Treewidth Inapproximability and Tight ETH Lower Bound

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

Abstract

Despite the (algorithmic) importance of treewidth, both its complexity and approximability present large knowledge gaps. While the best polynomial-time approximation algorithm has ratio $O(\sqrt{\log \mathrm{OPT}})$, no approximation factor could be ruled under P $\neq$ NP alone. There are $2^{O(n)}$-time algorithms to compute the treewidth of n-vertex graphs, but the Exponential-Time Hypothesis (ETH) was only known to imply that $2^{\Omega(\sqrt{n})}$ time is required. The reason is that all the known hardness constructions use Cutwidth or Pathwidth on bounded-degree graphs as an intermediate step in a (long) chain of reductions, for which no inapproximability nor sharp ETH lower bound is known.

We present a simple, self-contained reduction from 3-SAT to Treewidth. This starts filling the former gap, and completely fills the latter gap. Namely, we show that 1.00005-approximating Treewidth is NP-hard, and solving Treewidth exactly requires $2^{\Omega(n)}$ time, unless the ETH fails. We further derive, under the latter assumption, that there is some $\delta > 1$ such that δ-approximating Treewidth requires time $2^{n^{1-o(1)}}$.

Acknowledgements We thank Hans Bodlaender, Tuukka Korhonen, Daniel Lokshtanov, Nicolas Trotignon, and Rémi Watrigant for some valuable feedback.

1 Introduction

Treewidth [3, 11, 19] and tree-decompositions[1] have a central role in algorithm design and graph theory, among other areas. While it is known that computing the treewidth of a graph is NP-complete [1], various approximation and/or parameterized algorithms, and exact exponential algorithms have been developed. The current Pareto front features, on n-vertex graphs of treewidth k, a polynomial-time $O(\sqrt{\log k})$-approximation algorithm [9], a $O(1.7347^n)$-time exact algorithm [10], a $2^{O(k)}n$-time 2-approximation algorithm [15], a $2^{O(k^3)}n$-time exact algorithm [4], a $2^{O(k^2)}n^4$-time exact algorithm, and a $2^{O(\frac{k \log k}{\varepsilon})}n^4$-time $(1+\varepsilon)$-approximation algorithm for any $\varepsilon > 0$ [16]; see the latter reference for a detailed overview of the state of the art.

On the complexity side, we did not know any sharp lower bound. The original reduction [1], as well as subsequent constructions strengthening the NP-hardness of Treewidth to graphs of maximum degree at most 9 [7], and even to cubic graphs [5], all rely on the NP-hardness of Cutwidth or Pathwidth on bounded-degree graphs. However, the known reductions for the latter results (see for instance [17]) incur a quadratic blow-up in the input size, and do not preserve any multiplicative[2] inapproximability. Therefore, prior to the current paper, a polynomial-time approximation scheme (PTAS) for Treewidth could not be ruled out under the sole assumption[3] that P $\neq$ NP, and the best lower bound based on the

[1] See their definition in Section 2.2.
[2] Some *additive* inapproximability is known for treewidth [6].
[3] Assuming the so-called *Small Subset Expansion* conjecture (that the edge expansion of sublinear vertex subsets is hard to approximate), it can be showed that any constant-approximation of Treewidth is NP-complete [20].

Exponential-Time Hypothesis[4] (ETH) only implied that Treewidth requires $2^{\Omega(\sqrt{n})}$ time (see for instance the appendix of [16]).

In this paper, we present a simple self-contained linear reduction from 3-SAT to Treewidth. This improves our understanding in the approximability and complexity of treewidth. We make the reduction modular so that depending on the instances of 3-SAT we start with, we derive the following three theorems.

▶ **Theorem 1.** *1.00005-approximating Treewidth is NP-hard.*

▶ **Theorem 2.** *Unless the ETH fails, Treewidth requires $2^{\Omega(n)}$ time on n-vertex graphs.*

▶ **Theorem 3.** *Unless the ETH fails, there is some constant $\delta > 1$ such that δ-approximating Treewidth requires $2^{n^{1-o(1)}}$ time on n-vertex graphs.*

The first theorem rules out a PTAS for Treewidth unless P = NP (and establishes its APX-hardness). There is still a very large gap between this inapproximability factor and the $O(\sqrt{\log \mathrm{OPT}})$-approximation algorithm of Feige, Hajiaghayi, and Lee [9], but at least the lower bound has moved for the first time. The second theorem establishes a tight ETH lower bound: running time $2^{O(n)}$ (for instance [10]) is best possible (under the ETH) to exactly compute treewidth. The third theorem combines the previous two hardness features. In particular, it (loosely) complements the $2^{O(\frac{k \log k}{\varepsilon})} n^{O(1)}$-time $(1+\varepsilon)$-approximation algorithm of Korhonen and Lokshtanov [16], in the sense that even for some fixed sufficiently small $\varepsilon > 0$ (depending on the ETH constant λ), the exponent $O(k \log k)$ in the running time cannot be improved to $O(k^{1-\eta})$ with $\eta > 0$.

