Cutwidth: obstructions and algorithmic aspects[a,b]

Archontia C. Giannopoulou[c] Michał Pilipczuk[d] Jean-Florent Raymond[d,e]

Dimitrios M. Thilikos[e,f] Marcin Wrochna[d]

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 $2^{O(k^3 \log k)}$. For our proof, we introduce the concept of a lean ordering that can be seen as the analogue of lean decompositions defined by Thomas in [A Menger-like property of tree-width: The finite case, *J. Comb. Theory, Ser. B*, 48(1):67–76, 1990] for the case of treewidth. 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, given by Thilikos, Bodlaender, and Serna in [Cutwidth I: A linear time fixed parameter algorithm, *J. Algorithms*, 56(1):1–24, 2005] and [Cutwidth II: Algorithms for partial w-trees of bounded degree, *J. Algorithms*, 56(1):25–49, 2005], our algorithm has the advantage of being simpler and self-contained; arguably, it explains better the combinatorics of optimum-width layouts.

Keywords: cutwidth, obstructions, immersions, fixed-parameter tractability.

---

[a]This work was partially done while Archontia C. Giannopoulou was holding a post-doc position at Warsaw Center of Mathematics and Computer Science. The research of Archontia 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 Michał Pilipczuk and Marcin Wrochna is supported by the Polish National Science Center grant SONATA UMO-2013/11/D/ST6/03073. The research of Jean-Florent Raymond is supported by the Polish National Science Center grant PRELUDIUM UMO-2013/11/N/ST6/02706. Michał Pilipczuk is supported by the Foundation for Polish Science (FNP) via the START stipend programme.

[b]Emails: archontia.giannopoulou@tu-berlin.de, michal.pilipczuk@mimuw.edu.pl, jean-florent.raymond@mimuw.edu.pl, sedthilk@thilikos.info, and m.wrochna@mimuw.edu.pl.

[c]Technische Universität Berlin, Berlin, Germany.

[d]Institute of Informatics, University of Warsaw, Poland.

[e]AlGCo project team, CNRS, LIRMM, Montpellier, France.

[f]Department of Mathematics, National and Kapodistrian University of Athens, Athens, Greece.

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 [6] for a broader discussion.

As finding a layout of optimum width is NP-hard [8], the algorithmic and combinatorial aspects of cutwidth were intensively studied. There is a broad range of polynomial-time algorithms for special graph classes [11,12,23], approximation algorithms [15], and fixed-parameter algorithms [19, 20]. In particular, Thilikos, Bodlaender, and Serna [19, 20] 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] [16], 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 graph parameters like pathwidth, treewidth, or cutwidth, has been studied before. For minor-closed parameters pathwidth and treewidth, Lagergren [14] 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 tree decompositions he showed an upper bound double-exponential in $O(k^5)$. Less is known about immersion-closed parameters, like cutwidth. Govindan and Ramachandramurthi [10] showed that the number of minimal immersion obstructions for the class of graphs of cutwidth at most k is at least $3^{k-7} + 1$, and their construction actually exemplify 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.

[1] Thilikos, Bodlaender, and Serna [19, 20] 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.

1.1 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.*

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 [10].

Our approach for Theorem 1 follows the technique used by Lagergren [14] 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*, also known as *lean path decompositions*; cf. [21].

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, each comprising prefixes that have the same "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 introduce *lean orderings* for cutwidth and prove the analogue of the result of Thomas [21] for treewidth: there is always an optimum-width ordering that is lean (see also [1]).

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. These results are presented in Section 6.

1.2 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 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 at most twice. We then run the compression algorithm on the obtained 2-approximate ordering to get an optimum one. The main point is that, 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 [19, 20]. 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 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 $\sigma_1 = \langle y_1, \ldots, y_{r_2} \rangle$ 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 $G' \lneqq G$ 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 $G' \lneqq G$, 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 [16] 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 [16] 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 run 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 run 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 $\mu(j)$

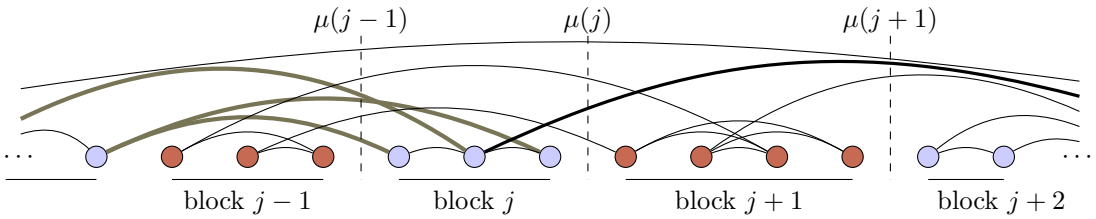

Figure 1: A cut (A, B) is highlighted (blue, red), with the corresponding blocks underlined and cuts between them marked with dashed lines. Edges counted as p_j and s_j are thickened.

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.

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* $\texttt{cuts}(G, \sigma, T, i)$ *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* $\texttt{cuts}(G, \sigma, T, i)$*. Formally,*

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

Notice that every σ-cut of G is in $\texttt{cuts}(G, \sigma, T, i)$ 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

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

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. *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* $\texttt{width}(\mathbf{A} \oplus \mathbf{B}, \sigma, T, i) = \texttt{width}(A, \sigma|_A, T|_A, i) + \texttt{width}(B^*, \sigma|_{B^*}, T|_{B^*}, i)$*.*

Proof. Consider any cut (L, R) in $\texttt{cuts}(G, \sigma, T, i)$. Observe that for every edge e of $E_{\mathbf{A}\oplus\mathbf{B}}(L, R)$ one of the following holds:

1. $e \in E_A(L \cap V(A), R \cap V(A))$ or
2. $e \in E_B(L \cap V(B), R \cap V(B))$ or
3. $e \in E_G(L \cap V(A), R \cap V(B))$, or
4. $e \in E_G(R \cap V(A), L \cap V(B))$.

Since we do not distinguish between the vertices x_i and the vertices y'_i, we equivalently obtain that for every edge $e \in E_{\mathbf{A}\oplus\mathbf{B}}(L, R)$, e is either an edge in $E_A(L \cap V(A), R \cap V(A))$ or an edge in $E_{B^*}(L \cap V(B^*), R \cap V(B^*))$. Observe that $(L \cap V(A), R \cap V(A))$ is a cut in $\texttt{cuts}(A, \sigma|_A, T|_A, i)$ and $(L \cap V(B^*), R \cap V(B^*))$ is a cut in $\texttt{cuts}(B^*, \sigma|_{B^*}, T|_{B^*}, i)$. Therefore, the total number of edges crossing these cuts is at most $\texttt{width}(A, \sigma|_A, T|_A, i) + \texttt{width}(B^*, \sigma|_{B^*}, T|_{B^*}, i)$. This proves that

$$\texttt{width}(\mathbf{A} \oplus \mathbf{B}, \sigma, T, i) \leq \texttt{width}(A, \sigma|_A, T|_A, i) + \texttt{width}(B^*, \sigma|_{B^*}, T|_{B^*}, i).$$

For the converse inequality, observe that since the bucket $T^{-1}(i)$ does not contain any vertices of A, we have $T|_A^{-1}(i) = \emptyset$. Hence there is exactly one cut in $\texttt{cuts}(A, \sigma|_A, T|_A, i)$, namely (L_A, R_A), where $L_A = T^{-1}(\{1, \ldots, i-1\}) \cap V(A)$ and $R_A = T^{-1}(\{i+1, \ldots, \ell\}) \cap V(A)$. Let (L_B, R_B) be a cut in $\texttt{cuts}(B^*, \sigma|_{B^*}, T|_{B^*}, i)$ maximizing $|E_{B^*}(L_B, R_B)|$. Then, since we assumed that $T^{-1}(i)$ does not contain any vertices of A (and thus, may only contain vertices of B), it follows that $(L_A \cup L_B, R_A \cup R_B)$ is a cut in $\texttt{cuts}(G, \sigma, T, i)$. As above, every edge of $\mathbf{A} \oplus \mathbf{B}$ crossing this cut is either in $E_A(L_A, R_A)$ or in $E_{B^*}(L_B, R_B)$, where we again do not distinguish between the vertices x_i and y'_i. Hence

$$\begin{aligned}
\texttt{width}(\mathbf{A} \oplus \mathbf{B}, \sigma, T, i) &\geq |E_{\mathbf{A}\oplus\mathbf{B}}(L, R)| \\
&= |E_A(L_A, R_A)| + |E_{B^*}(L_B, R_B)| \\
&= \texttt{width}(A, \sigma|_A, T|_A, i) + \texttt{width}(B^*, \sigma|_{B^*}, T|_{B^*}, i).
\end{aligned}$$

□

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 $k, \ell \in \mathbb{N}$, an* (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 *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* $\mathbf{cw}(\mathbf{A}_2 \oplus \mathbf{B}) = \mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B})$*.*

Proof. Let $\mathbf{A}_i = (A_i, \bar{x}^i), \mathbf{B} = (B, \bar{y})$ and suppose that $\mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B}) \leq r$. By Lemma 4, there is an optimum cutwidth ordering σ_1 of the vertices of $\mathbf{A}_1 \oplus \mathbf{B}$ that has at most $\ell - 1$ $(V(A_1), V(B))$-blocks. In particular $\mathbf{cw}_{\sigma_1}(\mathbf{A}_1 \oplus \mathbf{B}) = \mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B}) \leq r$. By adding an empty block at the front, if necessary, we may assume that the number of blocks is at most ℓ, while odd-indexed blocks are $V(A_1)$-blocks and even-indexed blocks are $V(B)$-blocks. Then, there is an ℓ-bucketing T_1 of σ_1 such that $T_1(v)$ is odd for $v \in A_1$ and even for $v \in B$. Therefore $\sigma_1|_{A_1^*}$ and $T_1|_{A_1^*}$ certify that the following (k, ℓ)-bucket interface conforms with $\mathbf{A}_1$:

- $b(i) = T_1(x_i^1)$ and $b'(i) = T_1|_{A_1^*}(x_i^{1\prime}) = T_1(y_i)$ for $i \in [k]$,
- $\mu(i) = \texttt{width}(A_1, \sigma_1|_{A_1}, T_1|_{A_1}, i)$ and $\mu^*(i) = \texttt{width}(A_1^*, \sigma_1|_{A_1^*}, T_1|_{A_1^*}, i)$ for $i \in [\ell]$.

By (k, ℓ)-similarity there is an ordering σ_2 of A_2^* and its ℓ-bucketing T_2 such that:

- each bucket $T_2^{-1}(i)$ is contained in A_2 for odd $i \in [\ell]$ and in $\{x_1^{2\prime}, \ldots, x_k^{2\prime}\}$ for even $i \in [\ell]$
- $b(i) = T_2(x_i^2)$ and $b'(i) = T_2(x_i^{2\prime})$ for $i \in [k]$,
- $\mu(i) \geq \texttt{width}(A_2, \sigma_2|_{A_2}, T_2|_{A_2}, i)$ and $\mu^*(i) \geq \texttt{width}(A_2^*, \sigma_2|_{A_2^*}, T_2|_{A_2^*}, i)$ for $i \in [\ell]$.

