



HAL
open science

Linear Kernels for Edge Deletion Problems to Immersion-Closed Graph Classes

Dimitrios M. Thilikos, Archontia Giannopoulou, Michal Pilipczuk,
Jean-Florent Raymond, Marcin Wrochna

► **To cite this version:**

Dimitrios M. Thilikos, Archontia Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Marcin Wrochna. Linear Kernels for Edge Deletion Problems to Immersion-Closed Graph Classes. SIAM Journal on Discrete Mathematics, 2021, 35 (1), pp.105-151. 10.1137/18M1228839 . hal-03327286

HAL Id: hal-03327286

<https://hal.science/hal-03327286>

Submitted on 22 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Linear kernels for edge deletion problems to immersion-closed graph classes

Archontia C. Giannopoulou^{a,b} Michał Pilipczuk^{c,d} Jean-Florent Raymond^{c,e,f}
Dimitrios M. Thilikos^{e,g} Marcin Wrochna^{c,d}

August 8, 2018

Abstract

Suppose \mathcal{F} is a finite family of graphs. We consider the following meta-problem, called \mathcal{F} -IMMERSION DELETION: given a graph G and integer k , decide whether the deletion of at most k edges of G can result in a graph that does not contain any graph from \mathcal{F} as an immersion. This problem is a close relative of the \mathcal{F} -MINOR DELETION problem studied by Fomin et al. [FOCS 2012], where one deletes vertices in order to remove all minor models of graphs from \mathcal{F} . We prove that whenever all graphs from \mathcal{F} are connected and at least one graph of \mathcal{F} is planar and subcubic, then the \mathcal{F} -IMMERSION DELETION problem admits:

- a constant-factor approximation algorithm running in time $\mathcal{O}(m^3 \cdot n^3 \cdot \log m)$;
- a linear kernel that can be computed in time $\mathcal{O}(m^4 \cdot n^3 \cdot \log m)$; and
- a $\mathcal{O}(2^{\mathcal{O}(k)} + m^4 \cdot n^3 \cdot \log m)$ -time fixed-parameter algorithm,

where n, m count the vertices and edges of the input graph. These results mirror the findings of Fomin et al. [FOCS 2012], who obtained a similar set of algorithmic results for \mathcal{F} -MINOR DELETION, under the assumption that at least one graph from \mathcal{F} is planar. An important difference is that we are able to obtain a *linear* kernel for \mathcal{F} -IMMERSION DELETION, while the exponent of the kernel of Fomin et al. for \mathcal{F} -MINOR DELETION depends heavily on the family \mathcal{F} . In fact, this dependence is unavoidable under plausible complexity assumptions, as proven by Giannopoulou et al. [ICALP 2015]. This reveals that the kernelization complexity of \mathcal{F} -IMMERSION DELETION is quite different than that of \mathcal{F} -MINOR DELETION.

1 Introduction

On the \mathcal{F} -MINOR DELETION problem. Given an class of graphs \mathcal{G} , we denote by $\mathbf{obs}_{\text{mn}}(\mathcal{G})$ the *minor-obstruction set* of \mathcal{G} , that is the set of minor-minimal graphs that do not belong in \mathcal{G} . Let us fix some finite family of graphs \mathcal{F} . A graph G is called *\mathcal{F} -minor-free* if G does not contain any graph from \mathcal{F} as a minor. The celebrated Graph Minors Theorem of Robertson and Seymour [38]

^aTechnische Universität Berlin, Berlin, Germany.

^bThe research of this author 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) and by the Warsaw Center of Mathematics and Computer Science.

^cInstitute of Informatics, University of Warsaw, Poland.

^dSupported by the Polish National Science Center grant SONATA DEC-2013/11/D/ST6/03073.

^eAlGCo project team, CNRS, LIRMM, Montpellier, France.

^fSupported by the Polish National Science Centre grant PRELUDIUM DEC-2013/11/N/ST6/02706.

^gDepartment of Mathematics, National and Kapodistrian University of Athens, Greece.

implies that for every family of graphs Π that is closed under taking minors, the set $\mathcal{F}_G = \mathbf{obs}_{\text{mn}}(\mathcal{G})$ is finite. In other words, \mathcal{G} is characterized by the minor-exclusion of finite set of graphs; that is \mathcal{G} is exactly the class of \mathcal{F}_G -minor-free graphs. Hence, studying the classes of \mathcal{F} -minor-free graphs for finite families \mathcal{F} is the same as studying general minor-closed properties of graphs.

