Cutwidth: Obstructions and Algorithmic Aspects*†

Archontia C. Giannopoulou[1], **Michał Pilipczuk**[2], **Jean-Florent Raymond**[3], **Dimitrios M. Thilikos**[4], and **Marcin Wrochna**[5]

1 **Technische Universität Berlin, Berlin, Germany**
`archontia.giannopoulou@tu-berlin.de`

2 **Institute of Informatics, University of Warsaw, Poland**
`michal.pilipczuk@mimuw.edu.pl`

3 **Institute of Informatics, University of Warsaw, Poland; and AlGCo project team, CNRS, LIRMM, Montpellier, France**
`jean-florent.raymond@mimuw.edu.pl`

4 **AlGCo project team, CNRS, LIRMM, Montpellier, France; and Department of Mathematics, National and Kapodistrian University of Athens, Greece**
`sedthilk@thilikos.info`

5 **Institute of Informatics, University of Warsaw, Poland**
`m.wrochna@mimuw.edu.pl`

Abstract

Cutwidth is one of the classic layout parameters for graphs. It measures how well one can order the vertices of a graph in a linear manner, so that the maximum number of edges between any prefix and its complement suffix is minimized. As graphs of cutwidth at most k are closed under taking immersions, the results of Robertson and Seymour imply that there is a finite list of minimal immersion obstructions for admitting a cut layout of width at most k. We prove that every minimal immersion obstruction for cutwidth at most k has size at most $2^{O(k^3 \log k)}$.

As an interesting algorithmic byproduct, we design a new fixed-parameter algorithm for computing the cutwidth of a graph that runs in time $2^{O(k^2 \log k)} \cdot n$, where k is the optimum width and n is the number of vertices. While being slower by a $\log k$-factor in the exponent than the fastest known algorithm, due to Thilikos, Bodlaender, and Serna [17, 18], our algorithm has the advantage of being simpler and self-contained; arguably, it explains better the combinatorics of optimum-width layouts.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases cutwidth, obstructions, immersions, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.IPEC.2016.15

* A full version of the paper is available at `http://arxiv.org/abs/1606.05975`.

† This work was partially done while A. C. Giannopoulou was holding a post-doc position at Warsaw Center of Mathematics and Computer Science. The research of A. C. Giannopoulou has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (ERC consolidator grant DISTRUCT, agreement No 648527). The research of Mi. Pilipczuk and M. Wrochna is supported by the Polish National Science Center grant SONATA UMO-2013/11/D/ST6/03073. The research of J.-F. Raymond is supported by the Polish National Science Center grant PRELUDIUM UMO-2013/11/N/ST6/02706. Mi. Pilipczuk is supported by the Foundation for Polish Science (FNP) via the START stipend programme.

1 Introduction

The cutwidth of a graph is defined as the minimum possible *width* of a linear ordering of its vertices, where the width of an ordering σ is the maximum, among all the prefixes of σ, of the number of edges that have exactly one vertex in a prefix. Due to its natural definition, cutwidth has various applications in a range of practical fields of computer science: whenever data is expected to be roughly linearly ordered and dependencies or connections are local, one can expect the cutwidth of the corresponding graph to be small. These applications include circuit design, graph drawing, bioinformatics, and text information retrieval; we refer to the survey of layout parameters of Díaz, Petit, and Serna [5] for a broader discussion.

As finding a layout of optimum width is NP-hard [7], the algorithmic and combinatorial aspects of cutwidth were intensively studied. There is a broad range of polynomial-time algorithms for special graph classes [10, 11, 21], approximation algorithms [14], and fixed-parameter algorithms [17, 18]. In particular, Thilikos, Bodlaender, and Serna [17, 18] proposed a fixed-parameter algorithm for computing the cutwidth of a graph that runs[1] in time $2^{O(k^2)} \cdot n$, where k is the optimum width and n is the number of vertices. Their approach is to first compute the pathwidth of the input graph, which is never larger than the cutwidth. Then, the optimum layout can be constructed by an elaborate dynamic programming procedure on the obtained path decomposition. To upper bound the number of relevant states, the authors had to understand how an optimum layout can look in a given path decomposition. For this, they borrow the technique of *typical sequences* of Bodlaender and Kloks [3], which was introduced for a similar reason, but for pathwidth and treewidth instead of cutwidth.

Since the class of graphs of cutwidth at most k is closed under immersions, and the immersion order is a well-quasi ordering of graphs[2] [15], it follows that for each k there exists a *finite* obstruction set $\mathcal{L}_k$ of graphs such that a graph has cutwidth at most k if and only if it does not admit any graph from $\mathcal{L}_k$ as an immersion. However, this existential result does not give any hint on how to generate, or at least estimate the sizes of the obstructions. The sizes of obstructions are important for efficient treatment of graphs of small cutwidth; this applies also in practice, as indicated by Booth et al. [4] in the context of VLSI design.

The estimation of sizes of minimal obstructions for several graph parameters has been studied before. For minor-closed parameters pathwidth and treewidth, Lagergren [13] showed that any minimal minor obstruction to admitting a path decomposition of width k has size at most single-exponential in $O(k^4)$, whereas for treewidth he showed an upper bound double-exponential in $O(k^5)$. Less is known about immersion-closed parameters, like cutwidth. Govindan and Ramachandramurthi [9] showed that the number of minimal immersion obstructions for the class of graphs of cutwidth at most k is at least $3^{k-7}+1$. The construction in [9] exemplifies minimal obstructions for cutwidth at most k with $(3^{k-5}-1)/2$ vertices. To the best of our knowledge, nothing was known about upper bounds for the cutwidth case.

Results on obstructions. Our main result concerns the sizes of obstructions for cutwidth.

▶ **Theorem 1.** *Suppose a graph G has cutwidth larger than k, but every graph with fewer vertices or edges (strongly) immersed in G has cutwidth at most k. Then G has at most $2^{O(k^3 \log k)}$ vertices and edges.*

[1] Thilikos, Bodlaender, and Serna [17, 18] do not specify the parametric dependence of the running time of their algorithm. A careful analysis of their algorithm yields the above claimed running time bound.
[2] All graphs considered in this paper may have parallel edges, but no loops.

The above result immediately gives the same upper bound on the sizes of graphs from the minimal obstruction sets $\mathcal{L}_k$ as they satisfy the prerequisites of Theorem 1. This somewhat matches the $(3^{k-5}-1)/2$ lower bound of Govindan and Ramachandramurthi [9].