Given this, we define an assignment of vertices into buckets $\Pi \colon V(\mathbf{A}_2 \oplus \mathbf{B}) \to [\ell]$ as follows.

- $\Pi(v) = T_1(v)$ for $v \in V(B)$ and
- $\Pi(v) = T_2(v)$ for $v \in V(A_2)$.

Clearly,

$$\Pi|_B = T_1|_B \qquad \text{and} \tag{2}$$
$$\Pi|_{A_2} = T_2|_{A_2}. \tag{3}$$

We claim that $\Pi|_{A_2^*} = T_2|_{A_2^*}$ and $\Pi|_{B^*} = T_1|_{B^*}$ also hold. Indeed,

$$\begin{aligned}
\Pi|_{A_2^*}(x_i^{2'}) &= \Pi(y_i) && \text{(we consider } x_i^{2'} \text{ as } y_i\text{)} \\
&= T_1(y_i) && \text{(by definition)} \\
&= b'(i) && \text{(}(k,\ell)\text{-bucket interface)} \\
&= T_2(x_i^{2'}) && \text{(}(k,\ell)\text{-similarity)}
\end{aligned}$$

and, similarly,

$$\begin{aligned}
\Pi|_{B^*}(y_i') &= \Pi(x_i^2) && \text{(we consider } y_i' \text{ as } x_i^2\text{)} \\
&= T_2(x_i^2) && \text{(by definition)} \\
&= b(i) && \text{(}(k,\ell)\text{-bucket interface)} \\
&= T_1(x_i^1) && \text{(}(k,\ell)\text{-similarity)} \\
&= T_1|_{B^*}(y_i') && \text{(by definition).}
\end{aligned}$$

Thus, we obtain that

$$\Pi|_{A_2^*} = T_2|_{A_2} \tag{4}$$
$$\Pi|_{B^*} = T_1|_{B^*}. \tag{5}$$

Note also that vertices of A_2 are mapped to odd buckets and vertices of B are mapped to even buckets. We use Π to define an ordering π of the vertices of $\mathbf{A_2} \oplus \mathbf{B}$ as follows. Formally, we let $u <_\pi u$ if and only if one of the following conditions hold:

1. $\Pi(u) < \Pi(v)$,
2. $u <_{\sigma_2} v$ and $\Pi(u) = \Pi(v)$ is odd, or
3. $u <_{\sigma_1} v$ and $\Pi(u) = \Pi(v)$ is even.

Note that this is a linear ordering as it first sorts the vertices according to the bucket they belong to and then according to the ordering induced in this bucket by the orderings σ_1 and σ_2. Note also that by definition Π is an ℓ-bucketing of π. Recall that, from Eq. (4), $\Pi|_{A_2^*} = T_2|_{A_2}$. This, together with the observation that the vertices of A_2 are mapped to odd buckets of Π, implies that

$$\pi|_{A_2^*} = \sigma_2|_{A_2^*} \qquad \text{and that} \tag{6}$$
$$\pi|_{A_2} = \sigma_2|_{A_2}. \tag{7}$$

Moreover, recall that $\Pi|_{B^*} = T_1|_{B^*}$. This, together with the fact that the vertices of B are mapped to even buckets of Π, implies that

$$\pi|_{B^*} = \sigma_1|_{B^*} \qquad \text{and that} \tag{8}$$

$$\pi|_B = \sigma_1|_B. \tag{9}$$

We now bound the width of each bucket. Let $i \in [\ell]$. Notice that if i is even the by construction $\Pi^{-1}(i)$ contains only vertices from B. Therefore,

$$\begin{aligned}
\texttt{width}(\mathbf{A}_2 \oplus \mathbf{B}, \pi, \Pi, i) &= \texttt{width}(A_2, \pi|_{A_2}, \Pi|_{A_2}, i) + \texttt{width}(B^*, \pi|_{B^*}, \Pi|_{B^*}, i) \\
&= \texttt{width}(A_2, \sigma_2|_{A_2}, T_2|_{A_2}, i) + \texttt{width}(B^*, \sigma_1|_{B^*}, T_1|_{B^*}, i) \\
&\leq \mu(i) + \texttt{width}(B^*, \sigma_1|_{B^*}, T_1|_{B^*}, i) \\
&= \texttt{width}(A_1, \sigma_1|_{A_1}, T_1|_{A_1}, i) + \texttt{width}(B^*, \sigma_1|_{B^*}, T_1|_{B^*}, i) \\
&= \texttt{width}(\mathbf{A}_1 \oplus \mathbf{B}, \sigma_1, T_1, i),
\end{aligned} \tag{10}$$

where the first equality follows from Lemma 8, the second equality holds by Eq. (3), (7), (8), and (5), the third inequality follows from the (k, ℓ)-bucket interface, and the fifth equality follows from Lemma 8. We similarly argue, using μ^* instead of μ, that for odd $i \in [\ell]$, $\texttt{width}(\mathbf{A}_2 \oplus \mathbf{B}, \pi, \Pi, i) = \texttt{width}(\mathbf{A}_1 \oplus \mathbf{B}, \sigma_1, T_1, i)$. In particular,

$$\begin{aligned}
\texttt{width}(\mathbf{A}_2 \oplus \mathbf{B}, \pi, \Pi, i) &= \texttt{width}(A_2^*, \pi|_{A_2^*}, \Pi|_{A_2^*}, i) + \texttt{width}(B, \pi|_B, \Pi|_B, i) \\
&= \texttt{width}(A_2^*, \sigma_2|_{A_2^*}, T_2|_{A_2^*}, i) + \texttt{width}(B, \sigma_1|_B, T_1|_B, i) \\
&\leq \mu^*(i) + \texttt{width}(B, \sigma_1|_B, T_1|_B, i) \\
&= \texttt{width}(A_1^*, \sigma_1|_{A_1^*}, T_1|_{A_1^*}, i) + \texttt{width}(B, \sigma_1|_B, T_1|_B, i) \\
&= \texttt{width}(\mathbf{A}_1 \oplus \mathbf{B}, \sigma_1, T_1, i).
\end{aligned} \tag{11}$$

Similarly, to Eq. 10, we get that the first equality follows from Lemma 8, the second equality holds by Eq. (4), (6), (2), and (9), the third inequality follows from the (k, ℓ)-bucket interface, and the fifth equality follows from Lemma 8.

Therefore, from Eq. (10) and (11) we obtain that

$$\mathbf{cw}_\pi(\mathbf{A}_2 \oplus \mathbf{B}) = \max_{i \in [\ell]} \texttt{width}(\mathbf{A}_2 \oplus \mathbf{B}, \pi, \Pi, i) \leq \max_{i \in [\ell]} \texttt{width}(\mathbf{A}_1 \oplus \mathbf{B}, \sigma_1, T_1, i) = \mathbf{cw}_{\sigma_1}(\mathbf{A}_1 \oplus \mathbf{B}).$$

Moreover, since $\mathbf{cw}(\mathbf{A}_2 \oplus \mathbf{B}) \leq \mathbf{cw}_\pi(\mathbf{A}_2 \oplus \mathbf{B})$ and σ_1 is an optimum cutwidth ordering for $\mathbf{A}_\cdot \oplus \mathbf{B}$, it follows that

$$\mathbf{cw}(\mathbf{A}_2 \oplus \mathbf{B}) \leq \mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B}) \leq r.$$

So in particular $\mathbf{cw}(\mathbf{A}_2 \oplus \mathbf{B}) \leq r$. By applying the same reasoning, but with $\mathbf{A}_1$ and $\mathbf{A}_2$ reversed, we obtain also the converse inequality $\mathbf{cw}(\mathbf{A}_2 \oplus \mathbf{B}) \leq \mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B})$. This proves that indeed $\mathbf{cw}(\mathbf{A}_2 \oplus \mathbf{B}) = \mathbf{cw}(\mathbf{A}_1 \oplus \mathbf{B})$. □

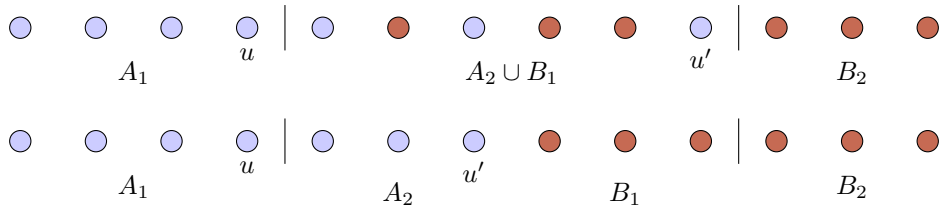

Figure 2: An ordering of vertices with the minimum cut (A, B) between A_1 and B_2 of size i highlighted in blue and red. Below, the modified ordering, with cutwidth bounded using submodularity.

4 Obstruction sizes and lean orderings

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

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

Lemma 13. *For every graph G, there is a lean 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 lean.

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 $V_u^\sigma \subseteq A$ and $V \setminus V_{u'}^\sigma \subseteq B$ has size $\delta(A) \leq i$. 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 $A_2 = A \setminus V_u^\sigma = \{v \mid u <_\sigma v \leq_\sigma u'\} \cap A$ and that $B_1 = B \setminus (V \setminus V_{u'}^\sigma) = \{v \mid u <_\sigma v \leq_\sigma u'\} \cap B$. 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 $\delta(V_v^{\sigma'}) \leq \delta(V_v^\sigma)$, 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 $\delta(V_v^\sigma \cup A) + \delta(V_v^\sigma \cap A) \leq \delta(A) + \delta(V_v^\sigma)$. 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 $\delta(A) \leq \delta(V_v^\sigma \cup A)$. Therefore, $\delta(V_v^\sigma \cap A) \leq \delta(V_v^\sigma)$. As $V_v^{\sigma'} = V_v^\sigma \cap A$, we obtain that $\delta(V_v^{\sigma'}) \leq \delta(V_v^\sigma)$.

Symmetrically, let now $v \in B_1$. Then $V_v^{\sigma'} = V_v^\sigma \cup A$. By the submodularity of cuts we have $\delta(V_v^\sigma \cup A) + \delta(V_v^\sigma \cap A) \leq \delta(A) + \delta(V_v^\sigma)$. 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 $\delta(A) \leq \delta(V_v^\sigma \cap A)$. Therefore, $\delta(V_v^\sigma \cup A) \leq \delta(V_v^\sigma)$. As $V_v^{\sigma'} = V_v^\sigma \cup A$, we obtain that $\delta(V_v^{\sigma'}) \leq \delta(V_v^\sigma)$.

Thus, $\delta(V_v^{\sigma'}) \leq \delta(V_v^\sigma) \leq \mathbf{cw}(G)$ for every $v \in V$, and hence $\mathbf{cw}_{\sigma'}(G) = \mathbf{cw}(G)$. Finally, note that $\delta(V_v^{\sigma'}) = \delta(A) \leq i < \delta(V_v^\sigma)$ 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 lean ordering of V with $\mathbf{cw}_\sigma(G) = \mathbf{cw}(G)$. □

The rest of Section 4 is devoted to the proof of Theorem 1. Before we proceed with this proof, we need a series of auxiliary lemmas.