Techniques. The known hardness constructions leverage the fact that on co-bipartite graphs (complements of bipartite graphs), tree-decompositions behave orderly. They are tame enough to allow a simple reduction from Cutwidth,[5] and actually treewidth and pathwidth are equal on co-bipartite graphs [1]. However, it seems challenging to design a linear reduction from 3-SAT to Treewidth on co-bipartite graphs. It is indeed unclear how to design a (binary) choice gadget in this restricted setting.

We thus move to co-tripartite graphs G with tripartition (A, B, C) (into cliques) where A encodes the clauses, and $B \cup C$ encodes the variables, with B associated to their positive form, and C, their negation. More precisely, a blow-up (where vertices are replaced by clique modules) of a semi-induced matching[6] between B and C constitutes our variable gadgets. For A, we add $7 = 2^3 - 1$ vertices for each clause, one for each partial satisfying assignment of the clause, each adjacent to the three modules corresponding to its literals; see Figure 1. This is to turn the disjunctive nature of 3-SAT into some conjunctive encoding, which better fits tree-decompositions.

Although co-tripartite graphs provide the greater generality (compared to co-bipartite graphs) that allows us to simply design choice gadgets, their tree-decompositions are still tame enough. In particular, they always admit a tree-decomposition of minimum width whose underlying tree is a subdivided claw, and whose three leaf bags contain A, B, and C, respectively. The crucial property that these tree-decompositions shares is that the bag of the unique degree-3 node is a vertex cover of the graph $I(G)$, defined as G deprived of the edges within the cliques A, B, and C. A reader familiar with the notion of *bramble* may

[4] The ETH asserts that there is a $\lambda > 0$ such that no algorithm solves n-variable 3-SAT in $O(\lambda^n)$ time [13].

[5] We will *not* need the definition of cutwidth.

[6] Here, an induced matching if not for the cliques B and C.

already observe that the edge set of $I(G)$ is a bramble of G, hence the vertex cover number of $I(G)$ minus one indeed lower bounds the treewidth of G; a useful fact for the "negative" 3-SAT instances.

This is where moving from co-bipartite graphs to co-tripartite graphs is decisive: Min Vertex Cover is polynomial-time solvable on bipartite graphs, but NP-complete on tripartite graphs. We can thus simply rely on the hardness of Min Vertex Cover, while this was impossible for the previous reductions based on co-bipartite graphs. This is indeed what we do. The treewidth upper bound for the "positive" 3-SAT instances can easily be derived via the *Cops and Robber game*. We will in fact *not* use *brambles* nor the *Cops and Robber game* in order to make the paper self-contained and more widely accessible. Readers comfortable with these notions will be able to skip Section 3.2, and find their own alternative (and shorter) proof in Section 3.3.

2 Definitions and notation

If i is a positive integer, we denote by $[i]$ the set of integers $\{1, 2, \ldots, i\}$.

2.1 Classical graph theory

We denote by $V(G)$ and $E(G)$ the set of vertices and edges of a graph G, respectively. For $S \subseteq V(G)$, the *subgraph of G induced by S*, denoted $G[S]$, is obtained by removing from G all the vertices that are not in S (together with their incident edges). A set $X \subseteq V(G)$ is *connected* (in G) if $G[X]$ has a single connected component, and *disconnected* otherwise. A graph is *co-tripartite* (resp. *co-bipartite*) if it is the complement of a tripartite graph (resp. bipartite) graph, or equivalently can have its vertex set partitioned into three (resp. two) cliques. A *vertex cover* of a graph G is a subset $S \subseteq V(G)$ such that every edge of G has at least one of its two endpoints in S.

We denote by $N_G(v)$ and $N_G[v]$, the open, respectively closed, neighborhood of v in G. For $S \subseteq V(G)$, we set $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$ and $N_G[S] := N_G(S) \cup S$. A *module* in G is a subset $Y \subseteq V(G)$ such that for every $u, v \in Y$, $N_G(u) \setminus Y = N_G(v) \setminus Y$. The *degree* $d_G(v)$ of a vertex $v \in V(G)$ is the size of $N_G(v)$, and the *maximum degree* of G is $\max_{v \in V(G)} d_G(v)$. In all the previous notations, we may omit the graph subscript if it is clear from the context. A *subdivided claw* is any tree with exactly three leaves. Note that any subdivided claw has exactly one vertex of degree 3, and apart from it and its three leaves, only vertices of degree 2.

2.2 Tree-decompositions and treewidth

A *tree-decomposition* of a graph G is a pair (T, β) where T is a tree and β is a map from $V(T)$ to $2^{V(G)}$ satisfying the following properties:

- for every $v \in V(G)$, $\{t \in V(T) \ : \ v \in \beta(t)\}$ induces a non-empty subtree of T, and
- for every $uv \in E(G)$, there is a $t \in V(T)$ such that $\{u, v\} \subseteq \beta(t)$.

