Lean Tree-Cut Decompositions: Obstructions and Algorithms

Archontia C. Giannopoulou, O-Joung Kwon, Jean-Florent Raymond, Dimitrios M. Thilikos

Lean Tree-Cut Decompositions: Obstructions and Algorithms

Archontia C. Giannopoulou
LaS team, Technische Universität Berlin, Germany
archontia.giannopoulou@tu-berlin.de

O-joung Kwon
Department of Mathematics, Incheon National University, South Korea
ojoungkwon@inu.ac.kr

Jean-Florent Raymond
LaS team, Technische Universität Berlin, Germany
raymond@tu-berlin.de

Dimitrios M. Thilikos
AlGCo project-team, LIRMM, Université de Montpellier, CNRS, Montpellier, France
sedthilk@thilikos.info

Abstract

The notion of tree-cut width has been introduced by Wollan in [*The structure of graphs not admitting a fixed immersion*, Journal of Combinatorial Theory, Series B, 110:47–66, 2015]. It is defined via tree-cut decompositions, which are tree-like decompositions that highlight small (edge) cuts in a graph. In that sense, tree-cut decompositions can be seen as an edge-version of tree-decompositions and have algorithmic applications on problems that remain intractable on graphs of bounded treewidth. In this paper, we prove that every graph admits an optimal tree-cut decomposition that satisfies a certain Menger-like condition similar to that of the lean tree decompositions of Thomas [*A Menger-like property of tree-width: The finite case*, Journal of Combinatorial Theory, Series B, 48(1):67–76, 1990]. This allows us to give, for every $k \in \mathbb{N}$, an upper-bound on the number immersion-minimal graphs of tree-cut width k. Our results imply the *constructive* existence of a linear FPT-algorithm for tree-cut width.

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

Keywords and phrases tree-cut width, lean decompositions, immersions, obstructions, parameterized algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2019.32

Related Version The full version of the paper can also be found on arXiv at `https://arxiv.org/pdf/1808.00863`.

Funding *Archontia C. Giannopoulou*: Supported by the ERC consolidator grant DISTRUCT-648527.
O-joung Kwon: Supported by the ERC consolidator grant DISTRUCT-648527 and by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (No. NRF-2018R1D1A1B07050294).
Jean-Florent Raymond: Supported by ERC consolidator grant DISTRUCT-648527.
Dimitrios M. Thilikos: Supported by projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010). Supported by the Research Council of Norway and the French Ministry of Europe and Foreign Affairs, via the Franco-Norwegian project PHC AURORA 2019.

Acknowledgements We are grateful to Michał Pilipczuk and Marcin Wrochna for extensive discussions about the proof of Theorem 1.

1 Introduction

Menger's theorem is a classic result in graph theory and arguably one of the most used. It can be informally stated as follows: two vertex sets of a graph can be linked by k vertex-disjoint paths if and only if they cannot be separated by the removal of k vertices [25]. A significant feature of this result is that it provides a concise certificate for the optimality (in terms of cardinality) of a collection of k disjoint paths between two sets, in the form of a separator of size k. Many variants of Menger's theorem are known, including edge and directed versions [25, 19].

In 1990, Thomas [31] obtained a similar duality between paths and separators in tree decompositions. Tree decompositions are central objects in the theory of graph minors of Robertson and Seymour. Introduced in the early times of their Graph Minors series [28], they soon became the standard way to approach problems related to minors (such as Disjoint Paths, Feedback Vertex Set, etc.) and to minor-closed graph classes. The result of Thomas is that every graph admits an optimal tree decomposition, said to be *lean*, where the following holds: two bags of the decomposition[1] can be connected by k vertex-disjoint paths if and only if they cannot be separated by the removal of a bag on at most k vertices. That is, while Menger's theorem provides a set of k vertices separating the two bags, Thomas proved that such vertices (again forming a concise certificate of optimality) can moreover be found in a unique bag. Lean tree decompositions had important applications as they have been used to optimize parameter dependencies and to simplify arguments in proofs of the Graph Minors series [2, 9].

The objects of our attention in this paper are tree-like decompositions of graphs called *tree-cut decompositions*. These decompositions, introduced by Wollan in [32], highlight small (edge) cuts in graphs and are particularly suited to deal with problems related to the immersion ordering[2] and immersion-closed graph classes. (Comparatively, tree decompositions display small vertex separators and are tailored for minor-related problems.) A strength of tree-cut decompositions compared to other decompositions defined by edge cuts (such as cutwidth orderings or carving decompositions) is that they enjoy two highly desirable properties that mirror those of tree decompositions: (a) they admit an excluded wall immersion theorem [32] (comparable to the excluded grid minor theorem of Robertson and Seymour [29]) and (b) they can be used for dynamic programming, additionally for problems that cannot be tackled under the bounded-treewidth framework [11, 21, 15]. Thus they are believed to be a credible translation of tree decompositions to the settings of immersions. The importance of tree decomposition-based techniques in graph algorithms is an additional motivation to study their possible counterpart in the immersion world. Our first contribution is to show that paths and cuts in tree-cut decompositions obey a duality as in the aforementioned result of Thomas.

▶ **Theorem 1.** *Every graph has an optimal tree-cut decomposition that is lean.*

For the above statement to be meaningful, we adapted the notion of leanness to the setting of tree-cut decomposition; in particular it relates edge-disjoint paths with edge cuts.[3] A similar notion of leanness has been previously studied in [16] for the simpler setting of cutwidth orderings. We also note that analogs of lean tree decompositions appeared for

[1] We follow here the standard terminology about tree-decompositions, see Section 6 for details.
[2] The immersion ordering is a partial order on graphs that has properties similar to the minor ordering, see Section 2 for details.
[3] The formal definition is given in Section 2.

several different width parameters such as θ-tree-width [6, 13], pathwidth [24], directed path-width [22], DAG-width [23], rank-width [26], linear-rankwidth [20], profile- and block-width [9], matroid treewidth [12, 1, 9] and matroid branchwidth [12].

To prove Theorem 1, we design an improvement procedure that, in a finite number of steps, transforms a tree-cut decomposition that is not lean into a lean one.

Tree-cut decompositions can be used to define the graph parameter of *tree-cut width*, in the same way that treewidth can be defined via tree decompositions. Our second result pertains to *obstructions* for bounded tree-cut width. We say that a graph is an (*immersion*)-*obstruction* for tree-cut width at most k if it has tree-cut width more than k and all its proper immersions have tree-cut width at most k. Intuitively, such graphs delimit the border between the graphs of small tree-cut width (i.e. at most k) and the rest. As a consequence of the results of Robertson and Seymour in [27], the set $\mathbf{obs}(k)$ of obstructions for tree-cut width at most k is finite, for every $k \in \mathbb{N}$. Unfortunately the techniques used to prove this result (namely, properties of well-quasi-orders) do not provide explicit upper-bounds on its size. We consider here the related question of estimating the size of the elements of $\mathbf{obs}(k)$. This question has received significant attention in related settings. In [24], Lagergren gave exponential upper-bounds on the order of minor-obstructions to graphs of pathwidth or treewidth at most k ($2^{O(k^4)}$ and $2^{2^{O(k^5)}}$), respectively. Moreover, the size of immersion-obstructions to graphs of cutwidth at most k is known to lie between $(3^{k-5}-1)/2$ [17] and $2^{O(k^3 \log k)}$ [16]. As noted in [5], these size estimations can be important in practice. Our result on this topic is a constructive upper-bound on the size of obstructions for bounded tree-cut width.