Our approach for Theorem 1 follows the technique used by Lagergren [13] to prove that minimal minor obstructions for pathwidth at most k have sizes single-exponential in $O(k^4)$. Intuitively, the idea of Lagergren is to take an optimum decomposition for a minimal obstruction, which must have width $k+1$, and to assign to each prefix of the decomposition one of finitely many "types", so that two prefixes with the same type "behave" in the same manner. If there were two prefixes, one being shorter than the other, with the same type, then one could replace one with the other, thus obtaining a smaller obstruction. Hence, the upper bound on the number of types, being double-exponential in $O(k^4)$, gives some upper bound on the size of a minimal obstruction. This upper bound can be further improved to single-exponential by observing that types are ordered by a natural domination relation, and the shorter a prefix is, the weaker is its type. An important detail is that one needs to make sure that the replacement can be modeled by minor operations. For this, Lagergren uses the notion of *linked path decompositions* (a weaker variant of *lean path decompositions*; cf. [19, 1]).

To prove Theorem 1, we perform a similar analysis of prefixes of an optimum ordering of a minimal obstruction. We show that prefixes can be categorized into a bounded number of types, depending on their "behavior". Provided two prefixes with equally strong type appear one after the other, we can "unpump" the part of the graph in their difference.

To make sure that unpumping is modeled by taking an immersion, we define *linked orderings* for cutwidth and reprove the analogue of the result of Thomas [19] (see [1] for simplified proofs): there is always an optimum-width ordering that is linked. We remark this already follows from more general results on submodular functions: the same is true for parameters like *linear rank-width*, as observed by Kanté and Kwon [12], which in turns follows from the proof of an analogous theorem of Geelen et al. [8] that applies to branch-decompositions, and thus, e.g., to parameters known as *branch-width* and *carving-width*.

The proof of the upper bound on the number of types essentially boils down to the following setting. We are given a graph G and a subset X of vertices, such that at most ℓ edges have exactly one endpoint in X. The question is how X can look like in an optimum-width ordering of G. We prove that there is always an ordering where X is split into at most $O(k\ell)$ blocks, where k is the optimum width. This allows us to store the relevant information on the whole X in one of a constant number of types (called *bucket interfaces*). The swapping argument used in this proof holds the essence of the typical sequences technique of Bodlaender and Kloks [3], while being, in our opinion, more natural and easier to understand.

As an interesting byproduct, we can also use our understanding to treat the problem of removing edges to get a graph of small cutwidth. More precisely, for parameters w, k, we consider the class of all graphs G, such that w edges can be removed from G to obtain a graph of cutwidth at most k. We prove that for every constant k, the minimal (strong) immersion obstructions for this class have sizes bounded *linearly* in w. Moreover we give an exponential lower bound to the number of these obstructions. Due to the auxiliary character of these results, we defer the precise statement and discussion to the full version of this paper.

Algorithmic results. Consider the following "compression" problem: given a graph G and its ordering σ of width ℓ, we would like to construct, if possible, a new ordering of the vertices of G of width at most k, where $k < \ell$. Then the types defined above essentially match states that would be associated with prefixes of σ in a dynamic programming algorithm solving this problem. Alternatively, one can think of building an automaton that traverses the ordering σ of width ℓ while constructing an ordering of G of width at most k. Hence, our upper bound on the number of types can be directly used to limit the state space in such a dynamic programming procedure/automaton, yielding an FPT algorithm for the above problem.

With this result in hand, it is not hard to design of an exact FPT algorithm for cutwidth. One could introduce vertices one by one to the graph, while maintaining an ordering of optimum width. Each time a new vertex is introduced, we put it anywhere into the ordering, and it can be argued that the new ordering has width at most three times larger than the optimum. Then, the dynamic programming algorithm sketched above can be used to "compress" this approximate ordering to an optimum one in linear FPT time.

The above approach yields a quadratic algorithm. To match the optimum, linear running time, we use a similar trick as Bodlaender in his linear-time algorithm for computing the treewidth of the graph [2]. Namely, we show that instead of processing vertices one by one, we can proceed recursively by removing a significant fraction of all the edges at each step, so that their reintroduction increases the width by a factor of at most two. We then run the compression algorithm on the obtained 2-approximate ordering to get an optimum one. Since we remove a large portion of the graph at each step, the recursive equation on the running time solves to a linear function, instead of quadratic. This gives the following.

▶ **Theorem 2.** *There exists an algorithm that, given an n-vertex graph G and an integer k, runs in time $2^{O(k^2 \log k)} \cdot n$ and either correctly concludes that the cutwidth of G is larger than k, or outputs an ordering of G of width at most k.*

The algorithm of Theorem 2 has running time slightly larger than that of Thilikos, Bodlaender, and Serna [17, 18]. The difference is the $\log k$ factor in the exponent, the reason for which is that we use a simpler bucketing approach to bound the number of states, instead of the more entangled, but finer, machinery of typical sequences. We believe the main strength of our approach lies in its explanatory character. Instead of relying on algorithms for computing tree or path decompositions, which are already difficult by themselves, and then designing a dynamic programming algorithm on a path decomposition, we directly approach cutwidth "via cutwidth", and not "via pathwidth". That is, the dynamic programming procedure for computing the optimum cutwidth ordering on an approximate cutwidth ordering is technically far simpler and conceptually more insightful than performing the same on a general path decomposition. We also show that the reduction-by-a-large-fraction trick of Bodlaender [2] can be performed also in the cutwidth setting, yielding a self-contained, natural, and understandable algorithm.

2 Preliminaries

We denote the set of non-negative integers by $\mathbb{N}$ and the set of positive integers by $\mathbb{N}^+$. For $r, s \in \mathbb{N}$ with $r \leq s$, we denote $[r] = \{1, \ldots, r\}$ and $[r, s] = \{r, \ldots, s\}$. Notice that $[0] = \emptyset$.

Graphs. All graphs considered in this paper are undirected, without loops, and may have multiple edges. The vertex and edge sets of a graph G are denoted by $V(G)$ and $E(G)$, respectively. For disjoint $X, Y \subseteq V(G)$, by $E_G(X, Y)$ we denote the set of edges of G with one endpoint in X and one in Y. If $S \subseteq V(G)$, then we denote $\delta_G(S) = |E_G(S, V(G) \setminus S)|$. We drop the subscript if it is clear from the context. Every partition (A, B) of $V(G)$ is called a *cut of* G; the *size* of the cut (A, B) is $\delta(A)$.