The *width* of (T, β) is defined as $\max_{t \in V(T)} |\beta(t)| - 1$, and the *treewidth* of G, denoted by $\mathrm{tw}(G)$, is the minimum width of (T, β) taken among every tree-decomposition (T, β) of G. A *path-decomposition* is a tree-decomposition (T, β) where T is a path. And *pathwidth* is defined as *treewidth* with *path-decompositions* instead of *tree-decompositions*.

We may call $\beta(t)$ the *bag* of $t \in V(T)$. We also call *trace* of $v \in V(G)$ the set $\{t \in V(T) : v \in \beta(t)\}$. We may say that an edge $uv \in E(G)$ is *covered* by a tree-decomposition (T, β) (not a priori claimed to be one of G) if there is a $t \in V(T)$ such that $\{u, v\} \subseteq \beta(t)$. More specifically, the edge uv is *covered* by node t. A pair (T, β) where T is a tree and β is a map

from the nodes of T to subsets of some universe U is a (valid) tree-decomposition (of some graph) if the trace of each element of U induces a non-empty subtree of T. It further is a tree-decomposition *of* G if $U = V(G)$ and every edge of G is covered by (T, β).

3 Treewidth hardness

Let φ be a 3-CNF formula. We denote by $x_1, \ldots, x_n$ the variables of φ, and by $C_1, \ldots, C_m$, its clauses, each of them on exactly three literals.

3.1 Construction of $G(\varphi)$

We build a co-tripartite graph $G := G(\varphi)$ with $2\gamma n + 7m$ vertices, where $\gamma = O(1)$ is a natural number to be instantiated. Set $V(G)$ is partitioned into (A, B, C) with $G[A]$, $G[B]$, and $G[C]$ each being a clique. The set A represents the clauses, and $B \cup C$, the variables, with B corresponding to their positive form, and C their negation.

For every variable x_i, we add γ vertices $b_i^1, \ldots, b_i^\gamma$ to B, and γ vertices $c_i^1, \ldots, c_i^\gamma$ to C. We keep the value of the natural number γ generic. The only constraint on γ is that there is *no literal* with p positive occurrences and q negative occurrences in φ such that $4p + 3q > \gamma$. To clarify, every clause of φ containing $\neg x_i$ counts for a *positive* occurrence of literal $\neg x_i$, and every clause of φ containing x_i counts for a negative occurrence of literal $\neg x_i$.

(For the concreteness of showing Theorem 1, the reader can assume that every variable appears exactly twice positively and exactly twice negatively, and $\gamma := 4 \cdot 2 + 3 \cdot 2 = 14$.) We set $B(x_i) := \{b_i^1, \ldots, b_i^\gamma\}$ and $C(x_i) := \{c_i^1, \ldots, c_i^\gamma\}$. For every $i, i' \in [n]$ and $h, h' \in [\gamma]$, the two vertices $b_i^h, c_{i'}^{h'}$ are made adjacent if and only if $i = i'$. Each set $B(x_i)$ and $C(x_i)$ will remain a module in $G(\varphi)$, and we will therefore rarely refer to b_i^h individually.

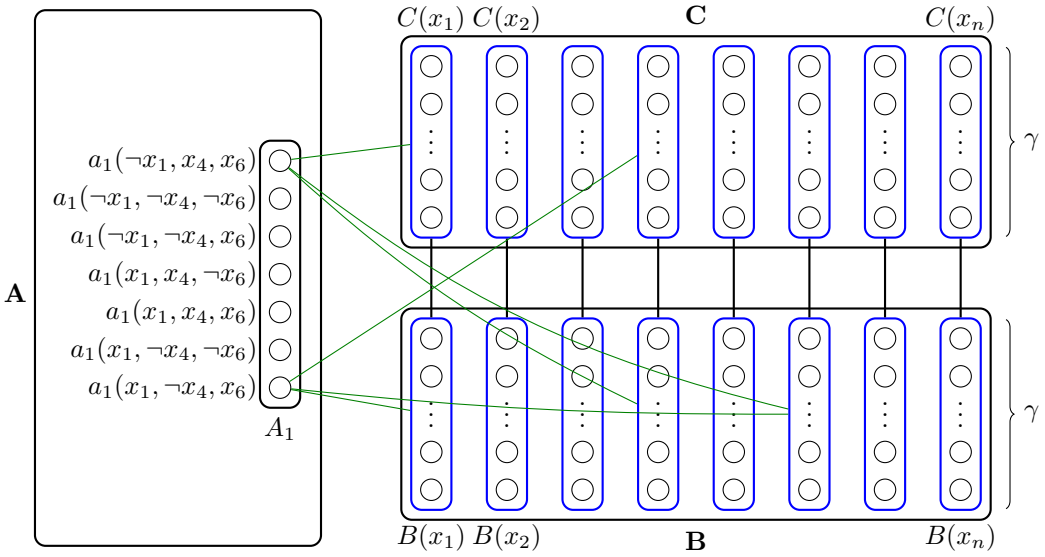

Figure 1 Illustration of $G(\varphi)$ with $n = 8$. For the sake of not cluttering the picture, we did not draw the edges within the cliques A, B, and C. We also only represented the vertices of A encoding the clause $C_1 = x_1 \vee \neg x_4 \vee x_6$, i.e., A_1, and only drew the edges incident to $a_1(x_1, \neg x_4, x_6)$ and to $a_1(\neg x_1, x_4, x_6)$. The blue vertical boxes are modules.