▶ **Theorem 2.** *Let $k \in \mathbb{N}$. If G has tree-cut width more than k and every proper immersion of G has tree-cut width at most k, then G has at most $2^{2^{O(k^4)}}$ vertices.*

Our proof has two main ingredients. The first one is an encoding into a finite number of types of the features of a graph that are relevant when computing tree-cut width, in the fashion of the folios of [30] and the protrusion machinery of [4], but adapted to the different setting studied here (and similar in this sense to the techniques developed in [16, 7]). The second ingredient is our aforementioned result on leanness. Informally, Theorem 1 allows us, when "redundancy" has been identified in a graph (using the first ingredient), to obtain an immersion that has the same tree-cut width and less vertices. To the best of our knowledge, no explicit upper-bound concerning $\mathbf{obs}(k)$ was known before.

The third problem that we consider is of algorithmic nature: how fast can the tree-cut width of a graph be computed? This problem is NP-hard [21] and therefore has been approached through parameterized complexity and approximation algorithms. A consequence of the aforementioned result of Robertson and Seymour [27] and a result in [18] is the existence – and the existence only – of an FPT algorithm that decides, given a graph G and $k \in \mathbb{N}$, whether G has tree-cut width at most k. The first step towards a constructive FPT algorithm for tree-cut width was achieved by Kim et al. [21] who designed a $2^{O(k^2 \log k)} \cdot n^2$-time factor 2 approximation algorithm. Our third result is the first constructive parameterized algorithm for tree-cut width.

▶ **Theorem 3.** *It is possible to construct an algorithm that given an n-vertex graph G and an integer r decides whether G has tree-cut width more than r in $f(r) \cdot n$ steps, where f is some recursive function.*

Our algorithm relies on the fact that graphs of small tree-cut width have small treewidth and on our result on the size of the obstructions. Graphs of large treewidth can be immediately rejected, while graphs of small treewidth can be searched for the existence of an obstruction to tree-cut width at most k using standard techniques on tree decompositions. We leave as an open problem to determine the optimal order of magnitude for f.

Organization of this extended abstract. In Section 2, we give preliminary definitions. The proofs of Theorem 1 is given in Section 3. In Sections 4 and 5 we introduce the machinery used to prove Theorems 2 and 3 whose proofs are given in Section 6. Finally, in Section 7, we provide some discussion on how to bound the parametric dependence of the algorithm of Theorem 3. Due to the space constraints, the proofs of the lemmas marked by ★ have been omitted. They can be found in the full version [14].

2 Preliminaries

In this paper, graphs are undirected, finite, loopless, and may have multiple edges. We respectively denote by $V(G)$ and $E(G)$ the vertex and edge sets of a graph G and by $|G|$ and $\|G\|$ the cardinalities of these sets (counting edges with multiplicities).

A *cut* in a graph G is a set $F \subseteq E(G)$ such that $G - F$ has more connected components than G. If $A, B \subseteq V(G)$, the cut F is an (A, B)-cut if there is no path connecting a vertex of A to a vertex of B in $G - F$. If $A, B \subseteq E(G)$, we say that F is an (A, B)-cut if there is no path from an endpoint of an edge of A to an endpoint of an edge of B in $G - F$. If (X, Y) is a partition of $V(G)$, we sometimes refer to the cut $\{xy \in E(G),\ x \in X \text{ and } y \in Y\}$ by (X, Y). For $k \in \mathbb{N}$, a graph is said to be *k-edge-connected* if it has no cut on (strictly) less than k edges.

If T is a tree and $a, b \in E(T)$, we denote by aTb the (unique) path of T from a to b and containing these edges.

Immersions. For graphs H and G, a pair (ϕ, ψ) is an *H-immersion model* in G if

- ϕ is an injection from $V(H)$ to $V(G)$,
- ψ maps every edge uv of H to a path of G between $\phi(u)$ and $\phi(v)$ such that different edges are mapped to edge-disjoint paths.

We say that H is an *immersion* of G if there is an H-immersion model in G and we denote this by $H \leq G$. This defines a partial order on graphs. We also say that H is a *proper immersion* of G if $H \leq G$ and H is not isomorphic to G.

Tree-cut decompositions. A *near-partition* of a set S is a family of pairwise disjoint subsets $S_1, \ldots, S_k \subseteq S$ such that $\bigcup_{i=1}^{k} S_i = S$. Observe that this definition allows some sets S_i of the family to be empty.

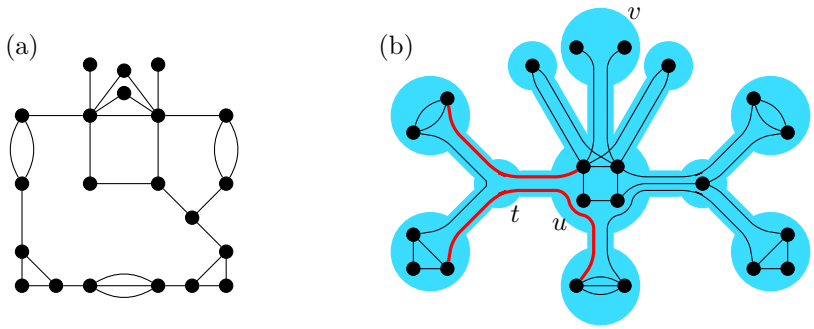

Figure 1 Example of a tree-cut decomposition (right) of a graph (left).

A *tree-cut decomposition* of a graph G is a pair $(T, \mathcal{X})$ where T is a tree and $\mathcal{X} = \{X_t\}_{t \in V(T)}$ is a near-partition of $V(G)$. We call elements of $V(T)$ *nodes* and elements of $V(G)$ *vertices* for clarity. The set X_t is called the *bag* of the decomposition corresponding to the node t, or just the *bag at* t.

An example of a tree-cut decomposition $(\mathcal{X}, T)$ of a graph G is depicted in Figure 1. In this picture, the tree T is depicted in blue and G is drawn on top of it to highlight which vertices (resp. edges) belong to which bags (resp. adhesions). This decomposition has an empty bag, at node t.

For an edge $\{u, v\} \in E(T)$, we denote by T_{uv} and T_{vu} the connected components of $T - \{uv\}$ containing u and v, respectively. The *adhesion* $\mathsf{adh}_{(T,\mathcal{X})}(uv)$ of some edge $uv \in E(T)$ is defined as the set of edges of G that have an endpoint in a bag of T_{uv} and the other one in a bag of T_{vu}. We drop the subscript when it is clear from the context. In Figure 1, the adhesion of the edge tu of T consists in the two thick red edges of G.