Cutwidth. Let G be a graph and σ be an ordering of $V(G)$. For $u, v \in V(G)$, we write $u <_\sigma v$ if u appears before v in σ. Given two disjoint sequences $\sigma_1 = \langle x_1, \ldots, x_{r_1} \rangle$ and $\sigma_2 = \langle y_1, \ldots, y_{r_2} \rangle$ of vertices in $V(G)$, we define their *concatenation* as $\sigma_1 \circ \sigma_2 = \langle x_1, \ldots, x_{r_1}, y_1, \ldots, y_{r_2} \rangle$. For $X \subseteq V(G)$, let σ_X be the ordering of X induced by σ, i.e., the ordering obtained from σ if we remove the vertices that do not belong in X. For a vertex v we denote by V_v^σ the set $\{u \in V(G) \mid u \leq_\sigma v\}$. A *$\sigma$-cut* is any cut of the form $(V_v^\sigma, V(G) \setminus V_v^\sigma)$ for $v \in V(G)$. The *cutwidth of an ordering σ of G* is defined as $\mathbf{cw}_\sigma(G) = \max_{v \in V(G)} \delta(V_v^\sigma)$. The *cutwidth of G*, $\mathbf{cw}(G)$, is the minimum of $\mathbf{cw}_\sigma(G)$ over all possible orderings of $V(G)$.

Obstructions. Let $\leq$ be a partial order on graphs. We say that $G' \lneq G$ if $G' \leq G$ and G' is not isomorphic to G. A graph class $\mathcal{G}$ is *closed under* $\leq$ if whenever $G' \leq G$ and $G \in \mathcal{G}$, we also have that $G' \in \mathcal{G}$. Given a partial order $\leq$ and a graph class $\mathcal{G}$ closed under $\leq$, we define the *(minimal) obstruction set* of $\mathcal{G}$ w.r.t. $\leq$, denoted by $\mathbf{obs}_\leq(\mathcal{G})$, as the set containing all graphs where the following two conditions hold: **O1**: $G \notin \mathcal{G}$, i.e., G is not a member of $\mathcal{G}$, and **O2**: for each G' with $G' \lneq G$, we have that $G' \in \mathcal{G}$.

We say that a set of graphs $\mathcal{H}$ is a *$\leq$-antichain* if it does not contain any pair of comparable elements wrt. $\leq$. By definition, for any class $\mathcal{G}$ closed under $\leq$, the set $\mathbf{obs}_\leq(\mathcal{G})$ is an antichain.

Immersions. Let H and G be graphs. We say that G contains H as an *immersion* if there is a pair of functions (ϕ, ψ), called an *H-immersion model of G*, such that ϕ is an injection from $V(H)$ to $V(G)$ and ψ maps every edge uv of H to a path of G between $\phi(u)$ and $\phi(v)$ so that different edges are mapped to edge-disjoint paths. Every vertex in the image of ϕ is called a *branch vertex*. If we additionally demand that no internal vertex of a path in $\psi(E(H))$ is a branch vertex, then we say that (ϕ, ψ) is a *strong H-immersion model* and H is a *strong immersion* of G. We denote by $H \leq_{\mathrm{i}} G$ ($H \leq_{\mathrm{si}} G$) the fact that H is an immersion (strong immersion) of G; these are partial orders. Clearly, for any two graphs H and G, if $H \leq_{\mathrm{si}} G$ then $H \leq_{\mathrm{i}} G$. This implies the following observation:

▶ **Observation 3.** *If $\mathcal{G}$ is a graph class closed under $\leq_{\mathrm{i}}$, then $\mathbf{obs}_{\leq_{\mathrm{i}}}(\mathcal{G}) \subseteq \mathbf{obs}_{\leq_{\mathrm{si}}}(\mathcal{G})$.*

Robertson and Seymour proved in [15] that every $\leq_i$-antichain is finite and conjectured the same for $\leq_{\mathrm{si}}$. It is well-known that for every $k \in \mathbb{N}$, the class $\mathcal{C}_k$ of graphs of cutwidth at most k is closed under immersions. It follows from the results of [15] that $\mathbf{obs}_{\leq_{\mathrm{i}}}(\mathcal{C}_k)$ is finite; the goal of this paper is to provide good estimates on the sizes of graphs in $\mathbf{obs}_{\leq_{\mathrm{si}}}(\mathcal{C}_k)$. As the cutwidth of a graphs is the maximum cutwidth of its connected components, it follows that graphs in $\mathbf{obs}_{\leq_{\mathrm{si}}}(\mathcal{C}_k)$ are connected. Moreover, every graph in $\mathbf{obs}_{\leq_{\mathrm{si}}}(\mathcal{C}_k)$ has cutwidth exactly $k+1$, because the removal of any of its edges decreases its cutwidth to at most k.

3 Bucket interfaces

Let G be a graph and σ be an ordering of $V(G)$. For a set $X \subseteq V(G)$, the *X-blocks* in σ are the maximal subsequences of consecutive vertices of σ that belong to X. Suppose (A, B) is a cut of G. Then we can write $\sigma = b_1 \circ \ldots \circ b_p$, where $b_1, \ldots, b_p$ are the A- and B-blocks in σ; these will be called jointly *(A, B)-blocks*. The next lemma is the cornerstone of our approach: we prove that given a graph G and a cut (A, B) of G, there exists an optimum cutwidth ordering of G where number of blocks depends only on the cutwidth and the size of (A, B).

▶ **Lemma 4.** *Let $\ell \in \mathbb{N}^+$ and G be a graph. If (A, B) is a cut of G of size ℓ, then there is an optimum cutwidth ordering σ of $V(G)$ with at most $(2\ell+1) \cdot (2\mathbf{cw}(G)+3) + 2\ell$ (A, B)-blocks.*

Proof. Let σ be an optimum cutwidth ordering such that, subject to the width being minimum, the number of (A,B)-blocks it defines is also minimized. Let $\sigma = b_1 \circ b_2 \circ \cdots \circ b_r$, where $b_1, b_2, \ldots, b_r$ are the (A,B)-blocks of σ. If σ defines less than three blocks, then the claim already follows, so let us assume $r \geq 3$.