For every clause $C_j = \ell_1 \vee \ell_2 \vee \ell_3$ with $\ell_p = (\neg) x_{i_p}$ for $p \in [3]$, we add a set A_j of 7 vertices to A, one for each assignment of its three variables satisfying C_j. For $(s_1, s_2, s_3) \in (\{\ell_1, \neg\ell_1\} \times$

$\{\ell_2, \neg\ell_2\} \times \{\ell_3, \neg\ell_3\}) \setminus \{(\neg\ell_1, \neg\ell_2, \neg\ell_3)\}$, we denote by $a_j(s_1, s_2, s_3)$ the corresponding vertex of A, while syntactically replacing $\neg\neg x$ by x. For each $p \in [3]$, we make $a_j(s_1, s_2, s_3)$ fully adjacent to $B(x_{i_p})$ if $s_p = x_{i_p}$, or to $C(x_{i_p})$ if $s_p = \neg x_{i_p}$. This finishes the construction of G; see Figure 1.

3.2 Tree-decompositions of co-tripartite graphs

Here we show that every co-tripartite graph G with tripartition (A, B, C) has a tree-decomposition (T, β) of width $\mathrm{tw}(G)$ such that T is a subdivided claw, and the bag of its vertex of degree 3 is a vertex cover of $I(G)$, the graph G deprived of the edges within the cliques A, B, and C. A reader familiar with *brambles* may observe that the family of pairs $\mathcal{B} := \{\{u, v\} \ : \ uv \in E(G) \setminus \bigcup_{X \in \{A,B,C\}} E(G[X])\}$ is a bramble of G, and that a hitting set of $\mathcal{B}$ is, by definition, a vertex cover of $I(G)$; thereby reaching the desired treewidth lower bound. They may then skip Section 3.2. We chose this presentation, as Lemma 6 better prepares to Sections 3.3 and 3.4.

We start by recalling a classical lemma.

▶ **Lemma 4.** *Let G be a graph, X be a clique of G, and (T, β) be a tree-decomposition of G. Then there is a node $t \in V(T)$ such that $X \subseteq \beta(t)$.*

Proof. Every two vertices $x, y \in X$ have to appear together in some bag of (T, β). We conclude, as subtrees in a tree have the Helly property. ◀

Lemma 4 justifies the existence of t_A, t_B, t_C in the assumption of the following lemma.

▶ **Lemma 5.** *Let G be a co-tripartite graph with tripartition (A, B, C), and (T, β) be a tree-decomposition of G. Let $t_A, t_B, t_C \in V(T)$ be such that $A \subseteq \beta(t_A)$, $B \subseteq \beta(t_B)$, and $C \subseteq \beta(t_C)$. Let T' be the minimal subtree of T containing t_A, t_B, t_C. Then (T', β) is a tree-decomposition of G (where β is used, for simplicity's sake, for its restriction to $V(T')$).*

Proof. In a tree, the intersection of two subtrees is also a subtree. Thus (T', β) is a valid tree-decomposition (of some graph). We show that (T', β) is a tree-decomposition of G. As $A \subseteq \beta(t_A)$, $B \subseteq \beta(t_B)$, and $C \subseteq \beta(t_C)$, every vertex of G is in some bag of (T', β). We next argue that no node of $V(T) \setminus V(T')$ is actually useful to cover the edges of G. Let $t \in V(T) \setminus V(T')$ and t' be the node of T' in the shortest path from t to $V(T')$. It holds that $\beta(t) \subseteq \beta(t')$, as otherwise the trace of any vertex of $\beta(t) \setminus \beta(t')$ in (T, β) is disconnected. Hence t does not cover edges that are not already covered by t'. ◀

If G is a co-tripartite graph with tripartition (A, B, C), we recall that $I(G)$ is the graph obtained from G by removing all the edges within the cliques A, B, and C.

▶ **Lemma 6.** *Let G be a co-tripartite graph with tripartition (A, B, C). Then G has a tree-decomposition (T, β) of width $tw(G)$ such that*

- *T is a subdivided claw,*
- *the three leaves t_A, t_B, t_C of T satisfy $X \subseteq \beta(t_X)$ for every $X \in \{A, B, C\}$, and*
- *the unique degree-3 node $t_\curlywedge$ of T is such that $\beta(t_\curlywedge)$ is a vertex cover of $I(G)$.*

Proof. By applying Lemma 5 with a tree-decomposition of minimum width, G has a tree-decomposition (T, β) of width $\mathrm{tw}(G)$ such that T is a subdivided claw whose three leaves t_A, t_B, t_C verify $X \subseteq \beta(t_X)$ for every $X \in \{A, B, C\}$. (Indeed, if the minimal subtree connecting t_A, t_B, t_C is a path, one can simply add a neighbor with the same bag to whichever of t_A, t_B, t_C is not yet a distinct leaf node.) Hence the first two items of the lemma are satisfied.