If G is 3-edge-connected, we define the width of $(T, \mathcal{X})$ as follows:

$$\mathtt{width}(T, \mathcal{X}) = \max \left\{ \max_{tt' \in E(T)} |\mathsf{adh}(tt')|, \max_{t \in V(T)} (|X_t| + \deg_T(t)) \right\}.$$

The *tree-cut width* of G is the maximum width over all tree-cut decompositions of G. For the simplicity of the presentation we only work on 3-edge-connected graphs in the extended abstract[4]. However, our results hold for all graphs and full proofs can be found in the full version [14]

3 A Menger-like property of tree-cut width

In this section, we give a formal definition of the notion of leanness for tree-cut decompositions and we sketch the proof of Theorem 1.

A tree-cut decomposition $(T, \mathcal{X})$ is said to be *lean* if, for every distinct edges $a, b \in E(T)$ and every $A \subseteq \mathsf{adh}(a), B \subseteq \mathsf{adh}(b)$ with $|A| = |B| =: k$,

- either there are k edge-disjoint paths linking A to B;
- or there is an edge e in aTb such that $|\mathsf{adh}(e)| < k$.

Observe that when the second point holds, $\mathsf{adh}(e)$ is an (A, B)-cut. Therefore, as with Thomas' notion of leanness mentioned in the introduction, the absence of a large collection of (here edge-disjoint) paths can be certified by a small (edge) separator that has restricted position.

Out first result (Theorem 1) shows that every graph admits a tree-cut decomposition that is lean and has optimum width. Its proof can in fact be reduced to the case of 3-edge-connected graphs.

▶ **Lemma 4** (★ (included in the proof of Theorem 1)). *Theorem 1 holds iff it holds for 3-edge-connected graphs.*

Therefore we can now focus on 3-edge-connected graphs. A crucial element of the proof of Theorem 1 is the definition of a potential on tree-cut decompositions, called *fatness*, as in Bellenbaum and Diestel's proof of Thomas leanness result for tree-decompositions [2]. Let G be a graph on m edges and let $(T, \mathcal{X})$ be a tree-cut decomposition of G. For every $i \in [m]$, we denote by $T^{\geq i}$ the subgraph of T induced by the edges that have adhesion at least i. The *fatness*, denoted by $\mathsf{fatness}(T, \mathcal{X})$, of $(T, \mathcal{X})$ is the $(2m)$-tuple $(\alpha_m, -\beta_m, \alpha_{m-1}, -\beta_{m-1}, \ldots, \alpha_1, -\beta_1)$, where for every $i \in [m]$,

[4] There is a definition the width for tree-cut decompositions of graphs that are not 3-edge-connected. It can be found in the full version [14]. The definition that we give here is equivalent to that of Wollan in [32].

- α_i is the number of edges of $T^{\geq i}$; and
- β_i is the number of connected components of $T^{\geq i}$.

We order fatnesses by lexicographic order. Informally, fatness is used to quantify the progress made when improving a non-lean tree-cut decomposition. This is captured by the following lemma which constitutes an important technical ingredient of the paper.

▶ **Lemma 5 (★).** *Let G be a 3-edge-connected graph. If $(T, \mathcal{X})$ is a tree-cut decomposition of G that is not lean, then G has a tree-cut decomposition of smaller fatness and no bigger width than the one of $(T, \mathcal{X})$.*

As the fatness of tree-decompositions of a given graph can take a finite number of values, Lemma 5 implies that we can, after a finite number of "improvement steps", obtain a lean tree-cut decomposition. Let us now describe what we mean by improvement step. Let G be

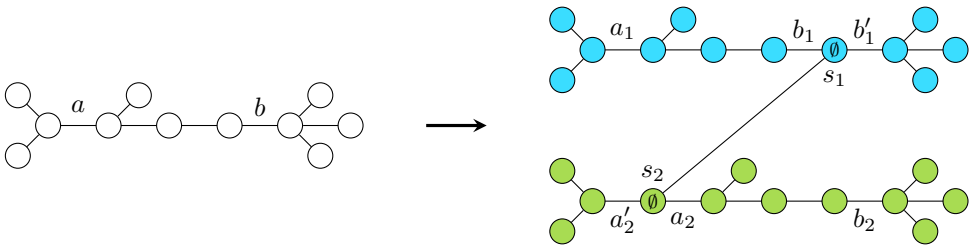

Figure 2 A tree-cut decomposition of a graph G (left) and its (a, b, F)-segregation (right), where F is a cut that separates G into G_A and G_B. The vertices of G_A and G_B respectively lie in blue and green bags. Newly introduced bags, corresponding to nodes s_1, s_2, are empty. The adhesion of $s_1 s_2$ is exactly F.

such a graph and let $(T, \mathcal{X})$ be a tree-cut decomposition of G. Let $a, b \in E(T)$, and let F be an inclusion-wise minimal cut separating a graph G into two graphs that we call G_A and G_B. We define the (a, b, F)-*segregation* of $(T, \mathcal{X})$ as the pair $(U, \mathcal{Y})$ obtained as follows:

1. consider a first copy T_1 of T, subdivide once the edge corresponding to b, call s_1 the subdivision vertex, and call the two created edges b_1 and b'_1, with the convention that (the copy of) a is closer to b_1 in T_1;
2. symmetrically, consider a second copy T_2 of T, subdivide once the edge corresponding to a, call s_2 the subdivision vertex, and call the two created edges a_2 and a'_2, with the convention that (the copy of) b is closer to a_2 in T_2;
3. in the disjoint union of these two trees, add an edge joining s_1 and s_2: this gives U;
4. for every $t \in V(U)$, let $Y_t = \begin{cases} X_t \cap V(G_A) & \text{if } t \in V(T_1) \setminus \{s_1\} \\ X_t \cap V(G_B) & \text{if } t \in V(T_2) \setminus \{s_2\}, \text{ and} \\ \emptyset & \text{if } t \in \{s_1, s_2\}. \end{cases}$

The construction of an (a, b, F)-segregation is depicted in Fig. 2. It is not hard to see that a segregation of a tree-cut decomposition is still a tree-cut decomposition.

For every non-lean tree-cut decomposition $(T, \mathcal{X})$ there are edges $a, b, e \in E(T)$ and subsets $A \subseteq \mathsf{adh}(a)$, $B \subseteq \mathsf{adh}(b)$ violating the definition of leanness. The proof of Lemma 5 consists in showing that the (a, b, F)-segregation of $(T, \mathcal{X})$, for some cut F carefully chosen using a, b, e, A, B, has the same width as $(T, \mathcal{X})$ and smaller fatness.

4 Tidy decompositions and branch interfaces

For the rest of the proofs we need the notion of *tidiness* of a tree-cut decomposition of a graph G, with respect to some edge-cut F of G. Informally, the *tidiness* property, is satisfied in a tree-cut decomposition of G, when, given a cut in a graph, the tree-cut decomposition can be *almost* partitioned (excluding a set of nodes whose size is upper-bounded linearly by the size of F) to subtrees in such a way that the vertices of the graph residing in each subtree reside also to the same side of the cut. (See also Figure 3.)

