(1):10:1–10:20, 2008. doi:10.1145/1435375.1435385.

31 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

32 Sigve Hortemo Sæther and Martin Vatshelle. Hardness of computing width parameters based on branch decompositions over the vertex set. *Theor. Comput. Sci.*, 615:120–125, 2016. doi:10.1016/j.tcs.2015.11.039.

33 James B. Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. Algebraic Discret. Methods*, 1(4):363–369, 1980. doi:10.1137/0601042.

34 André Schidler and Stefan Szeider. A SAT approach to twin-width. *CoRR (accepted at ALENEX '22)*, abs/2110.06146, 2021. arXiv:2110.06146.

35 Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discret. Appl. Math.*, 8(1):85–89, 1984. doi:10.1016/0166-218X(84)90081-7.