Deciding twin-width at most 4 is NP-complete[1]

Pierre Bergé — LIP, ENS Lyon, Université Lyon 1, France
Edouard Bonnet — LIP, ENS Lyon, Université Lyon 1, France
Hugues Déprés — LIP, ENS Lyon, Université Lyon 1, 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.

1 Introduction to twin-width

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

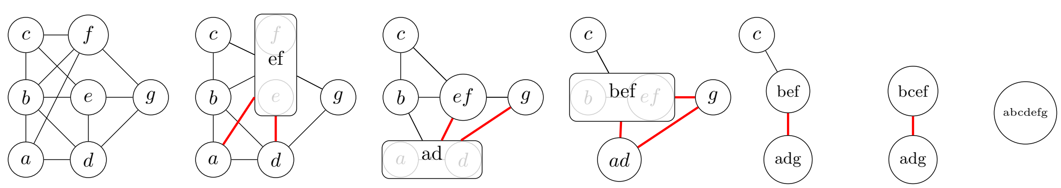


Figure 1: A 2-sequence witnessing that the initial graph has twin-width at most 2.

(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 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)

[1] The full version of this extended abstract can be found at: `https://arxiv.org/abs/2112.08953`

$f(|\varphi|, d) \cdot |V(G)|$ 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 [9, 11, 12, 10, 13] for hereditary (but not necessarily monotone) classes.

2 Results

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.

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.*

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], clique-width [8], rank-width [15] and mim-width [16]. 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]. In contrast, 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 [14], 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.*

We knew that those graphs have bounded twin-width [3], but not with the explicit bound.

3 Outline of the proof of Theorem 1

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.

Given an n-variable instance I of 3-SAT, we 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.

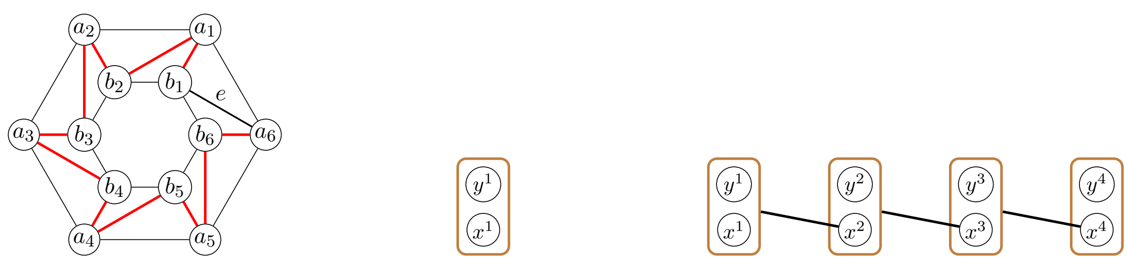
We proceed in two steps. First, we design a trigraph H, depending on I such that H has twin-width at most 4 if and only if I is satisfiable. However, we aim at showing the NP-hardness of computing twin-width on (plain) graphs, and not trigraphs. Hence, as a second step, we provide a construction allowing to encode trigraphs H into graphs G.

Given any 3-SAT instance I, the red graph of the trigraph H produced has maximum degree 2 and connected components of bounded size. Our encoding allows us, given a trigraph H with red degree at most d, to obtain a graph G such that H admits a $2d$-sequence iff G admits a $2d$-sequence. Hence, for $d = 2$:

Lemma 1. *Given any trigraph H whose red graph is a disjoint union of paths of bounded size 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.*

Now, we give some intuition for the first step of the reduction, *i.e.* transforming a 3-SAT instance I into an equivalent trigraph H for the problem of deciding whether $tww(H) = 4$. We propose several gadgets: their role is described below.

- the *fence* gadget is a trigraph F whose red graph is a 12-vertex path. Its vertex set can be partitionned into two sets $A = \{a_i \mid 1 \leqslant i \leqslant 6\}$ and $B = \{B_i \mid 1 \leqslant i \leqslant 6\}$. A set S of vertices is *attached* to the fence if it is fully adjacent to A and fully non-adjacent to B. We ensure in this reduction that every vertex not attached to F is fully adjacent to B or fully non-adjacent to A. Our intent is that, in a 4-sequence, a vertex of F can be contracted with another vertex only when S has been contracted into a single vertex.
- the *vertical set* is made up of two vertices (called a *vertical pair*) attached to a fence.
- the *propagation* gadget of two vertical sets: it forces, in a 4-sequence, one vertical pair to be contracted before the second one.