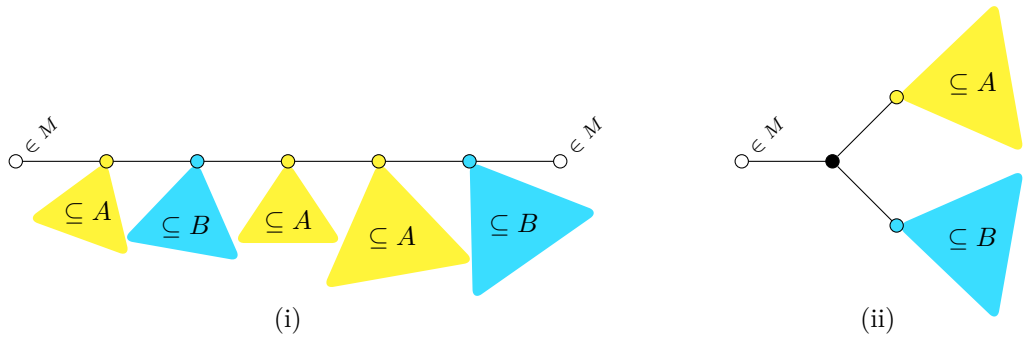

Figure 3 Parts of an (A, B)-tidy tree-cut decomposition, with subtrees colored differently depending whether the bags of their vertices are included in A or in B.

To clearly define tidy tree-cut decompositions and state the main lemma of this section we need the following.

For a tree T and a set $M \subseteq V(T)$, the *least common ancestor closure* (*lca-closure*) of M is the set $\mathsf{lca}(M) \subseteq V(T)$ obtained from M by repeatedly adding to it, for every triple x, y, z in the set, the common vertex of the paths xTy, yTz, and zTx.

▶ **Lemma 6** ([10]). *Let T be a rooted forest and $M \subseteq V(T)$. Then $|\mathsf{lca}(M)| \leq 2|M|$ and every connected component C of $T - \mathsf{lca}(M)$ has at most two neighbors in T.*

Let T be a tree, $M \subseteq V(T)$, and $v \in V(T)$. A node $u \in V(T)$ is a *M-descendant* of v if every path from u to a vertex of M contains v. We denote by $\mathsf{desc}_T^M(v)$ the set of such vertices (we drop the subscript when it is clear from the context). In particular $v \in \mathsf{desc}^M(v)$ holds for every $v \in V(T)$ and $\mathsf{desc}^M(v) = \{v\}$ holds for every $v \in M$. We also use the notation $\mathsf{desc}_T(v)$ instead of $\mathsf{desc}_T^{V(G)}(v)$.

▶ **Definition 7** (tidy tree-cut decomposition). *Let $\mathcal{D} = (T, \{X_u\}_{u \in V(T)})$ be a tree-cut decomposition of a graph G and let (A, B) be a cut of this graph. Let $M \subseteq V(T)$ be the lca-closure of the nodes of T whose bags contain endpoints of the cut (A, B). We denote by $\mathcal{C}_i^M$, $i \in [2]$ the set of connected components of $T - M$ that have exactly i neighbors in M (and drop the superscript M if it is clear from the context).*

We say that $\mathcal{D}$ is (A, B)-tidy *if the following conditions hold for every connected component F of $T - M$:*

- **(i)** *if $F \in \mathcal{C}_2^M$: for every $w \in V(F)$, all the bags at $T\,\mathsf{desc}^M(w)$ are subsets of one of A and B;*
- **(ii)** *if $F \in \mathcal{C}_1^M$: let u be the node of F adjacent to M. Then, first, u has at most two neighbors in F; let us call them v, w (if they exist). Second, X_u intersects at most one of A and B and all the bags at $T\,\mathsf{desc}^M(v)$ (resp. $T\,\mathsf{desc}^M(w)$) are subsets of A (resp. B), or the other way around.*

These two situations are again depicted on Fig. 3.

Furthermore, we show that there exists a tidy tree-cut decomposition of optimum width where the number of these subtrees is upper-bounded by a function of the size of the cut and the tree-cut width of the graph.

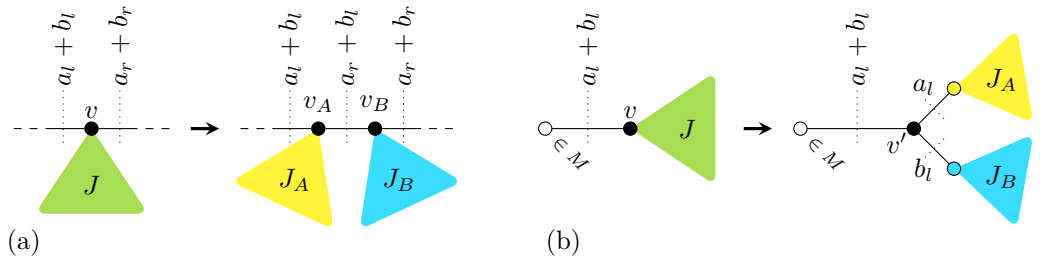

Figure 4 Improvement step towards obtaining a tidy tree-cut decomposition from a general tree-cut decomposition without increasing the width.

Before we present the lemma, we need the following definition.

Let G be a graph and let $\mathcal{D} = (T, \{X\}_{u \in V(T)})$ be a tree-cut decomposition of G. For $X \subseteq V(G)$, a *X-block* is a maximal subtree F of T such that:

- bags at nodes of F are subsets of X;
- at most two nodes of $T - V(F)$ have a neighbor in F.

If (A, B) is a cut, we refer to A- and B- blocks as *(A, B)-blocks*.

▶ **Lemma 8 (★).** *Let $\ell \in \mathbb{N}_{\geq 3}$ and G be a 3-edge-connected graph. If (A, B) is a cut of G of size ℓ, then there is an optimum tidy tree-cut decomposition $\mathcal{D} = (T, \mathcal{X})$ of G with less than $8\ell \cdot \mathbf{tcw}(G)$ (A, B)-blocks in $T - M$, where M is the lca-closure of the nodes of T whose bags contain endpoints of the cut (A, B).*

Note that from the definition of a block, two A-blocks (resp. B-blocks) cannot be connected by a link of T. This simple property is crucial in the proof of Lemma 8. It allows us, given a tree-cut decomposition of a graph G that has many (A, B)-blocks, to swap two of them (hence decreasing the total number of blocks) without increasing the width of the decomposition. Fig. 4 illustrates how we may obtain a tidy tree-cut decomposition of optimum width from a general tree-cut decomposition of optimum width while Fig. 5 illustrates the intuition behind swapping A- and B-blocks in order to decrease their total number.