Let t_Λ be the unique node of T with degree 3. Assume for the sake of contradiction that an edge $uv \in E(I(G))$ is such that $\{u, v\} \cap \beta(t_\Lambda) = \emptyset$. Without loss of generality, let us assume that $u \in A$ and $v \in B$. We claim that no bag of (T, β) may then include $\{u, v\}$. Indeed, vertex u may only be present in the path of T going from t_A to t_Λ, excluding t_Λ itself, while v is only present within the path from t_B to t_Λ (excluded). ◀

When we later use Lemma 6, only the third item will actually matter.

3.3 If one can satisfy m' clauses, then $\mathbf{tw}(G) \leqslant \gamma n + 7m - m' + \gamma - 1$

We go back to the particular co-tripartite graph $G := G(\varphi)$ with tripartition (A, B, C) constructed in Section 3.1, and exhibit, when a truth assignment satisfies m' clauses of φ, a tree-decomposition (T, β) of G where every bag has size at most $\gamma(n+1) + 7m - m'$. Let us recall that $|A| = 7m$ and $|B| = |C| = \gamma n$. The tree T is a subdivided claw. Let t_Λ be its degree-3 vertex, and t_A, t_B, t_C be its three leaves.

The bag of t_Λ. Let $\mathcal{A}$ be a truth assignment of $x_1, \ldots, x_n$ satisfying $m' \leqslant m$ clauses of φ. We denote by $\mathcal{A}^+ \subseteq [n]$ (resp. $\mathcal{A}^- \subseteq [n]$) the set of indices i such that $\mathcal{A}$ sets x_i to true (resp. to false). Thus $(\mathcal{A}^+, \mathcal{A}^-)$ partitions $[n]$. Let us define the bag of t_Λ. We set

$$B' := \bigcup_{i \in \mathcal{A}^+} B(x_i), \text{ and } C' := \bigcup_{i \in \mathcal{A}^-} C(x_i).$$

We define the subset $A' \subset A$ of size $7m - m'$, starting from A and removing, for each satisfied clause C_j, the vertex $a_j(s_1, s_2, s_3) \in A_j$ such that s_1, s_2, s_3 are all satisfied by $\mathcal{A}$. Thus $|A'| = |A| - m' = 7m - m'$. We set $\beta(t_\Lambda) := A' \cup B' \cup C'$.

Path-decompositions from t_Λ to the three leaf nodes. We now give the path-decompositions from t_Λ to t_X for each $X \in \{A, B, C\}$, such that: $X \subseteq \beta(t_X)$, and for every node $t \in V(T)$ in the path from t_Λ to t_X, for each $Y \in \{A, B, C\} \setminus X$, $\beta(t) \cap Y \subseteq Y'$. We will later see that path-decompositions satisfying the previous conditions combine to define a valid tree-decomposition.

The path-decomposition from t_Λ to t_B goes as follows. Initially, the *active node* is t_Λ. For increasing indices $i \in \mathcal{A}^-$, add a neighbor t' to the current active node $t \in V(T)$, and set $\beta(t') := \beta(t) \cup B(x_i)$. Then add a neighbor t'' to t', and set $\beta(t'') := \beta(t') \setminus C(x_i)$. Node t'' then becomes the active node. The active node after the last iteration in $\mathcal{A}^-$ is t_B. Note that $\beta(t_B) = A' \cup B$. The path-decomposition from t_Λ to t_C is defined analogously by replacing $\mathcal{A}^-$ with $\mathcal{A}^+$, and swapping the roles of $B(x_i)$ and $C(x_i)$.

We finally describe the path-decomposition from t_Λ to t_A. The path of T from t_Λ to t_A is: $t_\Lambda = t'_0, t_1, t'_1, t_2, t'_2, \ldots, t_n, t'_n = t_A$. For every integer i from 1 to n, we set $\beta(t_i) := \beta(t'_{i-1}) \cup Z_i$ where Z_i is the set of neighbors in A of vertices in $B(x_i)$ if $i \in \mathcal{A}^+$ (equivalently if $B(x_i) \subseteq B' \cup C'$), or of vertices in $C(x_i)$ if $i \in \mathcal{A}^-$ (equivalently if $C(x_i) \subseteq B' \cup C'$). And we set $\beta(t'_i) := \beta(t_i) \setminus (B(x_i) \cup C(x_i))$. Note that $\beta(t_A) = A$.

This finishes the construction of (T, β). We have three properties to check: the trace of every vertex of G (in (T, β)) makes a non-empty tree, all edges of G are covered by (T, β), and every bag of (T, β) has size at most $\gamma(n+1) + 7m - m'$.

The trace of every $v \in V(G)$ induces a non-empty tree. The simplest case is when $v \in A'$, as A' is included in *every* bag of (T, β). For each $X \in \{A, B, C\}$, the trace (in (T, β)) of any $v \in X \setminus X'$ is a path ending at t_X, since $v \in X \subseteq \beta(t_X)$, and in the path of T going from t_Λ to t_X, vertex v is added to the bag of some node t, and never removed in the path

from t to t_X. Furthermore, if $\{Y, Z\} = \{A, B, C\} \setminus X$, no vertex of $X \setminus X'$ is part of a bag in the path of T from t_Y to t_Z.