Consider any ordering σ' obtained by swapping two blocks, i.e., $\sigma' = b_1 \circ \cdots \circ b_{j-1} \circ b_{j+1} \circ b_j \circ b_{j+2} \ldots b_r$, for some $j \in [r-1]$. Observe that since the blocks $b_1, \ldots, b_r$ alternate as A-blocks and B-blocks, the ordering σ' has a strictly smaller number of blocks; indeed, either $j-1 \geq 1$, in which case $b_{j-1} \circ b_{j+1}$ defines a single block of σ', or $j = 1$ and hence $j+2 \leq r$, in which case $b_j \circ b_{j+2}$ does. Therefore, by choice of σ, for each $j \in [r-1]$, swapping b_j and b_{j+1} in σ must yield an ordering with strictly larger cutwidth.

We call a block *free* if it does not contain any endpoint of the cut edges $E_G(A,B)$. We now prove that any sequence of consecutive free blocks in σ has at most $2\mathbf{cw}(G)+3$ blocks. Since the cut (A,B) has size ℓ, there are at most 2ℓ blocks that are not free. This implies the claimed bound on the total number of all blocks in σ.

Suppose, to the contrary, that there exists a sequence of $q > 2\mathbf{cw}(G)+3$ consecutive free blocks in σ. Let these blocks be $b_r, b_{r+1}, \ldots, b_s$, where $s - r + 1 = q$. For $j \in [r, s-1]$, we define $\mu(j)$ to be the size of the cut between all vertices inside or preceding the vertices of block b_j and all vertices inside or following the vertices of block b_{j+1} in σ; see Figure 1.

▶ **Claim 5.** *For all $j \in [r+1, \ldots, s-2]$, we have that $\mu(j-1) > \mu(j)$ or $\mu(j) < \mu(j+1)$.*

Proof. Suppose that for some $j \in [r+1, s-2]$, $\mu(j) \geq \max(\mu(j-1), \mu(j+1))$. We will then show that the ordering σ' obtained by swapping the blocks b_j and b_{j+1} still has optimum cutwidth, a contradiction to the choice of σ. Notice that for every vertex v preceding all vertices of b_j or succeeding all vertices of b_{j+1}, $\delta(V_v^{\sigma'}) = \delta(V_v^{\sigma})$. Thus, it remains to show that for any vertex v belonging to the block b_j or to the block b_{j+1}, also $\delta(V_v^{\sigma'}) \leq \delta(V_v^{\sigma})$.

Let p_j be the number of edges of G with one endpoint in the block b_j and the other endpoint preceding (in σ) all vertices of b_j. Let also s_j be the number of edges of G with one endpoint in b_j and the other endpoint succeeding (in σ) all vertices of b_j (and hence succeeding all vertices of block b_{j+1}, since both b_j and b_{j+1} are free). Notice that $\mu(j) = \mu(j-1) - p_j + s_j$ and recall that $\mu(j) \geq \mu(j-1)$. This yields that $s_j \geq p_j$.

Similarly, let p_{j+1} be the number of edges of G with one endpoint in b_{j+1} and the other endpoint preceding all vertices of the block b_{j+1} (and, in particular, all vertices of block b_j). Let also s_{j+1} be the number of edges of G with one endpoint in b_{j+1} and the other endpoint succeeding all vertices of block b_{j+1}. Again, we have $\mu(j+1) = \mu(j) - p_{j+1} + s_{j+1}$ and $\mu(j) \geq \mu(j+1)$. This yields that $p_{j+1} \geq s_{j+1}$.

Let v be a vertex of the block b_j. Recall that the blocks b_j and b_{j+1} are free and thus, there are no edges between them. Observe then that $\delta(V_v^{\sigma'}) = \delta(V_v^{\sigma}) + s_{j+1} - p_{j+1} \leq \delta(V_v^{\sigma})$. Symmetrically, for any vertex v in b_{j+1}, observe that $\delta(V_v^{\sigma'}) = \delta(V_v^{\sigma}) + p_j - s_j \leq \delta(V_v^{\sigma})$. Thus, $\mathbf{cw}_{\sigma'}(G) \leq \mathbf{cw}_{\sigma}(G) = \mathbf{cw}(G)$, a contradiction. ◀

Claim 5 shows that for all $j \in [r+1, s-2]$, we have $\mu(j-1) > \mu(j)$ or $\mu(j) < \mu(j+1)$. It follows that any non-decreasing pair $\mu(j-1) \leq \mu(j)$ must be followed by an increasing pair $\mu(j) < \mu(j+1)$. Hence, if $j_{\min}$ is the minimum index such that $\mu(j_{\min}) \leq \mu(j_{\min}+1)$, then the sequence $\mu(j)$ has to be strictly decreasing up to $j_{\min}$ and strictly increasing from $j_{\min}+1$ onward. Since $\mu(j) \leq \mathbf{cw}(G)$ for all j, the length q of the sequence of consecutive free blocks cannot be longer than $2\mathbf{cw}(G)+3$ in total, concluding the proof. ◀

We use the above lemma to bound the number of "types" of prefixes in graph orderings. To describe such a prefix, i.e., one side of a cut in a graph, we use the following definition.

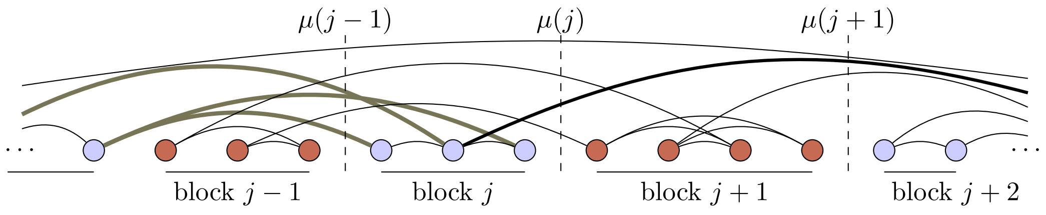

Figure 1 A cut (A, B) is highlighted (blue, red), with the corresponding blocks underlined and cuts between them marked with dashed lines. Edges counted as p_j and s_j are thickened.

▶ **Definition 6.** 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 *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 $i \in [k]$.