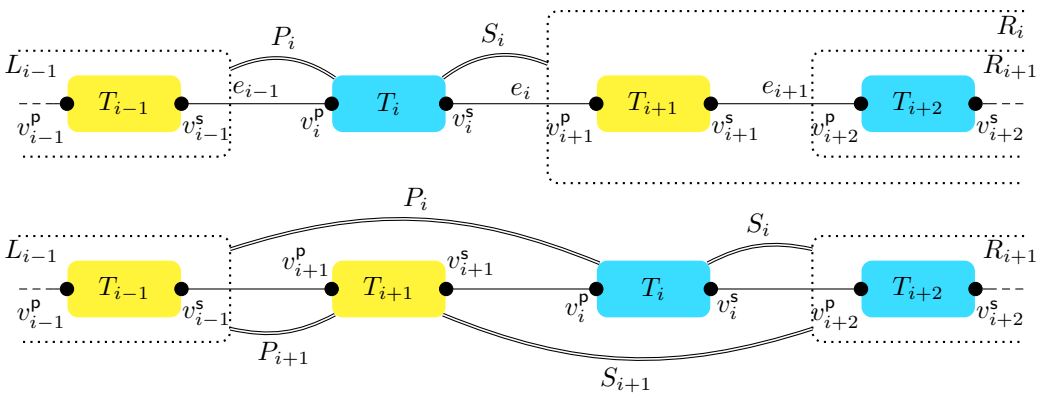

Figure 5 From $\mathcal{D}$ (top) to $\mathcal{D}'$ (bottom): swapping the blocks T_i and T_{i+1}.

5 Branch Interfaces

Tidy tree-cut decompositions will be used in this section to classify certain subgraphs of a graph into a bounded number of equivalence classes. This will be directly used to obtain our result on obstructions.

A *k-boundaried graph* is a pair $\mathbf{G} = (G, \bar{x})$ where G is a graph and $\bar{x} = (x_1, \ldots, x_k)$ is a k-tuple of the graph's vertices, called *boundary vertices* (ordered, not necessarily distinct). The *extension* of $\mathbf{G}$ is the graph G^* obtained from G by adding k new vertices $x'_1, \ldots, x'_k$ and edges $x_1x'_1, \ldots, x_kx'_k$. The *join* $\mathbf{A} \oplus \mathbf{B}$ of two k-boundaried graphs $\mathbf{A} = (A, \bar{x}), \mathbf{B} = (B, \bar{y})$ is the graph obtained from the disjoint union of A and B by adding an edge x_iy_i for $i \in [k]$. Intuitively, a boundaried graph can be seen as the subgraph corresponding to one side of an (edge) cut in a larger graph. Its boundary vertices then record the endpoints of the edges of the cut and can be glued to an other boundaried graph to recover the original graph.

We note that our definition of boundaried graph (already used in previous work [16, 15, 8, 7]) slightly differs from that usually followed in the literature (e.g. in [4]), in particular the join operation. This is because the problems that we consider are related to (edge) cuts – a setting where our definition is more natural – while other papers often deal with settings pertaining to (vertex) separators, typically graphs of bounded treewidth.

Let $\mathbf{A}$ and $\mathbf{B}$ be two k-boundaried graphs and let $G = \mathbf{A} \oplus \mathbf{B}$. (This is roughly the same as saying that G has a cut (A, B) of size k.) Let $\mathcal{D} = (T, \mathcal{X})$ be an (A, B)-tidy tree-cut decomposition of G. Let M denote the lca-closure of the vertices of T whose bags contain endpoints of the cut.

Let us construct a new pair $\mathcal{B} = (U, \mathcal{Y})$ from $\mathcal{D}$ as follows: for every (A, B)-block F of T, we contract F to a single vertex x and define Y_x as the set of vertices of G that are contained in bags at F. For every $x \in M$ we set $Y_x = X_x$. We call $\mathcal{B}$ a *bucketing* of G. In order to avoid confusion with the bags of a tree-cut decomposition, we refer to the elements of $\mathcal{Y}$ as *buckets*. Recall that by Lemma 8, the vertices of M are separated in T by a bounded number of alternations of (A, B)-blocks. Therefore the size of U is bounded by a function of k and $\mathbf{tcw}(G)$.

A *bucketing* of $\mathbf{A}$ is obtained from a bucketing of G by forgetting the content of the buckets corresponding to $\mathbf{B}$; formally it is the pair $(U, \mathcal{Y}')$ where $\mathcal{Y}' = \{Y_u \cap V(A),\ u \in V(U - M)\} \cup \{Y_u,\ u \in M\}$. Given the bucketing $(U, \mathcal{Y}')$ of $\mathbf{A}$ together with $\mathcal{D}$, we can compute:

- for each edge e of U, the size $z(e)$ of its adhesion (defined as for tree-cut decompositions);
- for each node $x \in V(T - M)$, the width $z(x)$ of the corresponding (A, B)-block in $\mathcal{D}$;[5]
- for each node $x \in V(T - M)$, the maximum $\alpha(x)$ of adhesions sizes over the subpath linking the attachement points (i.e. vertices with a neighbor outside of the block) of the corresponding (A, B)-block in $\mathcal{D}$;
- for each node $x \in M$, the sum $z(x)$ of its degree in T and bag size in $\mathcal{D}$;
- a function $f\colon [2k] \to V(U)$ specifying which buckets of U contain the endpoints of the cut.

Informally, these values encode all the relevant information about the contribution of $\mathbf{A}$ to the width of $\mathcal{D}$. If $\ell = \mathbf{tcw}(G)$, then we say that (U, z, α, f) (seen as a tuple independent of $\mathbf{A}$) is a (k, ℓ)-*branch interface*[6] and we say that $\mathbf{A}$ *conforms* with this branch interface.

[5] That is, the width of $\mathcal{D}$ restricted to this block.

[6] Note that we define here a slightly simplified notion of branch interface than in the full version [14], which is sufficient for the presentation given in this extended abstract.

Observe that the number of (k,ℓ)-branch interfaces can be upper-bounded by a function of k and ℓ. In particular we have the following.

▶ **Observation 9.** *For all $k,\ell \in \mathbb{N}$ there are $2^{O(\ell k \log k)}$ (k,ℓ)-branch interfaces.*

We denote by $\mathcal{I}_{k,\ell}(\mathbf{A})$ the set of (k,ℓ)-branch interfaces $\mathbf{A}$ conforms with. Intuitively, this set records all the possible ways for $\mathbf{A}$ to appear in a tidy tree-cut decomposition of a graph $\mathbf{A} \oplus \mathbf{B}$ of tree-cut width at most ℓ. That is, whatever k-boundaried graph $\mathbf{B}$ we consider (among those such that $\mathbf{tcw}(\mathbf{A} \oplus \mathbf{B}) \leq \ell$), all the possible ways (A,B)-blocks can interleave and contribute to the width in a tidy tree-cut decomposition of $\mathbf{A} \oplus \mathbf{B}$ is of bounded size and recorded in $\mathcal{I}_{k,\ell}(\mathbf{A})$. This is formalized by the following lemma.