It remains to check that the traces of vertices of $B' \cup C'$ induce subtrees of T. This is indeed the case since every vertex $v \in B'$ (resp. $v \in C'$) is in all the bags along the path from $t_\wedge$ to t_B (resp. to t_C), while no vertex in B (resp. in C) is ever *added* to a bag in the paths from $t_\wedge$ to t_A and from $t_\wedge$ to t_C (resp. to t_B).

Every edge $e \in E(G)$ is covered by (T, β). For each $X \in \{A, B, C\}$, every edge within the clique X is covered by t_X, as $X \subseteq \beta(t_X)$. Therefore, we shall just argue that every edge of $E(I(G))$ is covered by (T, β).

We start with the edges between B and C. Recall that all these edges are of the form $b_i^h c_i^{h'}$ with $i \in [n]$ and $h, h' \in [\gamma]$. If $i \in \mathcal{A}^-$, there is, in the path of T from $t_\wedge$ to t_B, a node where we just added $B(x_i)$, while $C(x_i) \subseteq C'$ is still present (and $C(x_i)$ is removed in the following bag). Thus this node covers $b_i^h c_i^{h'}$, as well as every edge between $B(x_i)$ and $C(x_i)$. Analogously, if $i \in \mathcal{A}^+$, there is in the path of T from $t_\wedge$ to t_C, a node where we just added $C(x_i)$, while $B(x_i) \subseteq B'$ is still present.

We now consider edges between A and $B \cup C$. Note that every edge between A' and B (resp. C) is covered by t_B (resp. t_C) since $\beta(t_B) = A' \cup B$ (resp. $\beta(t_C) = A' \cup C$). We thus turn our attention to edges between $A \setminus A'$ and $B \cup C$. By construction, no vertex of $A \setminus A'$ has a neighbor in $(B \cup C) \setminus (B' \cup C')$. Hence we are only left with edges between $A \setminus A'$ and $B' \cup C'$.

Fix an edge $uv \in E(G)$ with $u \in A \setminus A'$ and $v \in B' \cup C'$, and let $i \in [n]$ be such that v belongs to $\beta(t_i)$ but not to $\beta(t_i')$. Note that i is well-defined and corresponds to the index such that $v \in (B(x_i) \cup C(x_i)) \cap (B' \cup C')$. As $N_G((B(x_i) \cup C(x_i)) \cap (B' \cup C')) \cap A = N_G(v) \cap A \subseteq \beta(t_i)$, edge uv is covered by t_i (since $u \in N_G(v) \cap A$).

Every bag of (T, β) has size at most $\gamma(n+1) + 7m - m'$. First note that $|\beta(t_\wedge)| = \gamma n + 7m - m'$ and that each bag along the path of T between t_B and t_C has either size $\gamma n + 7m - m'$ or size $\gamma(n+1) + 7m - m'$. We finally claim that for every $i \in [n]$, $|\beta(t_i)| \leqslant |\beta(t'_{i-1})| + \gamma$ and $|\beta(t'_i)| \leqslant |\beta(t'_{i-1})|$. For the former inequality, notice that vertices of $B(x_i)$ (resp. $C(x_i)$) have the same four neighbors within each A_j such that x_i appears positively (resp. negatively) in C_j, and the same three neighbors within each A_j such that x_i appears negatively (resp. positively) in C_j, and no other neighbor in A. Further recall that every *literal* with p positive occurrences and q negative occurrences in φ satisfies $\gamma \geqslant 4p + 3q$. Consequently, to check that $|\beta(t'_i)| \leqslant |\beta(t'_{i-1})|$, simply recall that the bag of t'_i is that of t_i deprived of γ vertices. Therefore every bag along the path from $t_\wedge$ to t_A has size at most $|\beta(t'_0)| + \gamma = |\beta(t_\wedge)| + \gamma = \gamma(n+1) + 7m - m'$.

We have thus established that the treewidth of G is at most $\gamma n + 7m - m' + \gamma - 1$.

3.4 If at most m'' clauses are satisfiable, then $\mathsf{tw}(G) \geqslant \gamma n + 7m - m'' - 1$

By Lemma 6, there is a tree-decomposition of G of width $\mathrm{tw}(G)$ with a bag (that of $t_\wedge$) that is a vertex cover of $I(G)$. Therefore, we shall just argue that every vertex cover of $I(G)$ has size at least $\gamma n + 7m - m''$, when every truth assignment satisfies at most m'' clauses of φ. We fix any inclusion-wise minimal vertex cover S of $I(G)$, and set $A' := A \cap S$, $B' := B \cap S$, $C' := C \cap S$. As, for every $i \in [n]$, every vertex of $B(x_i)$ is adjacent to every vertex of $C(x_i)$, it necessarily holds that $B(x_i) \subseteq S$ or $C(x_i) \subseteq S$. As $B(x_i)$ (resp. $C(x_i)$) is a module (in G and in $I(G)$) and S is inclusion-wise minimal, it further holds that $S \cap (B(x_i) \cup C(x_i)) \in \{B(x_i), C(x_i), B(x_i) \cup C(x_i)\}$. We show the following replacement lemma, to obtain a more suitable and non-larger vertex cover S' of $I(G)$.