From Lemma 4 we derive that for any given cut (A, B) of size ℓ of a graph G with $\mathbf{cw}(G) \leq k$, there is an optimum cutwidth ordering in which the vertices of A occur in $O(k\ell)$ blocks. Our next goal is to show that the only information about A that can affect the cutwidth of G is: the placing of the endpoints of each cutedge (x_i and x'_i) into blocks, and the cutwidth of each block (as an induced subgraph of A or A^*). Recall that for an ordering σ of $V(G)$, *σ-cuts* are cuts of the form $(V^\sigma_v, V(G) \setminus V^\sigma_v)$, for $v \in V(G)$.

▶ **Definition 7.** Let G be a graph and σ be an ordering of its vertices. An *ℓ-bucketing* of σ is a function $T\colon V(G) \to [\ell]$ such that $T(u) \leq T(v)$ for any u appearing before v in σ. For every $i \in [\ell]$, the set $T^{-1}(i)$ will be called a *bucket*; a bucket is naturally ordered by σ. For every bucket $T^{-1}(i)$, $i \in [\ell]$, let $\mathtt{cuts}(G, \sigma, T, i)$ be the family of σ-cuts containing on one side all vertices of buckets appearing before i and a prefix (in σ) of the i-th bucket. For an ordering σ of the vertices of a graph G, define the *width* of the bucket i, $i \in [\ell]$, as the maximum width of any cut in the family $\mathtt{cuts}(G, \sigma, T, i)$. Formally,

$$\begin{aligned}
\mathtt{cuts}(G,\sigma,T,i) &= \big\{ \big(T^{-1}([1,i-1]) \,\cup\, L, \quad R \,\cup\, T^{-1}([i+1,\ell])\big) : \\
&\qquad (L,R) \text{ is a } \sigma\text{-cut of } T^{-1}(i) \big\}, \\
\mathtt{width}(G,\sigma,T,i) &= \max \{ \, |E_G(L,R)| \; : \; (L,R) \in \mathtt{cuts}(G,\sigma,T,i) \, \}.
\end{aligned}$$

Notice that every σ-cut of G is in $\mathtt{cuts}(G, \sigma, T, i)$ for at least one bucket $i \in [\ell]$; since $\mathbf{cw}_\sigma(G)$ is the maximum of $|E_G(L, R)|$ over σ-cuts (L, R), we have

$$\mathbf{cw}_\sigma(G) = \max_{i \in [\ell]} \mathtt{width}(G, \sigma, T, i). \tag{1}$$

For two k-boundaried graphs $\mathbf{A} = (A, \bar{x}), \mathbf{B} = (B, \bar{y})$, we slightly abuse notation and understand the edges $x_1x'_1, \ldots, x_kx'_k$ in A^* to be the same as $y'_1y_1, \ldots, y'_ky_k$ in B^* and as $x_1y_1, \ldots, x_ky_k$ in $\mathbf{A} \oplus \mathbf{B}$. That is, for an ordering σ of $\mathbf{A} \oplus \mathbf{B}$ with ℓ-bucketing T, we define $T|_{A^*}(v)$ to be $T(v)$ for $v \in V(A)$ and $T(y_i)$ for $v = x'_i$. We define $\sigma|_{A^*}$ as an ordering that orders x'_i just as σ orders y_i, with the order between x'_i and x'_j chosen arbitrarily when $y_i = y_j$. The following lemma shows that if an ℓ-bucketing respects the sides of a cut, then the width of any bucket can be computed as the sum of contributions of the sides.

▶ **Lemma 8** (♠[3]). *Let k, ℓ be positive integers and $\mathbf{A} = (A, \bar{x}), \mathbf{B} = (B, \bar{y})$ be two k-boundaried graphs. Let also σ be a vertex ordering of $\mathbf{A} \oplus \mathbf{B}$ with ℓ-bucketing T. If $T^{-1}(i)$ does not contain any vertex of A, for some $i \in [\ell]$, that is, $T^{-1}(i) \cap V(A) = \emptyset$, then it holds that* $\texttt{width}(\mathbf{A} \oplus \mathbf{B}, \sigma, T, i) = \texttt{width}(A, \sigma|_A, T|_A, i) + \texttt{width}(B^*, \sigma|_{B^*}, T|_{B^*}, i)$.

Replacing the roles of $\mathbf{A}$ and $\mathbf{B}$ above, we obtain that if $T^{-1}(i)$ does not contain any vertex of B, then $\texttt{width}(\mathbf{A} \oplus \mathbf{B}, \sigma, T, i) = \texttt{width}(A^*, \sigma|_{A^*}, T|_{A^*}, i) + \texttt{width}(B, \sigma|_B, T|_B, i)$. Intuitively, this implies that the cutwidth of $\mathbf{A} \oplus \mathbf{B}$ depends on A only in the widths of each block relative to A and A^* (in any bucketing where buckets are either A-blocks or B-blocks). Therefore, replacing $\mathbf{A}$ with another boundaried graph whose extension has an ordering and bucketing with the same widths preserves cutwidth (as long as endpoints of the cut edges are placed in the same buckets too). This is formalized in the next definition.

▶ **Definition 9.** For $k, \ell \in \mathbb{N}$, a *(k,ℓ)-bucket interface* consists of functions:
- $b, b' : [k] \to [\ell]$ identifying the buckets which contain x_i and x'_i, respectively and
- $\mu, \mu^* : [\ell] \to [0, k]$ corresponding to the widths of buckets.

A k-boundaried graph $\mathbf{G}$ *conforms* with a (k, ℓ)-bucket interface if there exists an ordering σ of the vertices of G^* and an ℓ-bucketing T of G^* such that:
- $T(v)$ is odd for $v \in V(G)$ and even for $v \in \{x'_1, \ldots, x'_k\}$,
- $T(x_i) = b(i)$ and $T(x'_i) = b'(i)$, for each $i \in [k]$,
- $\texttt{width}(G, \sigma|_G, T|_G, j) \leq \mu(j)$, for each $j \in [\ell]$,
- $\texttt{width}(G^*, \sigma, T, j) \leq \mu^*(j)$, for each $j \in [\ell]$.

▶ **Observation 10.** *For all $k, \ell \in \mathbb{N}^+$ there are $\leq 2^{2(k \log \ell + \ell \log(k+1))}$ (k, ℓ)-bucket interfaces.*

We call two k-boundaried graphs $\mathbf{G}_1, \mathbf{G}_2$ *(k,ℓ)-similar* if the sets of (k, ℓ)-bucket interfaces they conform with are equal. The following lemma subsumes the above ideas. The proof follows easily from Lemma 8 and the fact that $\mathbf{cw}_\sigma(G) = \max_{i \in [\ell]} \texttt{width}(G, \sigma, T, i)$ (Eq. (1)).