▶ **Theorem 10** (edge-protrusion replacement lemma ★)**.** *Let k,ℓ be two positive integers. Let also $\mathbf{A}_1$ and $\mathbf{A}_2$ be two k-boundaried graphs with $\mathcal{I}_{k,\ell}(\mathbf{A}_1) = \mathcal{I}_{k,\ell}(\mathbf{A}_2)$. Then for any k-boundaried graph $\mathbf{B}$ where $\mathbf{tcw}(\mathbf{A}_1 \oplus \mathbf{B}) \leq \ell$ and such that both $\mathbf{A}_1 \oplus \mathbf{B}$ and $\mathbf{A}_2 \oplus \mathbf{B}$ are 3-edge connected, it holds that $\mathbf{tcw}(\mathbf{A}_2 \oplus \mathbf{B}) = \mathbf{tcw}(\mathbf{A}_1 \oplus \mathbf{B})$.*

6 Obstructions and algorithms for tree-cut width

We explain in this section how the machinery introduced in the two previous sections can be used to obtain theorems 2 and 3.

Let $\mathbf{H} = (H,\overline{x})$ and $\mathbf{G} = (G,\overline{y})$ be two k-boundaried graphs. If there is an H^*-immersion model (ϕ,ψ) of G^* where $\phi(x_i) = y_i$ and $\phi(x_i') = y_i'$ for each $i \in [k]$, then we say that $\mathbf{H}$ is a *rooted immersion* of $\mathbf{G}$ and we denote this by $\mathbf{H} \leq \mathbf{G}$.

Immersion obstructions. For every $k \in \mathbb{N}$, we denote by $\mathbf{obs}(k)$ the set of all immersion-minimal graphs whose tree-cut width is strictly greater than k. For instance, it is easy to see that $\mathbf{obs}(1)$ contains only the graph with two vertices and a double edge between them.

Our purpose is to provide a bound to the order of the graphs in $\mathbf{obs}(k)$, as a function of k. It follows from [21, Lemmata 3 and 4] that all graphs in $\mathbf{obs}(k)$ are 3-edge-connected for $k \geq 2$.

We use the following lemmata.

▶ **Proposition 11** ([16])**.** *Let w,m be positive integers and let y be a word in $[w]^*$ of length m^w. Then there is a number $k \in [w]$ and a subword u of y such that all numbers in u are at least k and u contains the number k at least m times.*

▶ **Lemma 12** (★)**.** *Let $k \in \mathbb{N}$ and let $\mathbf{H} = (H,\overline{x})$ and $\mathbf{G} = (G,\overline{y})$ be two k-boundaried graphs. For every $\ell \in \mathbb{N}$, if $\mathbf{H} \leq \mathbf{G}$, then $\mathcal{I}_{k,\ell}(\mathbf{G}) \subseteq \mathcal{I}_{k,\ell}(\mathbf{H})$.*

▶ **Lemma 13.** *There exists a function $g : \mathbb{N}^2 \to \mathbb{N}$ such that for every $w,r \in \mathbb{N}$ and every 3-edge-connected graph G where $\mathbf{tcw}(G) \leq w$ and $|V(G)| > g(r,w)$, G contains as a proper immersion a graph H for which $\mathbf{tcw}(H) \leq r$ implies $\mathbf{tcw}(G) \leq r$. Moreover such a function can be chosen so that $g(r,w) = 2^{2^{O(r \cdot w^3)}}$.*

Proof. By Lemma 5, there exists a lean tree-cut decomposition $\mathcal{D} = (T,\mathcal{X})$ of G with width at most w. We also assume that for every node t of T of degree at most 2, the bag X_t is non-empty, otherwise we revise $\mathcal{D}$ by contracting a link incident to t. The 3-edge connectivity of G implies that $\Delta(T) \leq w$. We root T on some, arbitrarily chosen, node s. This permits us to see T as a directed tree where all links are oriented towards the root s. In particular, if (v,u) is a link of T, we always assume that u is closer than v to the root.

From Observation 9 there is a function $\alpha : \mathbb{N}^2 \to \mathbb{N}$, with $\alpha(w,\ell) = 2^{O(\ell \cdot w \cdot \log w)}$, such that the number of the different (w,ℓ) interfaces of w-boundaried graphs is upper-bounded by $\alpha(w,\ell)$. We define $d = m^w$ where $m = \alpha(w,\ell)+1$ and $\ell = 8wr$. Let z be a node of T of maximum distance from the root s. Notice that z is a leaf of T. Let also $f : \mathbb{N}^2 \to \mathbb{N}$ be a function such that if $|T| > f(d,w)$, then T contains a path P of length d from z to some node, say u that is different than the root s. Notice that, as $\Delta(T) \leq w$, we can choose f such that $f(d,w) = 2^{O(d \log w)}$. We also set up the function $g : \mathbb{N}^2 \to \mathbb{N}$ where $g(r,w) = f(d,w) \cdot w$. Clearly, $g(r,w) = f(m^w,w) \cdot w = 2^{O((2^{O(\ell \cdot w^2 \cdot \log w)})^w \cdot \log w)} \cdot w = 2^{2^{O(r \cdot w^3)}}$.

Notice that if $|G| > g(r,w)$, then $|T| > f(d,w)$ and the aforementioned path P exists in T. Let $e_1, \ldots, e_d$ be the links of P ordered so that e_1 is the one that is closer to the root and let $w_{e_i} = |\mathsf{adh}_{\mathcal{D}}(e_i)|$. We now see the string $y = w_{e_1} \ldots w_{e_d}$ as a word in $[w]^*$. As $d = m^w$, from Proposition 11 there is some $k \in [w]$ and a subword $y' = w_{e_i} \ldots w_{e_j}$ of y such that $w_{e_h} \geq k$ for $h \in [i,j]$ and there is some $I \subseteq [i,j]$ such that $m = |I|$ and $\forall h \in I\ w_{e_j} = k$. Let $f_1, \ldots, f_m$ be the links in $\{e_h \mid h \in I\}$ ordered as a subsequence of $e_1, \ldots, e_d$. As $\mathcal{D}$ is lean, we know that there exist k edge-disjoint paths from $\mathsf{adh}_{(T,\mathcal{X})}(f_1)$ to $\mathsf{adh}_{(T,\mathcal{X})}(f_m)$. We denote these paths by $P_1, \ldots, P_k$. Notice also that for every $q \in [m]$, $\mathsf{adh}_{(T,\mathcal{X})}(f_m)$ contains *exactly one* edge from each of the paths in $P_1, \ldots, P_k$. We denote by $e_{q,p}$ the unique edge in $E(P_k) \cap \mathsf{adh}_{(T,\mathcal{X})}(f_q)$, for $(p,q) \in [k] \times [m]$. for every $q \in [m]$, we define $\lambda_q : \mathsf{adh}_{(T,\mathcal{X})}(f_q) \to [w_{f_q}]$ so that $\lambda_q(e_{q,p}) = p$ for $p \in [k]$.