For every $s, r \in \mathbb{N}^+$, we set $A_{s,r} = [s, s+r-1]$. We prove the following.

Lemma 14. *Let N be a positive integer. For every $s, r \in \mathbb{N}^+$ and every word w over $A_{s,r}$ of length N^r there is a symbol $k \in A_{s,r}$ and a subword u of w such that (a) u contains only numbers not smaller than k, and (b) u contains the number k at least N times.*

Proof. We prove the lemma by induction on r. Notice that for $r = 1$, $A_{s,r} = \{s\}$ and thus the only word w of length N is s^N. Thus, the lemma holds with $k = s$ and $u = w$. We proceed to the inductive step for $r > 1$.

Let now $s \in \mathbb{N}$ and let w be a word over $A_{s,r}$ of length N^r. If s occurs at least N times, then again, the lemma holds with $k = s$ and $u = w$. Thus, we may assume that s occurs at most $N-1$ times. Then, since w has length at least N^r, there exists a subword w' of w of length at least N^{r-1} over $A_{s,r} \setminus \{s\} = A_{s+1,r-1}$. From the inductive hypothesis, there exists $k \in A_{s+1,r-1} \subseteq A_{s,r}$ and a subword u of w' such that k occurs at least N times in u and u contains only numbers at least k. Since w' is a subword of w, u is also a subword of w. This completes the inductive step and the proof of the lemma. □

We use Lemma 14 only for $s = 1$, giving the following corollary.

Corollary 15. *Let r, N be positive integers and let w be a word of length N^r over the alphabet $[r]$. Then there is a number $k \in [r]$ and a subword u of w such that (a) u contains only numbers not smaller than k, and (b) u contains the number k at least N times.*

We also need one additional statement about boundaried graphs and bucket interfaces.

Lemma 16. *Let $k, \ell \in \mathbb{N}$. Suppose $\mathbf{A} = (A, \bar{x})$ and $\mathbf{B} = (B, \bar{y})$ are two k-boundaried graphs, and suppose further that there is an immersion model (ϕ, ψ) of A in B such that $\phi(x_i) = y_i$, for all $i = 1, 2, \ldots, k$. Then for every (k, ℓ)-bucket interface (b, b', μ, μ^*), if $\mathbf{B}$ conforms to (b, b', μ, μ^*) then also $\mathbf{A}$ conforms to (b, b', μ, μ^*).*

Proof. First, we extend the immersion model (ϕ, ψ) to an immersion model (ϕ^*, ψ^*) of A^* in B^* by putting $\phi^*(x_i') = y_i'$ and $\psi^*(x_i x_i') = y_i y_i'$ for all $i \in [k]$. Suppose that ordering σ of $V(B^*)$ and its ℓ-bucketing T certify that $\mathbf{B}$ conforms to (b, b', μ, μ^*). We define ordering σ' of $V(A^*)$ and its ℓ-bucketing T' as follows:

- For $u, v \in V(A^*)$, we put $u <_{\sigma'} v$ if and only if $\phi^*(u) <_{\sigma'} \phi^*(v)$.
- For $u \in V(A^*)$, we put $T'(u) = T(\phi^*(u))$.

It is easy to see that T' is an ℓ-bucketing of σ'. We now verify that σ' and T' certify that $\mathbf{A}$ conforms to (b, b', μ, μ^*). The first two conditions of conforming follow directly from the definition of σ' and T', so we are left with the third and the fourth condition.

For the third condition, take any $j \in [\ell]$. It suffices to show that for any cut $(L', R') \in \texttt{cuts}(A, \sigma'|_A, T'|_A, j)$, we have that $|E_A(L', R')| \leq \mu(j)$. By the construction of (σ', T') it follows

that there is a cut $(L,R) \in \texttt{cuts}(B, \sigma|_B, T|_B, j)$ such that $\phi(L') \subseteq L$ and $\phi(R') \subseteq R$. Since (σ, T) certified that $\mathbf{B}$ conforms to (b, b', μ, μ^*), we have that $|E_B(L,R)| \leq \mu(j)$. Take any $uv \in E_A(L', R')$, and observe that $\psi(uv)$ is a path in B leading from $\phi(u) \in L$ to $\phi(v) \in R$. Consequently, one of the edges of this path must belong to $E_B(L,R)$. Since paths $\psi(uv)$ are pairwise edge-disjoint for different edges $uv \in E_A(L', R')$, we infer that

$$|E_A(L', R')| \leq |E_B(L,R)| \leq \mu(j).$$

This establishes the third condition. The fourth condition follows by the same argument applied to graphs A^* and B^*, instead of A and B. □

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 leanness 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 17. *Let k be a positive integer. If $G \in \mathbf{obs}_{\leq_{\mathrm{si}}}(\mathcal{C}_k)$, then $|V(G)| \leq N^{k+1}$, where $N = 2^{2((k+1)\log \ell + \ell \log(k+2))} + 2$ and $\ell = (2k+3) \cdot (2k+6)$.*

Proof. Take any $G \in \mathbf{obs}_{\leq_{si}}(\mathcal{C}_k)$ and assume, towards a contradiction, that $|V(G)| > N^{k+1}$. Let $\sigma = \langle v_1, v_2, \ldots, v_{|V(G)|} \rangle$ be a lean optimum cutwidth ordering of G, which exists by Lemma 13. We define $c_i = \delta(V^\sigma_{v_i})$, that is, c_i is the size of the cut between the vertices of G up to v_i and the rest of the graph. Notice that since $G \in \mathbf{obs}_{\leq_{si}}(\mathcal{C}_k)$, we have that $\mathbf{cw}(G) = k+1$ and G is connected. This implies that $c_i \in [k+1]$, for every $i \in [|V(G)| - 1]$.

Observe that $c_1 c_2 \ldots c_{|V(G)|-1}$ is a word of length at least N^{k+1} over the alphabet $[k+1]$. From Corollary 15, it follows that there exist $1 \leq s \leq t < |V(G)|$ and $q \in [k+1]$ such that for every $s \leq i \leq t$ we have $c_i \geq q$, and there also exist N distinct indices $s \leq i_1 < i_2 < \cdots < i_N \leq t$ such that $c_{i_j} = q$, for every $j \in [N]$. Without loss of generality we may assume that $i_1 = s$ and $i_N = t$.

For each $j \in [N]$, we can define a q-boundaried graph $\mathbf{G}_j = (G_j, (z^1_j, z^2_j, \ldots, z^q_j))$ in the following way. First, by leanness, we find edge-disjoint paths $P_1, \ldots, P_q$ between $V^\sigma_{v_{i_1}}$ and $V \setminus V^\sigma_{v_{i_N}}$. Notice that for each $j \in [N]$ the cut $E_G(V^\sigma_{v_{i_j}}, V(G) \setminus V^\sigma_{v_{i_j}})$ contains exactly one edge of each path P_i. Denote this edge by e^i_j, for $i \in [q]$. For $i \in [q]$, let x^i_j be the endpoint of e^i_j that belongs to $V^\sigma_{v_{i_j}}$, and let y^i_j be the endpoint that does not belong to $V^\sigma_{v_{i_j}}$. We construct G_j by taking $G[V^\sigma_{v_{i_j}}]$, adding fresh boundary vertices $(z^1_j, z^2_j, \ldots, z^q_j)$, and adding one fresh edge $x^i_j z^i_j$ for each $i \in [q]$.

Now consider any pair of indices $1 \leq j_1 < j_2 \leq N$. Observe that there exists an immersion model (ϕ, ψ) of $\mathbf{G}_{j_1}$ in $\mathbf{G}_{j_2}$ such that $\phi(z^i_{j_1}) = z^i_{j_2}$ for each $i \in [q]$. Indeed, we can put $\phi(u) = u$ for each $u \in V(G_{j_1})$ and $\phi(z^i_{j_1}) = z^i_{j_2}$ for each $i \in [q]$. Then ψ can be defined by taking $\psi(e) = e$ for each $e \in E(G_{j_1})$ and mapping each edge $x^i_{j_1} z^i_{j_1}$ to an appropriate infix of the path P_i, extended by the edge $x^i_{j_2} z^i_{j_2}$. Consequently, $\mathbf{G}_{j_1}$ and $\mathbf{G}_{j_2}$ satisfy the prerequisites of Lemma 16. We infer

that if by $\zeta(j)$ we denote the set of (q,ℓ)-bucket interfaces to which $\mathbf{G}_j$ conforms, then

$$\zeta(1) \supseteq \zeta(2) \supseteq \ldots \supseteq \zeta(N-1) \supseteq \zeta(N).$$

Observation 10 implies that N is larger by more than 1 than the total number of (q,ℓ)-bucket interfaces. It follows that there exists an index j, $1 \leq j < N$, such that

$$\zeta(j) = \zeta(j+1).$$

In other words, the q-boundaried graphs $\mathbf{G}_j$ and $\mathbf{G}_{j+1}$ are (q,ℓ)-similar.

Define a q-boundaried graph $\mathbf{G}' = (G', (y^1_{j+1}, \ldots, y^q_{j+1}))$ by taking $G' = G[V(G) \setminus V^\sigma_{i_{j+1}}]$. It can be now seen that $\mathbf{G}_{j+1} \oplus \mathbf{G}'$ is exactly the graph G with every edge of the cut $E_G(V^\sigma_{v_{i_j}}, V(G) \setminus V^\sigma_{v_{i_j}})$ subdivided once. Since subdividing edges does not change the cutwidth of the graph, we have that

$$\mathbf{cw}(\mathbf{G}_{j+1} \oplus \mathbf{G}') = \mathbf{cw}(G) > k. \tag{12}$$

On the other hand, q-boundaried graphs $\mathbf{G}_j$ and $\mathbf{G}_{j+1}$ are (q,ℓ)-similar. Since $\ell \geq (2q+3)\cdot(2q+6)$, by Theorem 11 we conclude that

$$\mathbf{cw}(\mathbf{G}_j \oplus \mathbf{G}') = \mathbf{cw}(\mathbf{G}_{j+1} \oplus \mathbf{G}'). \tag{13}$$

Examine the graph $\mathbf{G}_j \oplus \mathbf{G}'$. In the join operation, we added an edge $z^i_j y^i_{j+1}$ for each $i \in [q]$, which means each vertex z^i_j has exactly two incident edges in $\mathbf{G}_j \oplus \mathbf{G}'$: one connecting it to x^i_j and one connecting it to y^i_{j+1}. Let H be the graph obtained from $\mathbf{G}_j \oplus \mathbf{G}'$ by dissolving every vertex z^i_j, i.e., removing it and replacing edges $x^i_j z^i_j$ and $z^i_j y^i_{j+1}$ with a fresh edge $x^i_j y^i_{j+1}$. Subdividing edges does not change the cutwidth of a graph, so we obtain that:

$$\mathbf{cw}(H) = \mathbf{cw}(\mathbf{G}_j \oplus \mathbf{G}') \tag{14}$$

Finally, it is easy to see that G admits H as a strong immersion: a strong immersion model of H in G can be constructed by mapping the vertices and edges of G_j and G' identically, and then mapping each of the remaining edges $x^i_j y^i_{j+1}$ to a corresponding infix of the path P_i. Also, since $i_j < i_{j+1}$, the graph H has strictly less vertices than G. However, from Eq. (12), (13), and (14) we conclude that $\mathbf{cw}(H) = \mathbf{cw}(G) > k$. This contradicts the assumption that $G \in \mathbf{obs}_{\leq_{si}}(\mathcal{C}_k)$. □

Proof of Theorem 1. Theorem 17 provides an upper bound on the number of vertices of a graph in $\mathbf{obs}_{\leq_{si}}(\mathcal{C}_k)$. Observe that since such a graph has cutwidth $k+1$, each it its vertices has degree at most $2(k+1)$. It follows that any graph from $\mathbf{obs}_{\leq_{si}}(\mathcal{C}_k)$ has $2^{O(k^3 \log k)}$ vertices and edges. Finally, by Observation 3 we have $\mathbf{obs}_{\leq_i}(\mathcal{C}_q) \subseteq \mathbf{obs}_{\leq_{si}}(\mathcal{C}_q)$, 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 $k \leq r$ is given.