▶ **Theorem 11** (♠). *Let k, r be two positive integers. Let also $\mathbf{A}_1$ and $\mathbf{A}_2$ be two k-boundaried graphs that are (k, ℓ)-similar, where $\ell = (2k+1) \cdot (2r+4)$. Then for any k-boundaried graph $\mathbf{B}$ where $\mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B}) \leq r$, it holds that $\mathbf{cw}(\mathbf{A}_2 \oplus \mathbf{B}) = \mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B})$.*

4 Obstruction sizes and linked orderings

In this section we establish the main result on sizes of obstructions for cutwidth. We first define linked orderings and prove that there is always an optimum ordering that is linked.

▶ **Definition 12** (linked ordering). An ordering σ of $V(G)$ is *linked* if for any two vertices $u \leq_\sigma u'$, there are $\min\{\delta(V_v^\sigma) \mid u \leq_\sigma v \leq_\sigma u'\}$ edge-disjoint paths between V_u^σ and $V(G) \setminus V_{u'}^\sigma$ in G.

▶ **Lemma 13** ([8, 12]). *For every graph G, there is a linked ordering σ of $V(G)$ with $\mathbf{cw}_\sigma(G) = \mathbf{cw}(G)$.*

Proof. Without loss of generality, we may assume that the graph is connected. Let σ be an optimum cutwidth ordering of $V = V(G)$. Subject to the optimality of σ, we choose σ so that $\sum_{v \in V} \delta(V_v^\sigma)$ is minimized. We prove that σ defined in this manner is in fact linked.

Assume the contrary. Then by Menger's theorem, there exist vertices $u <_\sigma u'$ in V and $i \in \mathbb{N}$ such that $\delta(V_v^\sigma) > i$ for every $u \leq_\sigma v \leq_\sigma u'$, but a minimum cut (A, B) of G with

[3] Proofs of statements marked with ♠ are ommited from this extended abstract. The full version of the paper is available in http://arxiv.org/abs/1606.05975.

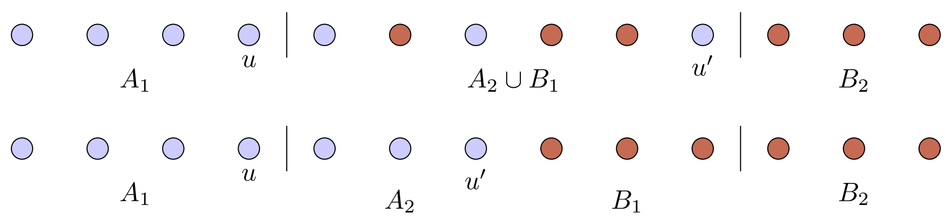

Figure 2 An ordering of vertices with the minimum cut (A, B) between A_1 and B_2 of size i highlighted in blue and red. Below, the modified ordering, with cutwidth bounded using submodularity.

$V_u^\sigma \subseteq A$ and $V \setminus V_{u'}^\sigma \subseteq B$ has size $\delta(A) \le i$. We partition A into sets A_1 and A_2, where $A_1 = V_u^\sigma$ and $A_2 = A \setminus A_1$, and we partition B into sets B_1 and B_2, where $B_2 = V \setminus V_{u'}^\sigma$ and $B_1 = B \setminus B_2$ (see Figure 2). Notice that $A_2 = A \setminus V_u^\sigma = \{v \mid u <_\sigma v \le_\sigma u'\} \cap A$ and that $B_1 = B \setminus (V \setminus V_{u'}^\sigma) = \{v \mid u <_\sigma v \le_\sigma u'\} \cap B$. Let σ' be the ordering of V obtained by concatenating $\sigma|_{A_1}$, $\sigma|_{A_2}$, $\sigma|_{B_1}$, and $\sigma|_{B_2}$.

We prove that $\delta(V_v^{\sigma'}) \le \delta(V_v^\sigma)$, for every $v \in V$. Observe first that for every vertex $v \in A_1 \cup B_2$ it holds that $V_v^{\sigma'} = V_v^\sigma$ and thus, $\delta(V_v^{\sigma'}) = \delta(V_v^\sigma)$. Let now $v \in A_2$. Then $V_v^{\sigma'} = V_v^\sigma \cap A$. By the submodularity of cuts it follows that $\delta(V_v^\sigma \cup A) + \delta(V_v^\sigma \cap A) \le \delta(A) + \delta(V_v^\sigma)$. Notice that $(V_v^\sigma \cup A, V \setminus (V_v^\sigma \cup A))$ is also a cut separating $A_1 = V_u^\sigma$ and $B_2 = V \setminus V_{u'}^\sigma$. From the minimality of (A, B) it follows that $\delta(A) \le \delta(V_v^\sigma \cup A)$. Therefore, $\delta(V_v^\sigma \cap A) \le \delta(V_v^\sigma)$. As $V_v^{\sigma'} = V_v^\sigma \cap A$, we obtain that $\delta(V_v^{\sigma'}) \le \delta(V_v^\sigma)$.

Symmetrically, let now $v \in B_1$. Then $V_v^{\sigma'} = V_v^\sigma \cup A$. By the submodularity of cuts we have $\delta(V_v^\sigma \cup A) + \delta(V_v^\sigma \cap A) \le \delta(A) + \delta(V_v^\sigma)$. Notice that $(V_v^\sigma \cap A, V \setminus (V_v^\sigma \cap A))$ is a cut separating A_1 and B_2. From the minimality of (A, B) it follows that $\delta(A) \le \delta(V_v^\sigma \cap A)$. Therefore, $\delta(V_v^\sigma \cup A) \le \delta(V_v^\sigma)$. As $V_v^{\sigma'} = V_v^\sigma \cup A$, we obtain that $\delta(V_v^{\sigma'}) \le \delta(V_v^\sigma)$.