(a) The fence gadget F. (b) A vertical set. (c) A series of propagation gadgets.

Figure 2: We represent every fence gadget as a brown rectangle surrounding set S it is attached to.

- the *AND* gadget: it forces two vertical pairs to be contracted before a third one. The *OR* gadget forces at least one among two vertical pairs to be contracted before a third one.
- the *variable* gadget: it contains an "input" fence with three vertices $x, \top, \bot$ and two "output" vertical pairs $\{x_\top, y_\top\}$, $\{x_\bot, y_\bot\}$. This gadget is constructed in such a way that, inside the 3-vertex fence, the first pair to be contracted must be either $(x, \top)$ or $(x, \bot)$. Contracting $(x, \top)$ (resp. $(x, \bot)$) precedes necessarily the contraction of the vertical pair $\{x_\top, y_\top\}$ (resp. $\{x_\bot, y_\bot\}$). The meaning of this action is to assign variable x to boolean True (resp. False).

By assembling these elements together, we can translate any instance I into an equivalent boolean circuit made up of fences and vertical sets. The trigraph produced is H. Each connected component of its red graph is the red graph of a fence, *i.e.* a 12-vertex path.

References

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

[2] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[3] E. Bonnet, C. Geniet, E. Kim, S. Thomassé, and R. Watrigant. Twin-width II: small classes. In *Proc. of SODA*, pages 1977–1996, 2021.

[4] E. Bonnet, U. Giocanti, P. Ossona de Mendez, P. Simon, S. Thomassé, and S. Toruńczyk. Twin-width IV: ordered graphs and matrices. *CoRR*, abs/2102.03117, 2021.

[5] E. Bonnet, E. Kim, A. Reinald, S. Thomassé, and R. Watrigant. Twin-width and polynomial kernels. In *Proc. of IPEC*, volume 214, pages 10:1–10:16, 2021.

[6] E. Bonnet, E. Kim, S. Thomassé, and R. Watrigant. Twin-width I: tractable FO model checking. In *Proc. of FOCS*, pages 601–612. IEEE, 2020.

[7] D. G. Corneil, M. Habib, J. Lanlignel, B. A. Reed, and U. Rotics. Polynomial-time recognition of clique-width $\leq$3 graphs. *Discret. Appl. Math.*, 160(6):834–865, 2012.

[8] M. R. Fellows, F. A. Rosamond, U. Rotics, and S. Szeider. Clique-width is NP-complete. *SIAM J. Discret. Math.*, 23(2):909–939, 2009.

[9] J. Flum and M. Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM J. Comput.*, 31(1):113–145, 2001.

[10] J. Gajarský, P. Hlinený, D. Lokshtanov, J. Obdrzálek, S. Ordyniak, M. S. Ramanujan, and S. Saurabh. FO model checking on posets of bounded width. In *Proc. of FOCS*, pages 963–974, 2015.

[11] J. Gajarský, P. Hlinený, J. Obdrzálek, and S. Ordyniak. Faster existential FO model checking on posets. *Logical Methods in Computer Science*, 11(4), 2015.

[12] R. Ganian, P. Hlinený, D. Král, J. Obdrzálek, J. Schwartz, and J. Teska. FO model checking of interval graphs. *Logical Methods in Computer Science*, 11(4), 2015.

[13] S. Guillemot and D. Marx. Finding small patterns in permutations in linear time. In *Proc. of SODA*, pages 82–101, 2014.

[14] M. Habib and C. Paul. A simple linear time algorithm for cograph recognition. *Discret. Appl. Math.*, 145(2):183–197, 2005.

[15] P. Hlinený and S. Oum. Finding branch-decompositions and rank-decompositions. *SIAM J. Comput.*, 38(3):1012–1032, 2008.

[16] S. H. Sæther and M. Vatshelle. Hardness of computing width parameters based on branch decompositions over the vertex set. *Theor. Comput. Sci.*, 615:120–125, 2016.