Our algorithm takes advantage of the given ordering σ and essentially computes, for each subgraph of G induced by a prefix of σ, the (r,ℓ)-bucket interfaces it conforms to. More precisely, in Lemma 18 we show that if G has an optimum ordering of width k, then there is an optimum ordering were *each* of these induced subgraphs occupies at most $\ell = O(rk)$ buckets, allowing to restrict our search to (r,ℓ)-bucket profiles (a variant of bucket interfaces to be defined later, refined so as to consider border vertices more precisely). The proof slightly strengthens that of Lemma 4.

Lemma 18. *Let G be a graph with an ordering σ of width r. Then there exists also an ordering τ of optimum width, i.e., with $\mathbf{cw}_\tau(G) = \mathbf{cw}(G)$, that has the following property: for every prefix X of σ, the number of X-blocks in τ is at most $2r \cdot \mathbf{cw}(G) + \mathbf{cw}(G) + 4r + 2$.*

Proof. Lemma 4 asserts that for each cut (A,B) of G of size at most r, there exists an optimum-width ordering of $V(G)$ where the number of (A,B)-blocks is at most

$$(2r+1)\cdot(2\mathbf{cw}(G)+3)+2r = 4r\cdot\mathbf{cw}(G)+2\mathbf{cw}(G)+8r+3.$$

As A-blocks and B-blocks appear alternately, at most half rounded up of the (A,B)-blocks can be A-blocks. Hence, the number of A-blocks in such an optimum-width ordering is at most $2r\cdot\mathbf{cw}(G)+\mathbf{cw}(G)+4r+2$; we denote this quantity by λ.

The proof of Lemma 4 in fact shows that for any ordering σ of $V(G)$ and any cut (A,B) of G of size at most r, either σ already has at most $2\lambda-1$ (A,B)-blocks, or an ordering σ' can be obtained from σ by swapping its (A,B)-blocks so that σ' has strictly less (A,B)-blocks. Therefore, by reordering (A,B)-blocks of σ, we eventually get a new ordering which has at most $2\lambda-1$ (A,B)-blocks, and hence at most λ A-blocks.

For $i = 1,2,\ldots,|V(G)|-1$, let (A_i,B_i) be the cut of G, where A_i is the prefix of σ of length i, while B_i is the suffix of σ of length $|V(G)|-i$. Let τ_0 be any optimum-width ordering of G. We now inductively construct orderings $\tau_1,\tau_2,\ldots,\tau_{|V(G)|-1}$, as follows: once τ_i is constructed, we apply the above reordering procedure to τ_i and cut (A_{i+1},B_{i+1}). This yields a new ordering τ_{i+1} of optimum width such that the number of A_{i+1}-blocks in τ_{i+1} is at most λ. Furthermore, τ_{i+1} is obtained from τ_i by reordering A_{i+1}- and B_{i+1}-blocks in τ_i. Hence, whenever X is a subset of A_{i+1}, then any X-block in τ_i remains consecutive in τ_{i+1}, as it is contained in one A_{i+1}-block in τ_i that is moved as a whole in the construction of τ_{i+1}. Consequently, if for all $j \le i$ we had that the number of A_j-blocks in τ_i is at most λ, then this property is also satisfied in τ_{i+1}. It is now clear that a straightforward induction yields the following invariant: for each $j \le i$, then number of A_j-blocks in τ_i is at most λ. Therefore $\tau = \tau_{|V(G)|-1}$ gives an ordering with the claimed properties. □

Bucket profiles. We now define a refinement of the widths of the buckets of a bucket interface as well as a refinement of the notion of bucket interfaces. They are used in the dynamic programming algorithm of Lemma 22.

Definition 19. *Let $(G,\bar{x})$ be a k-boundaried graph and let $S = \{x_1,\ldots,x_k,x'_1,\ldots,x'_k\} \subseteq V(G^*)$. Let now σ be an ordering of $V(G^*)$ and T be an ℓ-bucketing of σ. For every bucket $T^{-1}(i)$, $i \in [\ell]$,*

let $T^{-1}(i) \cap S = \{v_1, v_2, \ldots, v_p\}$ *for some* $v_1 <_\sigma v_2 <_\sigma \cdots <_\sigma v_p$*; we then define*

$$T_j^{-1}(i) = \begin{cases} \{v \in T^{-1}(i) \colon v <_\sigma v_1\} & \text{for } j = 0, \\ \{v \in T^{-1}(i) \colon v_j <_\sigma v <_\sigma v_{j+1}\} & \text{for } j \in [p-1], \\ \{v \in T^{-1}(i) \colon v_p <_\sigma v\} & \text{for } j = p. \end{cases}$$

Let also $\texttt{cuts}(G, \sigma, T, i, j)$ *be the family of* σ*-cuts containing on one side all vertices appearing before* v_{j-1} *(or, if* $j = 0$*, all vertices of buckets appearing before bucket* i*) and a prefix (in* σ*) of* $T_j^{-1}(i)$*. For an ordering* σ *of the vertices of a graph* G*, define the* width of j-th segment $T_j^{-1}(i)$ *of the bucket* i*,* $i \in [\ell]$*,* $j \in [0, p]$*, as the maximum width of any cut in the family* $\texttt{cuts}(G, \sigma, T, i, j)$*. Formally,*

$$\begin{aligned} \texttt{cuts}(G, \sigma, T, i, j) &= \big\{ \big(T^{-1}(\{1, \ldots, i-1\}) \cup L, T^{-1}(\{i+1, \ldots, \ell\}) \cup R\big) : \\ &\qquad (L, R) \text{ is a } \sigma\text{-cut of } T^{-1}(i) \text{ with } v_j \in L \text{ and } v_{j+1} \in R \big\}, \\ \texttt{width}(G, \sigma, T, i, j) &= \max \{\, |E_G(L, R)| \;:\; (L, R) \in \texttt{cuts}(G, \sigma, T, i, j) \,\}. \end{aligned}$$

We also need to refine the notion of a (k, ℓ)-bucket interface.

Definition 20. *For* $k, \ell \in \mathbb{N}$*, a* (k,ℓ)-bucket profile *consists of functions:*

- $b, b' : [k] \to [\ell]$ *identifying the buckets which contain* x_i *and* x'_i*, respectively,*
- $p, p' : [k] \to [k]$ *highlighting the ordering between the vertices* x_i *and* x'_i *inside a bucket, respectively,*
- $\nu : [\ell] \times [0, k] \to [0, k]$ *corresponding to the widths of segments of buckets defined by the vertices* x_i*, respectively.*

A k*-boundaried graph* $\mathbf{G}$ conforms *with a* (k, ℓ)*-bucket profile, if there exists an ordering* σ *of the vertices of* G^* *and an* ℓ*-bucketing* T *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* $i \in [k]$*,*
- $p(i) < p(j)$*, if* $b(i) = b(j)$ *and* $x_i <_\sigma x_j$*, and* $p'(i) < p'(j)$ *if* $b'(i) = b'(j)$ *and* $x'_i <_\sigma x'_j$*,*
- $\texttt{width}(G, \sigma|_G, T|_G, j, s) = \nu(j, s)$*, for each* $j \in [\ell]$ *and* $s \in [0, k]$*.*

From the fact that the boundary vertices of a k-boundaried graph $\mathbf{G}$ split the buckets defined by T into at most $2k$ segments in total it follows that:

Observation 21. *For any pair* (k, ℓ) *of positive integers, there is a set of at most*

$$2^{2k(\log \ell + \log k) + (\ell + 2k)\log(k+1)}$$

(k, ℓ)*-bucket profiles that a* k*-boundaried graph* $\mathbf{G}$ *can possibly conform with, and this set can be constructed in time polynomial in its size.*

The (k, ℓ)-bucket profiles that Observation 21 refers to will be called *valid*. By making use of these two notions we ensure that we will be able to update the widths of each bucket every time a new vertex is processed by the dynamic programming algorithm. We are now ready to prove Lemma 22.

Lemma 22. *Let $r \in \mathbb{N}^+$. Given a graph G and an ordering σ of its vertices with $\mathbf{cw}_\sigma(G) \leq r$, an ordering τ of the vertices of G with $\mathbf{cw}_\tau(G) = \mathbf{cw}(G)$ can be computed in time $2^{O(r^2 \log r)} \cdot |V(G)|$.*

Proof. The algorithm attempts to compute an ordering of width k for consecutive $k = 0, 1, 2, \ldots$. The first value of k for which the algorithms succeeds is equal to the value of the cutwidth, and then the constructed ordering may be returned. Since there is an ordering of width r, we will always eventually succeed for some $k \leq r$, which implies that we will make at most $r+1$ iterations. Hence, from now on we may assume that we know the target width $k \leq r$ for which we try to construct an ordering.

Given a graph G and an ordering σ of its vertices with $\mathbf{cw}_\sigma(G) \leq r$ we denote by G_w the graph induced by the vertices of the prefix of σ of length w. Then we naturally define the boundaried graph $\mathbf{G}_w$, where we introduce a boundary vertex x_i for each edge e_i of the cut $E_G(V(G_w), V(G) \setminus V(G_w))$. Note that this cut has at most r edges.

By Lemma 18, we know that there is an optimum-width ordering τ such that every prefix $V(G_w)$ of σ has at most ℓ blocks in τ. Our dynamic programming algorithm will simply inductively reconstruct all (k, ℓ)-bucket profiles that may correspond to $V(G_w)$-blocks in τ, for each consecutive w in the ordering σ, eventually reconstructing τ, if $\mathbf{cw}_\tau(G) \leq k$.