Thus, $\delta(V_v^{\sigma'}) \le \delta(V_v^\sigma) \le \mathbf{cw}(G)$ for every $v \in V$, and hence $\mathbf{cw}_{\sigma'}(G) = \mathbf{cw}(G)$. Finally, note that $\delta(V_v^{\sigma'}) = \delta(A) \le i < \delta(V_v^\sigma)$ for the last vertex v in A. Thus $\sum_v \delta(V_v^{\sigma'}) < \sum_v \delta(V_v^\sigma)$, contradicting the choice of σ. Therefore, σ is a linked ordering of V with $\mathbf{cw}_\sigma(G) = \mathbf{cw}(G)$. ◀

The following theorem is the technical counterpart of Theorem 1. Its proof is based on Theorem 11, Lemma 13, Observation 10 and the idea of "unpumping" repeating types, presented in the introduction. The linkedness is used to make sure that within the unpumped segment of the ordering, one can find the maximum possible number of edge-disjoint paths between the parts of the graph on the left side and on the right side of the segment. This ensures that the graph obtained from unpumping can be immersed in the original one.

▶ **Theorem 14** (♠). *Let k be a positive integer. If $G \in \mathbf{obs}_{\le_{\mathrm{si}}}(\mathcal{C}_k)$, then $|V(G)| \le N^{k+1}$, where $N = 2^{2((k+1)\log \ell + \ell \log(k+2))} + 2$ and $\ell = (2k+3) \cdot (2k+6)$.*

Theorem 14 provides an upper bound on the number of vertices of a graph in $\mathbf{obs}_{\le_{si}}(\mathcal{C}_k)$. Observe that since such a graph has cutwidth $k+1$, each of its vertices has degree at most $2(k+1)$. It follows that any graph from $\mathbf{obs}_{\le_{si}}(\mathcal{C}_k)$ has $2^{O(k^3 \log k)}$ vertices and edges. Finally, by Observation 3 we have $\mathbf{obs}_{\le_i}(\mathcal{C}_q) \subseteq \mathbf{obs}_{\le_{si}}(\mathcal{C}_q)$, so the same bound holds also for immersions instead of strong immersions. This concludes the proof of Theorem 1.

5 An algorithm for computing cutwidth

In this section we present an exact FPT algorithm for computing the cutwidth of the graph. First, we need to give a dynamic programming algorithm that given an approximate ordering σ of width r, finds, if possible, an ordering of width at most k, where $k \le r$ is given.

▶ **Lemma 15 (♠).** *Let $r \in \mathbb{N}^+$. Given a graph G and an ordering σ of its vertices with $\mathbf{cw}_\sigma(G) \le r$, an ordering τ of the vertices of G with $\mathbf{cw}_\tau(G) = \mathbf{cw}(G)$ can be computed in time $2^{O(r^2 \log r)} \cdot |V(G)|$.*

The main ingredient in the proof of Lemma 15 is the insight given by Lemma 4: any set X with $\delta(X) \le r$, in particular any prefix of σ, can be assumed to be split into at most $O(kr)$ blocks in some optimum ordering τ. A closer look into the proof of Lemma 4 shows that a stronger statement is true: there is some optimum ordering τ such that for *every* prefix X of σ, there are at most $O(kr)$ X-blocks in σ. This shows that an optimum ordering τ can be constructed by a dynamic programming procedure that scans σ from left to right while constructing an optimum ordering, maintained as a mapping of the already scanned vertices into at most ℓ blocks. The description of such a partial ordering is an enrichment of a (k, ℓ)-bucket interface that we call a *(k, ℓ)-bucket profile.* In such a profile, we additionally store some information about the sizes of cuts which is needed to make sure that the constructed ordering has width at most k. With this understanding, the construction of the dynamic programming algorithm is a routine, though technical task.

Having the algorithm of Lemma 15, a standard application of the iterative compression technique immediately yields a $2^{O(k^2 \log k)} \cdot n^2$ time algorithm for computing cutwidth, as sketched in Section 1. Simply add the vertices of G one by one, and apply the algorithm of Lemma 15 at each step. However, we can make the dependence on n linear by adapting the approach of Bodlaender [2]; more precisely, we make bigger steps. Such a big step consists of finding a graph H that can be immersed in the input graph G, which is smaller by a constant fraction, but whose cutwidth is not much smaller. This is formalized in the next lemma.

▶ **Lemma 16 (♠).** *There is an algorithm that given a positive integer k and a graph G, works in time $O(k^2 \cdot |V(G)|)$ and either concludes that $\mathbf{cw}(G) > k$, or finds a graph H immersed in G such that $|E(H)| \le |E(G)| \cdot (1 - 1/(2k+1)^{4(k+1)+3})$ and $\mathbf{cw}(G) \le 2\mathbf{cw}(H)$. Furthermore, in the latter case, given an ordering σ of the vertices of H, an ordering τ of the vertices of G with $\mathbf{cw}_\tau(G) \le 2\mathbf{cw}_\sigma(H)$ can be computed in $O(|V(G)|)$ time.*

The proof starts by iteratively dissolving vertices of degree two with both incident edges going to different neighbors; this operation preserves the cutwidth of the graph. Then, if a constant fraction (depending on k) of vertices has degree equal to one, we can find a large matching of edges incident to degree-1 vertices. If we remove them, the size of the graph drops by a constant fraction, but reintroducing them increases the cutwidth by at most one.

Otherwise, we arrive at the situation when there are no vertices of degree 2, apart from the ones with only one neighbor, and there is only a very small fraction of vertices of degree one. Our goal now is to find a large (constant fraction of $|E(G)|$, where the constant depends on k) packing of edge-disjoint cycles in G. If we succeed in this, then an easy charging argument shows that removing one edge from each of these cycles yields a graph H with $\mathbf{cw}(H) \ge \mathbf{cw}(G)/2$, whereas the size of H is smaller than G by a constant fraction. Now, if there are many vertices of degree 2 with both incident edges going to the same neighbor, then we have a large packing of 2-cycles and we are done. So we can assume that the total number of vertices of degree 1 and 2 is only a small fraction of the total size of the graph.

At this point, we greedily find a large family of disjoint balls of radius $2(k+1)$ (sets of vertices at distance at most $2(k+1)$ from some central vertex), each containing no vertex of degree 1 or 2. If any of these balls induced a tree in G, it would contain a full binary tree of height $2(k+1)$; but such a binary tree is known to have cutwidth larger than k by itself, allowing us to conclude that $\mathbf{cw}(G) > k$. Finally, if every ball contains a cycle, then we have found a large packing of vertex-disjoint, hence also edge-disjoint cycles.