▶ **Lemma 7.** *There is a vertex cover S' of $I(G)$ such that $S' \cap (B(x_i) \cup C(x_i)) \in \{B(x_i), C(x_i)\}$ for every $i \in [n]$, and $|S'| \leqslant |S|$.*

Proof. We initialize the set S' to S. For every $i \in [n]$ such that $B(x_i) \cup C(x_i) \subseteq S$, we remove $C(x_i)$ from S', and add $N_G(C(x_i)) \cap A$ to S'. It was already observed that the condition imposed on γ implies that $|N_G(C(x_i)) \cap A| \leqslant \gamma$, so the size of S' may only decrease (as $|C(x_i)| = \gamma$). The replacement preserves the property of being a vertex cover of $I(G)$, since all the neighbors of $C(x_i)$ end up in S' whenever $C(x_i)$ is removed. ◀

By Lemma 7, we just need to show that $|S'| \geqslant \gamma n + 7m - m''$. We note that $S' \cap (B \cup C)$ has size γn and defines a truth assignment $\mathcal{A}$ of $x_1, \ldots, x_n$: $\mathcal{A}$ sets x_i to true if $S' \cap (B(x_i) \cup C(x_i)) = B(x_i)$, and to false if $S' \cap (B(x_i) \cup C(x_i)) = C(x_i)$. Seeing S' as $(A \setminus \widehat{A}) \cup (S' \cap (B \cup C))$ for some $\widehat{A} \subseteq A$, it holds that $|S'| = 7m - |\widehat{A}| + \gamma n$. For S' to be a vertex cover of $I(G)$, a vertex v of $\widehat{A}$ has to have all its neighbors within $B \cup C$ included in S'. This can only happen if $v = a_j(s_1, s_2, s_3)$, C_j is satisfied by $\mathcal{A}$, and the literals s_1, s_2, s_3 are all satisfied by $\mathcal{A}$. This implies that $|\widehat{A}| \leqslant m''$. Therefore, $|S'| \geqslant \gamma n + 7m - m''$. We conclude that the treewidth of G is at least $\gamma n + 7m - m'' - 1$.

3.5 Instantiating the reduction

We now apply the previous reduction to hard (to approximate) 3-SAT instances in order to show Theorems 1–3. Recall that as long as $\gamma = O(1)$, our reduction produces graphs on $O(n+m)$ vertices from n-variable m-clause 3-SAT formulas. In particular, it is a polynomial reduction, and linear in n whenever $m = O(n)$.

Proof of Theorem 1. Following Berman, Karpinski, and Scott [2], we call $(3, 2B)$-*SAT* formula a 3-CNF formula where every clause has exactly three literals, and every variable appears exactly twice positively, and exactly twice negatively. The same authors showed that, for any $\varepsilon > 0$, it is NP-complete to distinguish within m-clause $(3, 2B)$-SAT formulas those for which at least $(1-\varepsilon)m$ clauses are satisfiable from those for which at most $(\frac{1015}{1016} + \varepsilon)m$ clauses are satisfiable.

We observe that m-clause $(3, 2B)$-SAT formulas have $n = \frac{3m}{4}$ variables. Substituting $m' := (1-\varepsilon)m$, $m'' := (\frac{1015}{1016} + \varepsilon)m$, $\gamma := 4 \cdot 2 + 3 \cdot 2 = 14$, and $n = \frac{3m}{4}$, we get that it is NP-hard to distinguish graphs of treewidth at most $(10.5 + 6 + \varepsilon)m + 13$ from graphs of treewidth at least $(10.5 + 7 - \frac{1015}{1016} - \varepsilon)m - 1$. We conclude as

$$\frac{(17.5 - \frac{1015}{1016} - \varepsilon)m - 1}{(16.5 + \varepsilon)m + 13} > 1.00005$$

for sufficiently small ε and large m.

We observe that one can increase the inapproximability gap by performing (with the required adjustments) our reduction from Max-2-SAT where every variable has at most four occurrences. This problem has a better known inapproximability factor than Max-$(3, 2B)$-SAT [2], and this would also allow to decrease the value of γ (to 7).

Proof of Theorem 2. The Sparsification Lemma of Impagliazzo, Paturi, and Zane [14] implies that, under the ETH, n-variable 3-SAT requires $2^{\Omega(n)}$ time even within formulas where every variable appears at most B times, for some constant B. We denote this fragment of 3-SAT by 3-SAT-B. We set $\gamma := 4B$ so that the condition on γ is verified, and we duplicate $\gamma + 1$ times the initial hard formula φ' of 3-SAT-B. Each copy is on a pairwise disjoint set of variables. We thus create an instance φ of 3-SAT-B on $(\gamma + 1)n$ variables,

such that it takes $2^{\Omega(n)}$ time to tell whether all m clauses of φ are satisfiable, or at most $m-(\gamma+1)$ of its clauses are, unless the ETH fails.