We now construct an auxiliary directed graph D that will model states and transitions of our dynamic programming algorithm. Let $\ell = 4rk + 2k + 8r + 4$. First, for every $w \in [0, |V(G)|]$ and every valid (k, ℓ)-bucket profile P, we add a vertex (w, P) to D. Thus, by Observation 21, the digraph D has at most

$$2^{2k(\log \ell + \log k) + (\ell + 2k)\log(k+1)} \cdot (|V(G)| + 1) = 2^{O(r^2 \log r)} \cdot |V(G)|$$

vertices. We add an edge $((w, P), (w+1, P'))$, whenever the (k, ℓ)-bucket profile P can be *expanded* to the (k, ℓ)-bucket profile P' in the sense that we explain now.

We describe which bucket profiles P' expand P by guessing where the new vertex would land in the bucket profile P, assuming that $\mathbf{G}_w$ conforms to P. After the guess is made, the updated profile P becomes the expanded profile P'. Different guesses lead to different profiles P' which extend P; this corresponds to different ways in which the construction of the optimum ordering can continue. As describing the details of this expansion relation is a routine task, we prefer to keep the description rather informal, and leave working out all the formal details to the reader.

Let v_{w+1} be the $(w+1)$-st vertex in the ordering σ, that is, $v_{w+1} \in V(G_{w+1}) \setminus V(G_w)$. We construct (by guessing) a (k, ℓ)-bucket profile P' from the (k, ℓ)-bucket profile P in the following way. First, we guess an even bucket of P to place each one of the vertices in $V(G^*_{w+1}) \setminus V(G^*_w)$: the new vertices of the extension that correspond to new edges of the cut $E_G(V(G_{w+1}), V(G) \setminus V(G_{w+1}))$ that are incident to v_{w+1}. Notice that each bucket contains, at any moment, at most r vertices. Therefore, we have at most $r+1$ possible choices about where each vertex will land in

each bucket (including the placing in the order, as indicated by the function $p'(\cdot)$. Notice also that there are at most $r+1$ vertices in $V(G^*_{w+1}) \setminus V(G^*_w)$. Therefore we have at most $(\ell(r+1))^{r+1}$ options for this guess.

Next, we choose the place v_{w+1} is going to be put in. If v_{w+1} is an endpoint of an edge from the cut $E_G(V(G_w), V(G) \setminus V(G_w))$, then this place is already indicated by functions $b'(\cdot)$ and $p'(\cdot)$ in the bucket profile P; if there are multiple edges in the cut $E_G(V(G_w), V(G) \setminus V(G_w))$ that have v_{w+1} as an endpoint, then all of them must be placed next to each other in the same even bucket (otherwise P has no extension). Otherwise, if v_{w+1} is not an endpoint of an edge from $E_G(V(G_w), V(G) \setminus V(G_w))$, we guess the placing of v_{w+1} by guessing an even bucket (one of at most $\ell+1$ options) together with a segment between two consecutive extension vertices in this bucket (one of at most $r+1$ options).

The placing of v_{w+1} may lead to one of three different scenarios; we again guess which one applies. First, v_{w+1} can establish a new odd bucket and split the even bucket into which it was put into two new even buckets, one on the left and one on the right of the new odd bucket containing v_{w+1}; the other extension vertices placed in this bucket are split accordingly. Second, v_{w+1} can be present at the leftmost or rightmost end of the even bucket it is placed in, so it gets merged into the neighboring odd bucket. Finally, if the even bucket in which v_{w+1} is placed did not contain any other extension vertices of G^*_w, then v_{w+1} can be declared to be the last vertex placed in this bucket, in which case we merge it together with both neighboring odd buckets. In these scenarios, whenever the extended profile turns out to have more than ℓ buckets, we discard this option.

Having guessed how the placing of v_{w+1} will affect the configuration of buckets, we proceed with updating the sizes of cuts, as indicated by function $\nu(\cdot)$. For this, we first examine all the edges of the cut $E_G(V(G_w), V(G) \setminus V(G_w))$ that have v_{w+1} as an endpoint. These edges did not contribute to the values of $\nu(\cdot)$ in the bucket profile P, but should contribute in P'. Note that given the placement of v_{w+1}, for each such edge we exactly see over which segments this edge "flies over", and therefore we can update the values of $\nu(\cdot)$ for these segments by incrementing them by one. Finally, when v_{w+1} got merged to a neighboring odd bucket (or to two of them), we may also need to take into account one more cut in the value of $\nu(\cdot)$ for the last/first segment of this bucket: the one between v_{w+1} and the vertices placed in this bucket. It is easy to see that from the value of $\nu(\cdot)$ for the segment in which v_{w+1} is placed, and the exact placement of the endpoints of all the boundary edges, we can deduce the exact size of this cut. Hence, the relevant value of $\nu(\cdot)$ can be efficiently updated by taking the maximum of the old value and the deduced size of the cut. We update the function ν in a similar fashion when v_{w+1} merges with both neighboring odd buckets. If at any point any of the values of $\nu(\cdot)$ exceeds k, we discard this guess.

This concludes the definition of the extension. For every (k,ℓ)-bucket profile P and every (k,ℓ)-bucket profile P' that extends it, we add to D an arc from (w,P) to $(w+1,P')$. It is easy to see from the description above that, given P and P', it can be verified in time polynomial in r whether such an arc should be added.

Finally, in the graph D we determine using, say, depth-first search, whether there is a directed

path from node $(0, P_\emptyset)$ to node $(|V(G)|, P_{\text{full}})$, where $P_\emptyset$ is an empty bucket profile and P_{full} is a bucket profile containing just one odd bucket. It is clear from the construction that if we find such a path, then by applying operations recorded along such a path we obtain an ordering of the vertices of G of width at most k. On the other hand, provided $k = \mathbf{cw}(G)$, by Lemma 18 we know that there is always an optimum-width ordering τ such that every prefix of σ has at most ℓ blocks in τ. Then the (k, ℓ)-bucket profiles naturally defined by the prefixes of σ in τ define a path from $(0, P_\emptyset)$ to $(|V(G)|, P_{\text{full}})$ in D.

The graph D has $2^{O(r^2 \log r)} \cdot |V(G)|$ vertices and arcs, and the depth-first search runs in time linear in its size. It is also trivial to reconstruct the optimum-width ordering of the vertices of G from the obtained path in linear time. This yields the promised running time bounds. □

Having the algorithm of Lemma 22, 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 22 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 formalised in Lemma 25. For its proof we we need the following definition and a known result about obstacles to small cutwidth.

Definition 23. *A* perfect binary tree *is a rooted binary tree in which all interior nodes have two children and all leaves have the same distance from its root. The* height *of a perfect binary tree is the distance between its root and one of its leaves.*

Lemma 24 ([10,13,18])**.** *If T is a perfect binary tree of height $2k$, then $\mathbf{cw}(T) \geq k$.*

Lemma 25. *There is an algorithm that given a positive integer k and a graph G, works in time $2^{O(k \log k)} \cdot |V(G)|$ and either concludes that $\mathbf{cw}(G) > k$, or finds a graph H immersed in G such that $|E(H)| \leq |E(G)| \cdot (1 - 1/(2k+1)^{4(k+1)+3})$ and $\mathbf{cw}(G) \leq 2\mathbf{cw}(H)$. Furthermore, in the latter case, given an ordering σ of the vertices of H, an ordering τ of the vertices of G with $\mathbf{cw}_\tau(G) \leq 2\mathbf{cw}_\sigma(H)$ can be computed in $O(|V(G)|)$ time.*

Proof. Without loss of generality we assume that G is connected, because we can apply the algorithm on the connected components of G separately and then take the disjoint union of the results.

Observe first that we may assume that every vertex in G is incident to at most $2k$ edges, as otherwise, we could immediately conclude that $\mathbf{cw}(G) > k$. This also implies that every vertex in G has at most $2k$ neighbors; by $N(v)$ we denote the set of neighbors of a vertex v, and $N(X) = (\bigcup_{v \in X} N(v)) \setminus X$ for a vertex subset X. Let G' be the graph obtained from G by exhaustively dissolving any vertices of degree 2 whose neighbors are different. That is, having such a vertex v, we delete it from the graph and replace the two edges incident to it with a fresh edge between its neighbors, and we proceed doing this as long as there are such vertices in the graph. Clearly, the eventually obtained graph G' can be immersed in G, we have $|E(G')| \leq |E(G)|$, the degree of every vertex in G' is the same to its degree in G, and $\mathbf{cw}(G') \leq \mathbf{cw}(G)$. However,

observe that any ordering of the vertices of G' can be turned into an ordering of the vertices of G with the same width by placing each dissolved vertex in any place between its two original neighbors. Thus, $\mathbf{cw}(G') = \mathbf{cw}(G)$.

Moreover, G' can be constructed in linear time by inspecting, in any order, all the vertices that have degree 2 in the original graph G. It is also easy to see that, given an ordering of vertices of G', one can reconstruct in linear time an ordering of G of at most the same width.

Altogether, it is now enough to either conclude that $\mathbf{cw}(G') > k$ or find a graph H immersed in G' such that

$$|E(H)| \leq |E(G')| \cdot (1 - 1/(2k+1)^{4(k+1)+2})$$

and $\mathbf{cw}(G') \leq 2\mathbf{cw}(H')$. Therefore, from now on we may assume that if the graph G' contains a vertex that is incident to two edges then this vertex is incident to an edge of multiplicity 2. Let V_1 be the set of vertices of degree 1 in G'. We consider two cases depending on the size of V_1.

Case 1. $|V_1| \geq |E(G')|/(2k+1)^{4(k+1)+2}$. Notice first that $V_1 \subseteq N(N(V_1))$, and recall that every vertex in G' is incident to at most $2k$ edges and therefore has at most $2k$ neighbors. It follows then that $|V_1| \leq 2k \cdot |N(V_1)|$ and hence $|N(V_1)| \geq |E(G')|/(2k+1)^{4(k+1)+3}$. Let H be the graph obtained from G' by removing, for each vertex in $N(V_1)$, one of its neighbors in V_1. Then $|E(H)| \leq |E(G')| \cdot (1-1/(2k+1)^{4(k+1)+3})$ and H is immersed in G' (as it is an induced subgraph). Hence, H is also immersed in G. Furthermore, let σ be any ordering of the vertices of H. Then, we can obtain an ordering of the vertices of G' by placing each deleted vertex next to its original neighbors. Notice that this placement increases the width of σ by at most 1 in total, and thus by a multiplicative factor of at most 2. As we already showed how to obtain an ordering of $V(G)$ from a given ordering of $V(G')$, the lemma follows for the case where $|V_1| \geq |E(G')|/(2k+1)^{4(k+1)+2}$.