We are now ready to put all the pieces together and prove Theorem 2: given an n-vertex graph G and an integer k, one can in time $2^{O(k^2 \log k)} \cdot n$ either conclude that $\mathbf{cw}(G) > k$, or output an ordering of G of width at most k. The proof follows the same recursive Reduction&Compression scheme as the algorithm of Bodlaender [2]. By applying Lemma 16, we obtain a significantly smaller immersion H, and we recurse on H. This recursive call either concludes that $\mathbf{cw}(H) > k$, which implies $\mathbf{cw}(G) > k$, or it produces an ordering of H of optimum width $\mathbf{cw}(H) \leq k$. This ordering can be lifted, using Lemma 16 again, to an ordering of G of width $\leq 2k$. Given this ordering, we apply the dynamic programming procedure of Lemma 15 to construct an optimum ordering of G in time $2^{O(k^2 \log k)} \cdot |V(G)|$.

Since at each recursion step the number of edges of the graph drops by a multiplicative factor of at least $1/(2k+1)^{4(k+1)+3}$, we see that the graph G_i at level i of the recursion will have at most $(1 - 1/(2k+1)^{4(k+1)+3})^i \cdot |E(G)|$ edges. Hence, the total work used by the algorithm is bounded by the sum of a geometric series:

$$\begin{aligned}\sum_{i=0}^{\infty} 2^{O(k^2 \log k)} \cdot |E(G_i)| &\leq 2^{O(k^2 \log k)} \cdot |E(G)| \cdot \sum_{i=0}^{\infty} (1 - 1/(2k+1)^{4k+7})^i \\ &= 2^{O(k^2 \log k)} \cdot |E(G)| \cdot (2k+1)^{4k+7} = 2^{O(k^2 \log k)} \cdot |E(G)|.\end{aligned}$$

6 Conclusions

In this paper we have proved that the immersion obstructions for admitting a layout of cutwidth at most k have sizes single-exponential in $O(k^3 \log k)$. The core of the proof can be interpreted as bounding the number of different behavior types for a part of the graph that has only a small number of edges connecting it to the rest. This, in turn, gives an upper bound on the number of states for a dynamic programming algorithm that computes the optimum cutwidth ordering on an approximate one. This last result, complemented with an adaptation of the reduction scheme of Bodlaender [2] to the setting of cutwidth, yields a direct and self-contained FPT algorithm for computing the cutwidth of a graph. In fact, we believe that our algorithm can be thought of "Bodlaender's algorithm for treewidth in a nutshell". It consists of the same two components, namely a recursive reduction scheme and dynamic programming on an approximate decomposition, but the less challenging setting of cutwidth makes both components simpler, thus making the key ideas easier to understand. For an alternative attempt of simplification of the algorithm of Bodlaender and Kloks [3], applied for the case of pathwidth, see [6].

In our proof of the upper bound on the number of types/states, we used a somewhat new bucketing approach. This approach holds the essence of the typical sequences of Bodlaender and Kloks [3], but we find it more natural and conceptually simpler. The drawback is that we lose a $\log k$ factor in the exponent. It is conceivable that we could refine our results by removing this factor provided we applied typical sequences directly, but this is a price that we are willing to pay for the sake of simplicity and being self-contained.

An important ingredient of our approach is the observation that there is always an optimum cutwidth ordering that is linked: the cutsizes along the ordering precisely govern the edge connectivity between prefixes and suffixes. Recently, there is a growing interest in parameters that are tree-like analogues of cutwidth: tree-cut width [20] and carving-width [16]. In future work, we aim to explore and use linkedness for tree-cut decompositions and carving decompositions in a similar manner as presented here.

Acknowledgements. The second author thanks Mikołaj Bojańczyk for the common work on understanding and reinterpreting the Bodlaender-Kloks dynamic programming algorithm [3], which influenced the bucketing approach presented in this paper. We also thank O-joung Kwon for pointing us to [8, 12], as well as an anonymous referee for noting that the running time in Lemma 16 can be reduced to polynomial by amortization.

References

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

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

3 Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21(2):358–402, 1996.

4 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.

5 Josep Díaz, Jordi Petit, and Maria J. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002.

6 Martin Fürer. Faster computation of path-width. In Veli Mäkinen, J. Simon Puglisi, and Leena Salmela, editors, *Combinatorial Algorithms: 27th International Workshop, IWOCA 2016, Helsinki, Finland, August 17-19, 2016, Proceedings*, pages 385–396, Cham, 2016. Springer International Publishing.

7 Michael R. Garey and David S. Johnson. *Computers and intractability*, volume 174. Freeman New York, 1979.

8 James F. Geelen, A. M. H. Gerards, and Geoff Whittle. Branch-width and well-quasi-ordering in matroids and graphs. *J. Comb. Theory, Ser. B*, 84(2):270–290, 2002. A correction is available at `http://www.math.uwaterloo.ca/~jfgeelen/Publications/bn-corr.pdf`.

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

10 Pinar Heggernes, Daniel Lokshtanov, Rodica Mihai, and Charis Papadopoulos. Cutwidth of split graphs and threshold graphs. *SIAM J. Discrete Math.*, 25(3):1418–1437, 2011.

11 Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Jesper Nederlof. Computing the cutwidth of bipartite permutation graphs in linear time. *SIAM J. Discrete Math.*, 26(3):1008–1021, 2012.

12 Mamadou Moustapha Kanté and O-joung Kwon. An upper bound on the size of obstructions for bounded linear rank-width. *CoRR*, arXiv:1412.6201, 2014.

13 Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *J. Comb. Theory, Ser. B*, 73(1):7–40, 1998.

14 Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

15 Neil Robertson and Paul D. Seymour. Graph minors XXIII. Nash-Williams' immersion conjecture. *J. Comb. Theory, Ser. B*, 100(2):181–205, 2010.

16 Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

17 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *J. Algorithms*, 56(1):1–24, 2005.

18 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth II: Algorithms for partial w-trees of bounded degree. *J. Algorithms*, 56(1):25–49, 2005.

19 Robin Thomas. A Menger-like property of tree-width: The finite case. *J. Comb. Theory, Ser. B*, 48(1):67–76, 1990.

20 Paul Wollan. The structure of graphs not admitting a fixed immersion. *J. Comb. Theory, Ser. B*, 110:47–66, 2015.

21 Mihalis Yannakakis. A polynomial algorithm for the min-cut linear arrangement of trees. *J. ACM*, 32(4):950–988, 1985.