For $G(\varphi)$, which has $7m+2\gamma n \leqslant 7Bn+2\gamma n = O(n)$ vertices, this translates into distinguishing whether its treewidth is at most $\gamma n+6m+\gamma-1$ or at least $\gamma n+7m-(m-(\gamma+1))-1=\gamma n+6m+\gamma$, implying Theorem 2.

Proof of Theorem 3. Building upon Håstad's polynomial-time inapproximability of 3-SAT [12], Moshkovitz and Raz proved that, unless the ETH fails, approximating n-variable 3-SAT within ratio smaller than $\frac{8}{7}$ requires time $2^{n^{1-o(1)}}$ [18]. Applyling the Sparsification Lemma [14] on the instances produced by Moshkovitz and Raz's reduction, the following is attained.

▶ **Theorem 8** ([14, 18], see for instance [8]). *For any constant $r \in (\frac{8}{7},1)$, there is an integer B, such that any r-approximation algorithm for n-variable 3-SAT-B requires $2^{n^{1-o(1)}}$ time.*

In effect, the proof of Theorem 8 takes a 3-CNF formula φ' on n variables and returns $2^{o(n)}$ formulas (subexponential blow-up due to the Sparsification Lemma) $(\varphi_i)_i$ of 3-SAT-B on $N := n^{1+o(1)}$ variables (quasilinear blow-up due to the PCP of Moshkovitz and Raz) and $m \leqslant BN$ clauses such that, if φ' is satisfiable, then at least $0.99m$ clauses of some φ_i are satisfiable, and if instead φ' is unsatisfiable, then at most $0.88m$ clauses of each φ_i is satisfiable.

To decide which case holds, we run our reduction on every instance φ_i. We again set $\gamma := 4B$, and observe that the obtained $G(\varphi_i)$ has $n^{1+o(1)}$ vertices. This creates a gap between the treewidth upper bound of $\gamma N+6.01m+\gamma-1$ (when φ' is satisfiable) and the treewidth lower bound of $\gamma N+6.12m-1$ (when φ' is unsatisfiable). For sufficiently large m, this makes an approximability gap of $\delta>1$ (for some δ), which implies Theorem 3.

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 Piotr Berman, Marek Karpinski, and Alex D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. *Electron. Colloquium Comput. Complex.*, TR03-049, 2003. URL: `https://eccc.weizmann.ac.il/eccc-reports/2003/TR03-049/index.html`, `arXiv:TR03-049`.

3 Umberto Bertele and Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973.

4 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`.

5 Hans L. Bodlaender, Édouard Bonnet, Lars Jaffke, Dusan Knop, Paloma T. Lima, Martin Milanic, Sebastian Ordyniak, Sukanya Pandey, and Ondrej Suchý. Treewidth is np-complete on cubic graphs. In Neeldhara Misra and Magnus Wahlström, editors, *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, September 6-8, 2023, Amsterdam, The Netherlands*, volume 285 of *LIPIcs*, pages 7:1–7:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: `https://doi.org/10.4230/LIPIcs.IPEC.2023.7`, `doi:10.4230/LIPICS.IPEC.2023.7`.

6 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. URL: `https://doi.org/10.1006/jagm.1995.1009`, `doi:10.1006/JAGM.1995.1009`.

7 Hans L. Bodlaender and Dimitrios M. Thilikos. Treewidth for graphs with small chordality. *Discret. Appl. Math.*, 79(1-3):45–61, 1997. `doi:10.1016/S0166-218X(97)00031-0`.

8 Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. On subexponential and fpt-time inapproximability. *Algorithmica*, 71(3):541–565, 2015. URL: https://doi.org/10.1007/s00453-014-9889-1, doi:10.1007/S00453-014-9889-1.

9 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.

10 Fedor V. Fomin, Ioan Todinca, and Yngve Villanger. Large induced subgraphs via triangulations and CMSO. *SIAM J. Comput.*, 44(1):54–87, 2015. doi:10.1137/140964801.

11 Rudolf Halin. S-functions for graphs. *Journal of geometry*, 8:171–186, 1976.

12 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

13 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.

14 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.

15 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.

16 Tuukka Korhonen and Daniel Lokshtanov. An improved parameterized algorithm for treewidth. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 528–541. ACM, 2023. doi:10.1145/3564246.3585245.

17 Burkhard Monien and Ivan Hal Sudborough. Min Cut is NP-Complete for Edge Weighted Treees. *Theor. Comput. Sci.*, 58:209–229, 1988. doi:10.1016/0304-3975(88)90028-X.

18 Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5):29:1–29:29, 2010. doi:10.1145/1754399.1754402.

19 Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

20 Yu (Ledell) Wu, Per Austrin, Toniann Pitassi, and David Liu. Inapproximability of treewidth and related problems. *J. Artif. Intell. Res.*, 49:569–600, 2014. URL: https://doi.org/10.1613/jair.4030, doi:10.1613/JAIR.4030.