Case 2. $|V_1| < |E(G')|/(2k+1)^{4(k+1)+2}$. For every $v \in V(G')$ and every positive integer s, we define $B_s(v)$ to be the ball of radius s around v, that is, the set of vertices at distance at most s from v in G'. Recall that every vertex of G' has at most $2k$ neighbors and observe then that $|B_s(v)| \leq (2k+1)^s$. We construct a set of vertices $v_1, v_2, \ldots, v_\ell \in V(G')$ whose pairwise distance is greater than $4(k+1)$ in the following greedy way. Having chosen $v_1, \ldots, v_i$, if $B_{4(k+1)}(v_1) \cup \cdots \cup B_{4(k+1)}(v_i) \neq V(G')$ then let v_{i+1} be any vertex outside of $B_{4(k+1)}(v_1) \cup \cdots \cup B_{4(k+1)}(v_i)$. If such a vertex does not exist, we stop by putting $\ell = i$ and consider the set $v_1, v_2, \ldots, v_\ell$. Observe here that we can calculate $B_{4(k+1)}(v_i)$ by breadth-first search in $O((2k+1)^{4(k+1)+1})$ time, by stopping the search at depth $4(k+1)$. Hence, sequence $v_1, \ldots, v_\ell$ can be computed in $2^{O(k \log k)} \cdot |V(G)|$ time by examining vertices one by one in any order, and marking those already covered by some ball. When the next vertex is uncovered, we set it as the next v_{i+1}, run the breadth-first search from it, and mark all vertices in the ball of radius $4(k+1)$ around it.

We now estimate the length ℓ of the sequence. Recall that for every $i \in [\ell]$, $|B_{4(k+1)}(v_i)| \leq (2k+1)^{4(k+1)}$ and that $V(G) = \bigcup_{i \in [\ell]} B_{4(k+1)}(v_i)$. From the above and the fact that $|E(G')| \leq 2k \cdot |V(G')|$ (as every vertex of G' is incident to at most $2k$ edges of G'), it follows that

$$\ell \geq |V(G')|/(2k+1)^{4(k+1)} \geq |E(G')|/(2k+1)^{4(k+1)+1}.$$

By construction, the distance between v_i and v_j is greater than $4(k+1)$, for distinct $i, j \in [\ell]$. Therefore, the balls $B_{2(k+1)}(v_1), \ldots, B_{2(k+1)}(v_\ell)$ are vertex-disjoint. Moreover, since we have that $|V_1| < |E(G')|/(2k+1)^{4(k+1)+2}$, at most $|E(G')|/(2k+1)^{4(k+1)+2}$ of those balls contain a vertex of degree 1. Therefore, the remaining $\ell - |E(G')|/(2k+1)^{4(k+1)+2}$ balls are disjoint with V_1. Let $I \subseteq [\ell]$ be the set of indices for which the balls $B_{2(k+1)}(v_i)$, $i \in I$, are disjoint from V_1. Observe that

$$|I| \geq \ell - |E(G')|/(2k+1)^{4(k+1)+2} \geq |E(G')|/(2k+1)^{4(k+1)+2}.$$

Claim 26. *In time $O(|E(G')|)$ we can either conclude that $\mathbf{cw}(G') > k$, or for each $i \in I$ find a cycle in G' passing only through the vertices of the ball $B_{2(k+1)}(v_i)$.*

Proof. Suppose for some $i \in I$, $B_{2(k+1)}(v_i)$ does not contain a cycle. We will prove that every vertex in $G'[B_{2(k+1)}(v_i)]$ has degree at least 3 in G', and that every edge appears with multiplicity 1. Notice first that every edge of the graph $G'[B_{2(k+1)}(v_i)]$ has multiplicity 1, as otherwise an edge with multiplicity at least 2 would form a cycle, a contradiction. Notice also that $B_{2(k+1)}(v_i)$ does not have any vertex that has degree 2 in G. Indeed, recall that by the construction of the graph G' any vertex of degree 2 is incident only to one edge of multiplicity 2, which is again a contradiction. Moreover, by the choice of $i \in [I]$, we obtain that $B_{2(k+1)}(v_i) \cap V_1 = \emptyset$ and therefore, $G'[B_{2(k+1)}(v_i)]$ does not have any vertex that has degree 1 in G. We conclude that every vertex in $G'[B_{2(k+1)}(v_i)]$ has degree at least 3 in G, and every edge appears with multiplicity 1. Recall that the subgraph of G' induced by $B_{2(k+1)}(v_i)$ contains the full breadth-first search tree of vertices at distance at most $2(k+1)$ from v_i. If $G'[B_{2(k+1)}(v_i)]$ did not contain any cycle, then it would be equal to this breadth-first search tree, and in this tree all vertices except possibly the last layer would have degrees at least 3. Hence, G' would contain as a subgraph a perfect binary tree of height $2(k+1)$. From Lemma 24, this tree has cutwidth at least $k+1$. The algorithm can thus check (by breadth-first search) for a cycle in the subgraph induced by $B_{2(k+1)}(v_i)$. If it does not find any such cycle it immediately concludes that $\mathbf{cw}(G) = \mathbf{cw}(G') > k$.

If for every $i \in I$, the breadth-first search in $G'[B_{2(k+1)}(v_i)]$ finds a cycle, then the algorithm obtained, in total time $O(|E(G')|)$, a set of at least $|I| \geq E(G')/(2(k+1))^{4(k+1)+2}$ vertex-disjoint (and hence edge-disjoint) cycles. ⌟

Let us assume that the algorithm has now found a set $\mathcal{C}$ of at least $E(G')/(2k+1)^{4(k+1)+2}$ edge-disjoint cycles and let H be the subgraph obtained from G' by removing one, arbitrarily chosen, edge e_C from each cycle $C \in \mathcal{C}$. Then H can be immersed in G' and $|E(H)| \leq |E(G')| \cdot (1 - 1/(2k+1)^{4(k+1)+2})$. To complete the proof of the lemma we will prove that if σ is any ordering of the vertices of H then σ is also an ordering of the vertices of G' such that $\mathbf{cw}_\sigma(G') \leq 2\mathbf{cw}_\sigma(H)$. Notice that by reintroducing an edge e_C of G' to H we increase the width of the σ-cuts separating its endpoints by exactly 1. Observe also that since e_C belongs to the cycle C, the rest of the cycle forms a path P_C in H that connects the endpoints of e_C. Therefore, each of the σ-cuts separating the endpoints of e_C has to contain at least one edge of P_C. Since for different edges e_C, for $C \in \mathcal{C}$, the corresponding paths P_C are pairwise edge-disjoint and they are present in H, it follows that the size of each σ-cut in G' is at most twice the size of this σ-cut in H. Therefore $\mathbf{cw}_\sigma(G') \leq 2\mathbf{cw}_\sigma(H)$. Thus, H can be returned, concluding the algorithm. □

We are now ready to put all the pieces together.

Proof of 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 25, 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 25 again, to an ordering of G of width $\leq 2k$. Given this ordering, we apply the dynamic programming procedure of Lemma 22 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:

$$\begin{aligned} \sum_{i=0}^{\infty} 2^{O(k^2 \log k)} \cdot |E(G_i)| &\leq 2^{O(k^2 \log k)} \cdot |E(G)| \cdot \sum_{i=0}^{\infty} (1 - 1/(2k+1)^{4k+7})^i \\ &= 2^{O(k^2 \log k)} \cdot |E(G)| \cdot (2k+1)^{4k+7} \qquad (15) \\ &= 2^{O(k^2 \log k)} \cdot |E(G)|. \end{aligned}$$

□

6 Obstructions to edge-removal distance to cutwidth

Throughout this section, by $O_k(w)$ we mean a quantity bounded by $c_k \cdot w + d_k$, for some constant c_k, d_k depending on k only.

Given a graph G and a $k \in \mathbb{N}$, we define the parameter $\mathbf{dcw}_k(G)$ as the minimum number of edges that can be deleted from G so that the resulting graph has cutwidth at most k. In other words:

$$\mathbf{dcw}_k(G) = \min\{|F| \colon F \subseteq E(G) \text{ and } \mathbf{cw}(G \setminus F) \leq k\}$$

Let $\mathcal{C}_{w,k} = \{G \mid \mathbf{dcw}_k(G) \leq w\}$. Notice that $\mathcal{C}_k = \mathcal{C}_{0,k}$.

In this section, we provide bounds to the sizes of the obstruction sets of the class of graphs G with $\mathbf{dcw}_k(G) \leq w$, for each $k, w \in \mathbb{N}$. Our results are the following.

Theorem 27. *For every $w, k \in \mathbb{N}$, every graph in $\mathbf{obs}_{\leq_{\mathrm{si}}}(\mathcal{C}_{w,k})$ has $O_k(w)$ vertices.*

Theorem 28. *For every $k, w \in \mathbb{N}$ where $k \geq 7$, the set $\mathbf{obs}_{\leq_{\mathrm{i}}}(\mathcal{C}_{w,k})$ contains at least $\binom{3^{k-7}+w+1}{w+1}$ non-isomorphic graphs.*

From Observation 3, both bounds of Theorems 27 and 28 holds for both immersions and strong immersions as well.

Given a collection $\mathcal{H}$ of graphs, we define the parameter $\mathbf{aic}_{\mathcal{H}}(G)$ as the minimum number of edges whose removal from G creates an $\mathcal{H}$-immersion free graph, that is, a graph that does not

admit any graph from $\mathcal{H}$ as an immersion. In both subsections that follow, we need the following observation.

Observation 29. *For every graph G and every $w \in \mathbb{N}$, it holds that* $\mathbf{dcw}_w(G) = \mathbf{aic}_{\mathrm{obs}(\mathcal{C}_w)}(G)$.

We remark that, within the same set of authors, we have recently studied kernelization algorithms for edge removal problems to immersion-closed classes. The following result has been obtained in a yet unpublished manuscript [9]: Whenever a finite collection of graphs $\mathcal{H}$ contains at least one planar subcubic graph, and all graphs from $\mathcal{H}$ are connected, then the problem of computing $\mathbf{aic}_{\mathcal{H}}(G)$, parameterized by the target value, admits a linear kernel. These prerequisites are satisfied for $\mathcal{H} = \mathrm{obs}(\mathcal{C}_w)$, and hence the problem of computing $\mathbf{dcw}_w(G)$, parameterized by the target value k, admits a linear kernel.

The connections between kernelization procedures and upper bounds on minimal obstruction sizes have already been noticed in the literature; see e.g. [7]. Intuitively, whenever the kernelization rules apply only minor or immersion operations, the kernelization algorithm can be turned into a proof of an upper bound on the sizes of minimal obstacles for the corresponding order. Unfortunately, this is not the case for the results of [9]: the main problem is the lack of lean decompositions for parameter tree-cut width, which plays the central role. Here, the situation is different, as we know that there are always lean orderings of optimum width. We therefore showcase how to use the leanness to obtain a linear upper bound on the sizes of obstructions for $\mathcal{C}_{w,k}$. The arguments are somewhat similar as in [9]: we use the idea of protrusions, adapted to the setting of edge cuts, and we try to replace protrusions with smaller ones having the same behavior. The main point is that leanness ensures us that the replacement results in an immersion of the original graph.

6.1 Upper bound

A *partial q-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 that consists either from vertices of G or from empty slots (that is indices that to not correspond to vertices of G). If x_i is an empty slot, we denote it by $x_i = \diamond$. The extension of such $\mathbf{G}$ is defined just as for q-boundaried graphs, but we put $x_i = \diamond$ iff $x'_i = \diamond'$. Given a q-boundaried graph $\mathbf{F} = (F, \bar{x})$ we denote by $\mathcal{P}(F)$ the set containing every partial q-boundaried graph $\mathbf{F}' = (F', \bar{x}')$ such that F' is a subgraph of F and a vertex x_i in $\bar{x}'$ is an empty slot iff $x_i \in V(F) \setminus V(F')$. Intuitively a partial q-boundaried graph extends the notion of a boundaried graph by demanding that the vertices in their boundary carry indices from a set whose cardinality might be bigger than the boundary.