For $q \in [m]$, we denote by t_q the tail of the edge f_q and we define $\mathbf{A}_q = (A_q, \bar{x}_q)$ where $A_q = G[\bigcup_{t \in \mathsf{desc}_T(t_q)} X_t]$ and $\bar{x}_q$ are the endpoints of the edges in $\mathsf{adh}_{(T,\mathcal{X})}(f_q)$ that belong to A_1, ordered according to λ_q (recall that $\mathsf{desc}_T(t_q)$ consists of the descendants of t_q, including t_q, in the rooted tree T). We also set $\mathbf{B}_q = (B_q, \bar{x}'_q)$ where $B_q = G - V(A_q)$ and $\bar{x}'_q$ are the endpoints of the edges in $\mathsf{adh}_{(T,\mathcal{X})}(f_q)$ that belong to B_q, again ordered according to λ_q. Notice that for every $q \in [m]$, $\mathbf{B}_q \oplus \mathbf{A}_q = G$. Moreover for every $(q,q') \in [m]^2$, where $q \leq q'$, there are k edge-disjoint paths in $\mathbf{A}_q$ that are joining, for each $p \in [k]$ the edge $e_{q,p}$ with the edge $e_{q',p}$. This implies that $\mathbf{A}_q \leq \mathbf{A}_{q'}$. This, combined with Lemma 12, yields that

$$\mathcal{I}_{k,\ell}(\mathbf{A}_1) \subseteq \cdots \subseteq \mathcal{I}_{k,\ell}(\mathbf{A}_m)$$

and, as $m = \alpha(r,w)+1$, there is a pair $(q,q') \in [m]^2$, where $q < q'$ and $\mathcal{I}_{k,\ell}(\mathbf{A}_q) = \mathcal{I}_{k,\ell}(\mathbf{A}_{q'})$. We set $H = \mathbf{A}_{q'} \oplus \mathbf{B}_q$. Since $q \neq q'$, H is a proper immersion of G. From Lemma 10, the fact that $\mathbf{tcw}(\mathbf{A}_q \oplus \mathbf{B}_q) = \mathbf{tcw}(G) \leq r$ implies that $\mathbf{tcw}(H) = \mathbf{tcw}(\mathbf{A}_q \oplus \mathbf{B}_{q'}) = \mathbf{tcw}(\mathbf{A}_q \oplus \mathbf{B}_q) = \mathbf{tcw}(G) \leq r$. ◀

▶ **Theorem 14** (Restatement of Theorem 2). *For every $k \in \mathbb{N}$, every graph in $\mathbf{obs}(k)$ has $2^{2^{O(k^4)}}$ vertices.*

Proof. Let $G \in \mathbf{obs}(k)$. This means that $\mathbf{tcw}(G) \geq k+1$ and that if H is a proper immersion of G, then $\mathbf{tcw}(H) < \mathbf{tcw}(G)$ and hence, $\mathbf{tcw}(H) \leq k$. It is easy to verify that if H is a proper immersion of G with $\|H\| = \|G\| - 1$ then $\mathbf{tcw}(H) \geq \mathbf{tcw}(G) - 1$. This last observation implies that $\mathbf{tcw}(G) \leq k+1$. If $|V(G)| > g(k,k+1)$ (where g is the function of Lemma 13) then Lemma 13 implies that G contains as a proper immersion a graph H for which $\mathbf{tcw}(H) \leq k$ implies that $\mathbf{tcw}(G) \leq k$. As $\mathbf{tcw}(H) \leq k$ for every proper immersion H of G, we deduce that $\mathbf{tcw}(G) \leq k$, a contradiction to the fact that $\mathbf{tcw}(G) \geq k+1$. The result follows as $g(k,k+1) = 2^{2^{O(k^4)}}$. ◀

Tree decompositions. A *tree decomposition* of a graph G is a pair $\mathcal{D} = (T, \mathcal{X})$, where T is a tree and $\mathcal{X} = \{X_t \mid t \in V(T)\}$ is a collection of subsets of $V(G)$ such that:

- $\bigcup_{t \in V(T)} X_t = V(G)$,
- for every $uv \in E$, there is a $t \in V(T)$ such that $\{u, v\} \subseteq X_t$, and
- for each $\{x, y, z\} \subseteq V(T)$ such that z lies on the unique path between x and y in T, $X_x \cap X_y \subseteq X_z$.

The *width* of a tree decomposition $\mathcal{D} = (T, \mathcal{X})$ is $\max_{t \in V(T)} |X_t| - 1$. The *treewidth* of a graph G, denoted by $\mathbf{tw}(G)$, is the smallest integer w such that there exists a tree decomposition of G of width at most w.

The proof of Theorem 3 comes as a direct consequence of Theorem 14.

Proof of Theorem 3. We set $n = |V(G)|$. Observe that $\Delta(G) = O_r(1)$. Moreover, if an edge has multiplicity strictly greater than r, then both its endpoints should be in the same bag of a tree-cut decomposition with width at most r. Therefore, we may replace each edge in G of multiplicity bigger than $r+1$ by one of multiplicity $r+1$ and this modification can be implemented in $O_r(n)$ steps. By applying this preprocessing we may assume that G has $O_r(n)$ edges.

In [11] it was proved that there exists a constant c_1 such that, for every graph H, $\mathbf{tw}(H) \leq c_1 \cdot (\mathbf{tcw}(H))^2$. We run the algorithm in [3] that, given a graph H and an integer q, either outputs a tree decomposition of H of width at most $5q$ or reports that $\mathbf{tw}(G) > q$, setting $H = G$ and $q = c_1 \cdot r^2$. If $\mathbf{tw}(G) > q$ we know that $\mathbf{tcw}(G) > r$ and we readily have an answer. Therefore, we can assume that we have a tree decomposition of G of width at most $5c_1 \cdot r^2$. As the property of the immersion-containment of a graph H can be expressed in Monadic Second Order Logic, by using Courcelle's theorem, it is possible to construct an algorithm that, given two graphs H_1 and H_2, outputs whether $H_1 \leq H_2$ in $O_{|V(H_1)|+\mathbf{tw}(G)}(|V(H_2)|)$ steps. By Theorem 14, the set $\mathbf{obs}(r)$ can be constructed in $O_r(1)$ steps. Then, by testing immersion-containment for every graph $H \in \mathbf{obs}(r)$, we can decide whether $\mathbf{tcw}(G) \leq r$ in $O_r(n)$ steps. ◀

7 Discussion

An interesting direction is to specify the parameterized dependence of the algorithm in Theorem 3. We argue below on why the proof of Lemma 13 can already give a first bound.