Let H be a graph and let (X_1, X_2) be its cut where $q = \delta(X_1)$. Let $E_H(X_1, X_2) = \{e_1, \ldots, e_q\}$ where $e_i = \{x_i^1, x_i^2\}$, $i \in [q]$, and such that $x_i^j \in X_j$ for $(i, j) \in [q] \times [2]$. For $j \in [2]$, we say that the pair (X_1, X_2) *generates* the q-boundaried graph $\mathbf{A}_j = (A_j, \overline{x}_j)$ if $A_i = G[X_i]$ and $\overline{x}_i = (x_1^j, \ldots, x_q^j)$.

We denote by $\mathcal{B}_{q,h}$ the collection containing every q-boundaried graph that can be generated from some cut (X_1, X_2) of some graph H where $|V(H)| + |E(H)| \leq h$ and $q = \delta(X_1)$. Moreover, we denote by $\mathcal{M}_{q,h} = \mathcal{P}(\mathcal{B}_{q,h})$. In other words, $\mathcal{M}_{q,h}$ contains all partial q-boundaried graphs

that can be generated by a graph whose number of edges and vertices does not exceed h. We insist that if $\mathbf{H} = (H, \overline{x}) \in \mathcal{M}_{q,h}$, then the vertices of H are taken from some fixed repository of h vertices and that an element x_i of $\overline{x}$ is either an empty slot (i.e., $x_i = \diamond$) or the i-th vertex of some predetermined ordering $(x_1, \ldots, x_q)$ of q vertices from this repository. This permits us to assume that $|\mathcal{M}_{q,h}|$ is bounded by some function that depends only on q and h.

Let $\mathbf{G} = (G, \overline{x})$ be a q-boundaried graph and $\mathbf{H} = (H, \overline{y})$ be a partial q-boundaried graph. Let also G^* and H^* be the extensions of G and H, respectively. We also assume that, for all $i \in [q]$, either $y_i = x_i$ or $y_i = \diamond$. For an edge subset $R \subseteq E(G^*)$, we say that $\mathbf{H}$ is an *R-avoiding strong immersion of* $\mathbf{G}$ if there is an H^*-immersion model (ϕ, ψ) of $G^* \setminus R$ where, for every $i \in [q]$ such that $y_i \neq \diamond$, it holds that $\phi(y_i) = x_i \neq \diamond$ and $\phi(y_i') = x_i' \neq \diamond$. We now define the *$R$-avoiding (q,h)-folio* of $\mathbf{G}$ as the set of all partial q-boundaried graphs in $\mathcal{M}_{q,h}$ that are R-avoiding strong immersions of $\mathbf{G}$ and we denote it by $\mathbf{folio}_{q,h,R}(\mathbf{G})$. We finally define

$$\mathcal{F}_{q,h}(\mathbf{G}) = \{\mathbf{folio}_{q,h,R}(\mathbf{G}) \mid R \subseteq E(G^*)\}.$$

Given two q-boundaried graphs $\mathbf{G}_1$ and $\mathbf{G}_2$ we write $\mathbf{G}_1 \sim_{q,h} \mathbf{G}_2$ in order to denote that $\mathcal{F}_{q,h}(\mathbf{G}_2) = \mathcal{F}_{q,h}(\mathbf{G}_2)$. As $\mathcal{F}_{q,h}$ maps each q-boundaried graph to a collection of subsets of $\mathcal{M}_{q,h}$ we have the following.

Lemma 30. *There is some function $f_1 : \mathbb{N}^2 \to \mathbb{N}$ such that for every two non-negative integers h and r, the number of equivalence classes of $\sim_{q,h}$ is at most $f_1(q,h)$.*

The next lemma is a consequence of the definition of the function $\mathcal{F}_{q,h}$.

Lemma 31. *Let $\mathcal{H}$ be some set of connected graphs, each of at most h vertices, and let $\mathbf{G}_i = (G_i, \overline{x}_i), i \in \{1,2\}$ be two q-boundaried graphs such that $\mathbf{G}_1 \sim_{q,h} \mathbf{G}_2$ and such both G_1, G_2 are $\mathcal{H}$-immersion free. Then, for every q-boundaried graph $\mathbf{F} = (F, \overline{y})$, it holds that $\mathbf{aic}_{\mathcal{H}}(\mathbf{F} \oplus \mathbf{G}_1) = \mathbf{aic}_{\mathcal{H}}(\mathbf{F} \oplus \mathbf{G}_2)$.*

The proof is omitted as it is very similar to the one in [5] where a similar encoding was defined in order to treat the topological minor relation. To see the main idea, recall that $\mathcal{F}_{q,h}(\mathbf{G}_i)$ registers all different "partial occurrences" of graphs of $\leq h$ vertices (and therefore also of graphs of $\mathcal{H}$) in $\mathbf{G}_i'$, for all possible ways to obtain $\mathbf{G}_i'$ from $\mathbf{G}$ after removing at most q edges. This encoding is indeed sufficient to capture the behavior of all possible edge sets whose removal from $\mathbf{F} \oplus \mathbf{G}_i$ creates an $\mathcal{H}$-free graph. Indeed, as both G_1, G_2 are $\mathcal{H}$-immersion free, any such set should have at most q edges inside $\mathbf{G}_i$ as, if not, the q-boundary edges between $\mathbf{F}$ and $\mathbf{G}_i$ would also make the same job. A similar discussion is also present in [9].

Given a graph G and $X \subseteq V(G)$, we write $\mathbf{cw}_\sigma(G, X) = \delta_G(X) + \mathbf{cw}_{\sigma_X}(G[X])$. We require the following extension of the definition of lean orderings.

Definition 32 (extended lean ordering)**.** *Let G be a n-vertex graph and $X \subseteq V(G)$. An ordering $\sigma = \langle v_1, \ldots, v_n \rangle$ of G is* X-lean *if $X = \{v_{n-|X|+1}, \ldots, v_n\}$ and for every $i, j \in [n - |X|, n]$ where $i < j$ there exist $\min\{\delta(\{v_1, \ldots, v_h\}) \mid i \leq_\sigma h \leq_\sigma j\}$ edge-disjoint paths between $\{v_1, \ldots, v_i\}$ and $\{v_j, \ldots, v_n\}$ in G.*

The proof of the following result is very similar to the one of Lemma 13. We just move X to the end of the ordering, in the order given by σ, and apply exhaustively the same refinement step based on submodularity, but only to the subordering induced by X.

Lemma 33. *For every graph G and every subset X of $V(G)$ there exists an ordering σ of G such that $\mathbf{cw}_\sigma(G,X) \leq r$, then there exist an X-lean ordering σ' of G such that $\mathbf{cw}_{\sigma'}(G,X) \leq r$.*

Let $w_1, w_2 \in \mathbb{N}$, G be a graph, and $X \subseteq V(G)$. We say that X is an (w_1,w_2)-*cutwidth-edge-protrusion* of G if $\delta(X) \leq w_1$ and $\mathbf{cw}(G[X_i]) \leq w_2$.

The next lemma uses an idea similar to the one of Lemma 17. Here $\sim_{q,h}$ plays the role of (q,ℓ)-similarity.

Lemma 34. *There is a computable function $f_2 : \mathbb{N}^2 \to \mathbb{N}$ such that the following holds: Let k be a non-negative integer and let $\mathcal{H}$ be a finite set of connected graphs, each having at most h vertices and edges. Let also G be a graph and let X be a $(2k,k)$-cutwidth-edge-protrusion of G. If $|X| > f_2(k,h)$, then G contains as a proper strong immersion a graph G' where $\mathbf{aic}_\mathcal{H}(G) = \mathbf{aic}_\mathcal{H}(G')$.*

Proof. We set $f_2(k,h) = (f_1(3k,h)+1)^{3k+1} - 1$. We have that $|X| > f_2(k,h)$ or, equivalently, $|X| \geq (f_1(3k,h)+1)^{3k+1}$. Denote $\ell = |X|$. Let $\sigma^* = \langle x_1, \ldots, x_\ell \rangle$ be an ordering of the vertices in X such that $\mathbf{cw}_{\sigma^*}(G[X]) \leq k$. Let $\sigma = \langle v_1, \ldots, v_{n-\ell}, v_{n-\ell+1}, \ldots, v_n \rangle$ be any ordering of $V(G)$ such that σ' is a suffix of σ, i.e. $\langle x_1, \ldots, x_\ell \rangle = \langle v_{n-\ell+1}, \ldots, v_n \rangle$. It follows that $\mathbf{cw}_\sigma(G,X) \leq \mathbf{cw}_{\sigma^*}(G[X]) + \delta_G(X) \leq k + 2k = 3k$. From Lemma 33, there is an X-lean ordering σ' of $V(G)$, where $\mathbf{cw}_{\sigma'}(G,X) \leq 3k$.

We set $k_i = \delta_{G[X]}(\{x_1, \ldots, x_{i-(n-\ell)}) + \delta_G(X)$ and observe that $k_i \leq k+2k = 3k, i \in [n-\ell, n-1]$. We set up the alphabet $\mathbb{A} = \{0, 1, \ldots, 3k\}$ and we see $w = k_{n-\ell}, k_{n-\ell+1}, \ldots, k_{n-1}$ as a word on $\mathbb{A}$. Let also $N = f_1(3k,h)$. Notice that $|w| = |X| = \ell \geq (f_1(3k,h)+1)^{3k+1} = (N+1)^{|\mathbb{A}|}$. From Corollary 15, if $|w| \geq (N+1)^{|\mathbb{A}|}$, there are $a, b \in [n-\ell, n-1], a < b$ and some $p \in \mathbb{A}$ such that $k_a, k_b \geq p$ and p appears in $\{k_a, \ldots, k_b\}$ at least $N+1$ times. Let these appearance be at indices $a \leq i_1 < i_2 < \ldots < i_{N+1} \leq b$.

By X-leanness, there are p edge-disjoint paths P^i, for $i \in [p]$, from $\{v_1, \ldots, v_{i_1}\}$ to $\{v_{i_{N+1}}, \ldots, v_{|V(G)|}\}$. Observe that for each $j \in [N+1]$, each path P^i must cross exactly one edge of the cut $\delta_G(\{v_1, \ldots, v_{i_j}\})$; let this edge be $z^i_j w^i_j$, where $z^i_j \in \{v_1, \ldots, v_{i_j}\}$ and $w^i_j \notin \{v_1, \ldots, v_{i_j}\}$. For each $j \in [N+1]$ we define p-boundaried graphs $\mathbf{F}_j = (F_j, (z^1_j, \ldots, z^p_j))$, where $F_j = G[\{v_1, \ldots, v_{i_j}\}]$, and $\mathbf{G}_j = (G_j, (w^1_j, \ldots, w^p_j))$ where $G_j = G[\{v_{i_j+1}, \ldots, v_{|V(G)|}\}]$.