Notice that Lemma 10 defines an equivalence relation of w-boundaried graphs. We say that two k-boundaried graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ are (w, r)-*equivalent*, namely $\mathbf{G}_1 \equiv_{w,r} \mathbf{G}_2$ if $\mathcal{I}_{w,8wr}(\mathbf{G}_1) = \mathcal{I}_{w,8wr}(\mathbf{G}_2)$. Notice that the proof of Lemma 13 already implies that a minimum-size of a representative of any of the equivalence classes of $\equiv_{w,r}$ has $g(r, w) = 2^{2^{O(r \cdot w^3)}}$ vertices. By brute-force checking on all graphs on $g(r, w) = 2^{2^{O(r \cdot w^3)}}$ vertices, we may construct a set $\mathcal{R}_{w,r}$ of representatives for $\equiv$. Moreover $|\mathcal{R}_{w,r}| = 2^{2^{2^{O(r \cdot w^3)}}}$. The knowledge of $\mathcal{R}_{w,r}$ permits us to avoid the use of Courcelle's theorem in Theorem 3. For this, we first transform the tree decomposition of the proof of Theorem 3 to a tree-cut decomposition with width $w = O(r^4)$ using [32, Lemma 12]. Then we implement an algorithmic version of the proof of Lemma 13: as long as the sub-tree of T rooted on u of height $2^{2^{r \cdot w^3}}$ exists, we reduce $G = \mathbf{A}_q \oplus \mathbf{B}_q$ to the smaller equivalent instance $G = \mathbf{A}_{q'} \oplus \mathbf{B}_q$. Given that $|\mathcal{R}_{w,r}| = 2^{2^{2^{O(r \cdot w^3)}}}$ and using suitable data-structures, this compression can be implemented in $2^{2^{2^{O(r^{13})}}}$ steps and as it is repeated at most n times, it will report a correct answer in $2^{2^{2^{O(r^{13})}}} \cdot n$ steps.

Clearly, the parameterized dependence of the above argumentation is still too heavy. We believe that a detailed dynamic programming based on how collections of branch interfaces are updated along a rooted tree-cut decomposition may yield a bound $f(r) = 2^{\text{poly}(r)}$ to the function of Theorem 3.

A second direction for further research is to obtain an algorithm that, besides computing the tree-cut width of a graph, also provides a tree-cut decomposition of optimal width.

References

1 Jeffrey Azzato. Linked tree-decompositions of represented infinite matroids. *Journal of Combinatorial Theory, Series B*, 101(3):123–140, 2011. doi:10.1016/j.jctb.2010.12.003.

2 Patrick Bellenbaum and Reinhard Diestel. Two short proofs concerning tree-decompositions. *Combinatorics, Probability and Computing*, 11(6):541–547, 2002.

3 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A $(c^k n)$ 5-Approximation Algorithm for Treewidth. *SIAM Journal of Computing*, 45(2):317–378, 2016.

4 Hans L Bodlaender, Fedor V Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M Thilikos. (Meta) kernelization. *Journal of the ACM*, 63(5):44, 2016.

5 Heather Booth, Rajeev Govindan, Michael A. Langston, and Siddharthan Ramachandramurthi. Cutwidth approximation in linear time. In *Proceedings of the Second Great Lakes Symposium on VLSI*, pages 70–73. IEEE, 1992.

6 Johannes Carmesin, Reinhard Diestel, M. Hamann, and Fabian Hundertmark. k-Blocks: A Connectivity Invariant for Graphs. *SIAM Journal on Discrete Mathematics*, 28(4):1876–1891, 2014.

7 Dimitris Chatzidimitriou, Jean-Florent Raymond, Ignasi Sau, and Dimitrios M. Thilikos. An $O(\log OPT)$-approximation for covering and packing minor models of θ_r. *Algorithmica*, April 2017. doi:10.1007/s00453-017-0313-5.

8 Dimitris Chatzidimitriou, Jean-Florent Raymond, Ignasi Sau, and Dimitrios M Thilikos. Minors in graphs of large θ_r-girth. *European Journal of Combinatorics*, 65:106–121, 2017.

9 Joshua Erde. A unified treatment of linked and lean tree-decompositions. *Journal of Combinatorial Theory, Series B*, 130:114–143, 2018.

10 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *Proceedings of FOCS 2012*, pages 470–479. IEEE Computer Society, 2012.

11 Robert Ganian, Eun Jung Kim, and Stefan Szeider. Algorithmic applications of tree-cut width. In *Mathematical foundations of computer science 2015. Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 348–360. Springer, Heidelberg, 2015.

12 James F. Geelen, A. M. H. Gerards, and Geoff Whittle. Branch-width and well-quasi-ordering in matroids and graphs. *Journal of Combinatorial Theory, Series B*, 84(2):270–290, 2002.

13 Jim Geelen and Benson Joeris. A generalization of the Grid Theorem. *arXiv preprint*, 2016. arXiv:1609.09098.

14 Archontia C. Giannopoulou, O-joung Kwon, Jean-Florent Raymond, and Dimitrios M. Thilikos. Lean tree-cut decompositions: obstructions and algorithms. *arXiv e-prints*, August 2018. arXiv:1808.00863.

15 Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Linear Kernels for Edge Deletion Problems to Immersion-Closed Graph Classes. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 57:1–57:15, 2017.

16 Archontia C. Giannopoulou, Michał Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: Obstructions and Algorithmic Aspects. *Algorithmica*, March 2018. doi:10.1007/s00453-018-0424-7.

17 Rajeev Govindan and Siddharthan Ramachandramurthi. A weak immersion relation on graphs and its applications. *Discrete Mathematics*, 230(1–3):189–206, 2001.

18 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding Topological Subgraphs is Fixed-parameter Tractable. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 479–488, New York, NY, USA, 2011. ACM.

19 Tibor Grünwald. Ein Neuer Beweis Eines Mengerschen Satzes. *Journal of the London Mathematical Society*, s1-13(3):188–192, 1938. doi:10.1112/jlms/s1-13.3.188.

20 Mamadou Moustapha Kanté and O-joung Kwon. An Upper Bound on the Size of Obstructions for Bounded Linear Rank-Width. *CoRR*, abs/1412.6201, 2014. arXiv:1412.6201.

21 Eun Jung Kim, Sang-il Oum, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. An FPT 2-Approximation for Tree-Cut Decomposition. *Algorithmica*, 80(1):116–135, 2018.

22 Ilhee Kim and Paul D. Seymour. Tournament minors. *Journal of Combinatorial Theory, Series B*, 112:138–153, 2015.

23 Shiva Kintali. Directed Minors III. Directed Linked Decompositions. *CoRR*, 2014. arXiv:1404.5976.

24 Jens Lagergren. Upper Bounds on the Size of Obstructions and Intertwines. *Journal of Combinatorial Theory, Series B*, 73(1):7–40, 1998.

25 Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 1(10):96–115, 1927.

26 Sang-il Oum. Rank-width and vertex-minors. *Journal of Combinatorial Theory, Series B*, 95(1):79–100, 2005.

27 Neil Robertson and Paul Seymour. Graph Minors. XXIII. Nash-Williams' Immersion Conjecture. *Journal of Combinatorial Theory, Series B*, 100(2):181–205, March 2010.

28 Neil Robertson and Paul D. Seymour. Graph Minors. II. Algorithmic Aspects of Tree-Width. *Journal of Algorithms*, 7(3):309–322, 1986.

29 Neil Robertson and Paul D. Seymour. Graph Minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41(2):92–114, 1986.

30 Neil Robertson and Paul D Seymour. Graph Minors. XIII. The disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.

31 Robin Thomas. A menger-like property of tree-width: The finite case. *Journal of Combinatorial Theory, Series B*, 48(1):67–76, 1990.

32 Paul Wollan. The structure of graphs not admitting a fixed immersion. *Journal of Combinatorial Theory, Series B*, 110:47–66, 2015.