As, from Lemma 30, the equivalence relation $\sim_{3k,h}$ has at most N equivalent classes, there are j_1, j_2 such that $a \leq j_1 < j_2 \leq b$ such that $\mathbf{G}_{j_1} \sim_{3k,h} \mathbf{G}_{j_2}$. Let $G' = \mathbf{F}_{i_{j_1}} \oplus \mathbf{G}_{i_{j_2}}$; it is easy to observe that G' is a proper immersion of G, because the edges added when joining can be modeled using appropriate infixes of the paths P_i. From Lemma 31, however, we have $\mathbf{aic}_\mathcal{H}(\mathbf{F}_i \oplus \mathbf{G}_i) = \mathbf{aic}_\mathcal{H}(\mathbf{F}_i \oplus \mathbf{G}_j)$, and therefore $\mathbf{aic}_\mathcal{H}(G) = \mathbf{aic}_\mathcal{H}(G')$. □

Lemma 35. *Let $k, w, \ell \in \mathbb{N}$ and let G be a graph. If $\mathbf{dcw}_k(G) \leq w$ and $|V(G)| \geq \ell \cdot (w+1) + 2w$, then G has a $(2k,k)$-cutwidth-edge-protrusion X where $|X| \geq \ell$.*

Proof. We denote $n = |V(G)|$. Let F be a set of edges of G such that if $G' = G \setminus F$, then $\mathbf{cw}(G') \leq k$. Let $\sigma = \langle v_1, \ldots, v_n \rangle$ be an ordering of $V(G)$ such that $\mathbf{cw}_\sigma(G) \leq k$. Let also S be the set of endpoints of the edges in F. Clearly $|S| \leq 2w$. We define a set of indices $I \subseteq \{1, \ldots, n\}$ such that $\{v_i \mid i \in I\} = S$. A pair $(i, j) \in I^2$ is *consecutive* if there is no $h \in I$ such that $i < h < j$. Among all consecutive pairs in I^2 let (i, j) be one maximizing the value $c = j - i - 1$. This implies that $c \geq (n - 2w)/(w + 1) \geq \ell$. Consider now the set $X = \{v_{i+1}, \ldots, v_{j-1}\}$ and observe that $|X| = c \geq \ell$. Notice that if $\sigma' = \langle v_{i+1}, \ldots, v_{j-1} \rangle$, then $\mathbf{cw}_{\sigma'}(G[X]) \leq k$. Moreover there are at most $\delta_{G'}(\{v_1, \ldots, v_i\}) + \delta_{G'}(\{v_j, \ldots, v_n\}) \leq 2k$ edges with one vertex in X and the other not in X. Therefore, $\delta_G(X) \leq 2k$ and X is a $(2k, k)$-cutwidth-edge-protrusion of G. □

Proof of Theorem 27. We set $\mathcal{H} = \mathbf{obs}_{\leq_{\mathrm{im}}}(\mathcal{C}_k)$. By Theorem 17, there is a function $f_3 : \mathbb{N} \to \mathbb{N}$ such that graphs from $\mathcal{H}$ have at most $h = f_3(k)$ vertices. Let $G \in \mathbf{obs}_{\leq_{\mathrm{im}}}(\mathcal{C}_{w,k})$. This means that $\mathbf{dcw}_k(G) = w + 1$, while, for every proper strong immersion G' of G, it holds that $\mathbf{dcw}_k(G') \leq w$. This, together with Observation 29 and Lemma 34, implies that G cannot have a $(2k, k)$-cutwidth-edge-protrusion X of more than $\ell = f_2(k, h)$, vertices. As $\mathbf{dcw}_k(G) = w + 1$, Lemma 35 implies that $|V(G)| < \ell \cdot (w + 2) + 2w + 2 = O_k(w)$ vertices. □

6.2 Lower bound

We now focus on the proof of Theorem 28. We need the following result.

Theorem 36 ([10])**.** *For every $k \geq 7$, the number of non-isomorphic connected minimal obstructions in $\mathbf{obs}_{\leq_i}(\mathcal{C}_k)$ is at least $3^{k-7} + 1$.*

Recall that, given a graph class $\mathcal{H}$, we defined $\mathbf{aic}_\mathcal{H}(G)$ as the minimum number of edges of G whose removal yields an $\mathcal{H}$-immersion-free graph. We set $\mathcal{C}_{w,\mathcal{H}} = \{G \mid \mathbf{aic}_\mathcal{H}(G) \leq w\}$. In particular $\mathcal{C}_{0,\mathcal{H}}$ is the class of all $\mathcal{H}$-immersion free graphs. If G and H are graphs, we denote by $G \uplus H$ the disjoint union of G and H.

The following observations follow directly from the definition of $\mathbf{aic}_\mathcal{H}$.

Observation 37. *If G and H are graphs, then $H \leq_i G$ implies that $\mathbf{aic}_\mathcal{H}(H) \leq \mathbf{aic}_\mathcal{H}(G)$.*

Observation 38. *If G and H are graphs, then $\mathbf{aic}_\mathcal{H}(G \uplus H) = \mathbf{aic}_\mathcal{H}(G) + \mathbf{aic}_\mathcal{H}(H)$.*

Observation 39. *If $G \in \mathbf{obs}_{\leq_i}(\mathcal{C}_{w,\mathcal{H}})$, then $\mathbf{aic}_\mathcal{H}(G) = w + 1$.*

Lemma 40. *Let $\mathcal{H}$ be some $\leq_i$-antichain. For every non-negative integer w, if $G_1, \ldots, G_{w+1}$ are (not necessarily distinct) members of $\mathcal{H}$, then $\biguplus_{i=1}^{w+1} G_i \in \mathbf{obs}_{\leq_i}(\mathcal{C}_{w,\mathcal{H}})$.*

Proof. Let $G = \biguplus_{i=1}^{w+1} G_i$. To prove that $G \in \mathbf{obs}_{\leq_i}(\mathcal{C}_{w,\mathcal{H}})$ we have to show that it satisfies **O1** and **O2**. Notice that since $\mathcal{H}$ is an $\leq_i$-antichain, $\mathbf{aic}_\mathcal{H}(H) = 1$ for every $H \in \mathcal{H}$. By Observations 38 and 39, $\mathbf{aic}_\mathcal{H}(G) = \sum_{i=1}^{w+1} \mathbf{aic}_\mathcal{H}(G_i) = w + 1$ and **O1** holds. Therefore, $G \notin \mathcal{C}_{w,\mathcal{H}}$. Let now G' is a proper immersion of G. This mean that $G' = \biguplus_{i=1}^{w+1} G'_i$ where $G'_i \leq_i G_i$ and at least one of $G'_1, \ldots, G'_{w+1}$ is different than G_i. W.l.o.g. we assume that this graph is G_{w+1}. As $\mathcal{H}$ is a $\leq_i$-antichain, G'_{w+1} is different than any of the graphs in $\mathcal{H}$. Therefore $\mathbf{aic}_\mathcal{H}(G'_{w+1}) = 0$. Then, by Observations 37 and 38, $\mathbf{aic}_\mathcal{H}(G') = \sum_{i=1}^{k} \mathbf{aic}_\mathcal{H}(G'_i) + \mathbf{aic}_\mathcal{H}(G'_{w+1}) \leq \sum_{i=1}^{w} \mathbf{aic}_\mathcal{H}(G_i) + 0 = w$ and **O2** holds. □

Theorem 41. *If k is a non-negative integer and $\mathcal{H}$ is a $\leq_i$-antichain that contains at least q connected graphs, then $|\mathbf{obs}_{\leq_i}(\mathcal{C}_{w,\mathcal{H}})| \geq \binom{q+w}{w+1}$.*

Proof. Let $\mathcal{H}'$ be some subset of $\mathcal{H}$ containing q connected graphs. Using Lemma 40, we observe that every multiset of cardinality $w+1$ whose elements belong to $\mathcal{H}'$ corresponds to a different (i.e. non-isomorphic) obstructions of $\mathcal{C}_{w,\mathcal{H}}$. Therefore, $|\mathbf{obs}_{\leq_i}(\mathcal{C}_{w,\mathcal{H}})$— is at least the number of multisets of cardinality $w+1$ the elements of which are taken from a set of cardinality q, which is known to be $\binom{q+w}{w+1}$. □

Proof of Theorem 27. From Observation 29, $\mathcal{C}_{w,k} = \mathcal{C}_{w,\mathcal{H}_k}$, where $\mathcal{H}_k = \mathbf{obs}_{\leq_i}(\mathcal{C}_k)$. This means that $\mathbf{obs}_{\leq_i}(\mathcal{C}_{w,k}) = \mathbf{obs}_{\leq_i}(\mathcal{C}_{w,\mathcal{H}_k})$. The result follows from Theorems 36 and 41. □

7 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.

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 lean: 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 [22] and carving-width [17]. For tree-cut decompositions and carving decompositions, one could define leanness in a very similar manner. For example for carving-width, the definition would be as follows: Suppose $(\mathcal{T}, \tau)$ is a carving decomposition of a graph G, where τ bijectively maps vertices of G to leaves of $\mathcal{T}$. Then $(\mathcal{T}, \tau)$ is considered *lean* if for every two disjoint subtrees $\mathcal{S}_1$ and $\mathcal{S}_2$ of $\mathcal{T}$, respectively rooted at nodes x_1 and x_2, the maximum number of edge-disjoint paths leading from $\tau^{-1}(\mathcal{S}_1)$ to $\tau^{-1}(\mathcal{S}_2)$ is equal to

the minimum cutsize among the edges of the path in $\mathcal{T}$ from x_1 to x_2. The definition for tree-cut width is very similar.

We conjecture that both for tree-cut width and carving-width, there is always an optimum decomposition that is lean; i.e., an analogue of Lemma 13 holds. Interestingly, when one tries to mimic the proof of Lemma 13, the refinement operation can be generalized without much effort. The problem lies in finding the right potential function for showing that the refinement cannot be applied indefinitely. If the conjectured result would be true for tree-cut width or carving-width, it is conceivable that an approach similar to the one of this paper would give upper bounds on the sizes of minimal immersion obstructions also for these parameters.

7.1 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.

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] Dimitris Chatzidimitriou, Jean-Florent Raymond, Ignasi Sau, and Dimitrios M. Thilikos. An $O(\log OPT)$ -approximation for covering/packing minor models of θ_r. In *Proceedings of WAOA 2015*, pages 122–132, 2015. Full version available at arXiv:1510.03945.

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

[7] 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.

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

[9] Archontia C. Giannopoulou, Michał Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Linear kernels for edge deletion problems to immersion-closed graph classes, 2016. Manuscript.

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

[11] 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.

[12] 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.

[13] Ephraim Korach and Nir Solel. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, 43(1):97 – 101, 1993.

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

[15] 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.

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

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

[18] Atsushi Takahashi, Shuichi Ueno, and Yoji Kajitani. Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discrete Math.*, 127(1-3):293–304, 1994. Graph theory and applications (Hakone, 1990).

[19] 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.

[20] 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.

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

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

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