Fomin et al. [19] performed an in-depth study of the following parameterized¹ problem, named \mathcal{F} -MINOR DELETION²: *Given a graph G and an integer parameter k , decide whether it is possible to remove at most k vertices from G to obtain an \mathcal{F} -minor-free graph.* By considering different families \mathcal{F} , the \mathcal{F} -MINOR DELETION problem generalizes a number of concrete problems of prime importance in parameterized complexity, such as VERTEX COVER, FEEDBACK VERTEX SET, or PLANARIZATION. It is easy to see that, for every fixed k , the graph class $\mathcal{G}_{k,\mathcal{F}}^{\text{mn}}$, consisting of the graphs in the YES-instances (G, k) of \mathcal{F} -MINOR DELETION, is closed under taking of minors. We define $\mathcal{O}_k^{\text{mn}} = \mathbf{obs}_{\text{mn}}(\mathcal{G}_{k,\mathcal{F}}^{\text{mn}})$. By the fact that $\mathcal{O}_k^{\text{mn}}$ is finite and the meta-algorithmic consequences of the Graph Minors series of Robertson and Seymour [36, 38], it follows (non-constructively) that \mathcal{F} -MINOR DELETION admits an FPT-algorithm. The optimization of the running time of such FPT-algorithms for several instantiations of \mathcal{F} has been an interesting project in parameterized algorithm design and so far it has been focused on problems generated by minor-closed graph classes.

The goal of Fomin et al. [19] was to obtain results of general nature for \mathcal{F} -MINOR DELETION, which would explain why many concrete problems captured as its subcases are efficiently solvable using parameterized algorithms and kernelization. This has been achieved under the assumption that \mathcal{F} contains at least one planar graph. More precisely, for any class \mathcal{F} that contains at least one planar graph, the work of Fomin et al. [19] gives the following:

- (i) a randomized constant-factor approximation running in time $\mathcal{O}(nm)$;
- (ii) a polynomial kernel for the problem; that is, a polynomial-time algorithm that, given an instance (G, k) of \mathcal{F} -MINOR DELETION, outputs an equivalent instance (G', k') with $k' \leq k$ and $|G'| \leq \mathcal{O}(k^c)$, for some constant c that depends on \mathcal{F} ;
- (iii) an FPT-algorithm solving \mathcal{F} -MINOR DELETION in time $2^{\mathcal{O}(k)} \cdot n^2$.
- (iv) a proof that every graph in $\mathcal{O}_k^{\text{mn}}$ has $k^{c_{\mathcal{F}}}$ vertices for some constant $c_{\mathcal{F}}$ that depends (non-constructively) on \mathcal{F} .

We remark that, for the FPT-algorithm, the original paper of Fomin et al. [19] needs one more technical assumption, namely that all the graphs from \mathcal{F} are connected. The fact that this condition can be lifted was proved in a subsequent work of Kim et al. [30].

The assumption that \mathcal{F} contains at least one planar graph is crucial for the approach of Fomin et al. [19]. Namely, from the Excluded Grid Minor Theorem of Robertson and Seymour [37] it follows that for such families \mathcal{F} , \mathcal{F} -minor-free graphs have treewidth bounded by a constant depending only of \mathcal{F} . Therefore, a YES-instance of \mathcal{F} -MINOR DELETION roughly has to look like a constant-treewidth graph plus k additional vertices that can have arbitrary connections. Having exposed this structure, Fomin et al. [19] apply protrusion-based techniques that originate in the work on *meta-kernelization* [4, 20]. Roughly speaking, the idea is to identify large parts of the graphs that have constant treewidth and a small interface towards the rest of the graph (so-called

¹A *parameterized* problem can be seen as a subset of $\Sigma^* \times \mathbb{N}$ where its instances are pairs $(x, k) \in \Sigma^* \times \mathbb{N}$. For graph problems, the string x usually encodes a graph G . A parameterized problem admits an FPT-algorithm, or, equivalently, belongs in the parameterized complexity class FPT, if it can be solved by an $f(k) \cdot |x|^{O(1)}$ step algorithm. See [13, 17, 35] for more on parameterized algorithms and complexity.

²Fomin et al. use the name \mathcal{F} -DELETION, but we choose to use the word “minor” explicitly to distinguish it from immersion-related problems that we consider in this paper.

protrusions), which can be replaced by smaller gadgets with the same combinatorial behaviour. Such preprocessing based on *protrusion replacement* is the base of all three aforementioned results for \mathcal{F} -MINOR DELETION. In the absence of a constant bound on the treewidth of an \mathcal{F} -minor-free graph, the technique breaks completely. In fact, the kernelization complexity of PLANARIZATION, that is, \mathcal{F} -MINOR DELETION for $\mathcal{F} = \{K_5, K_{3,3}\}$, is a notorious open problem.

An interesting aspect of the work of Fomin et al. [19] is that the exponent of the polynomial bound on the size of the kernel for \mathcal{F} -MINOR DELETION grows quite rapidly with the family \mathcal{F} . Recently, it has been shown by Giannopoulou et al. [23] that in general this growth is probably unavoidable: For every constant η , the TREEWIDTH- η DELETION problem (delete k vertices to obtain a graph of treewidth at most η) has no kernel with $\mathcal{O}(k^{\eta/4-\epsilon})$ vertices for any $\epsilon > 0$, unless $\text{NP} \subseteq \text{coNP/poly}$. Since graphs of treewidth η can be characterized by a finite set of forbidden minors \mathcal{F}_η , at least one of which is planar, this refutes the hypothesis that all \mathcal{F} -MINOR DELETION problems admit polynomial kernels with a uniform bound on the degree of the polynomial. However, as shown by Giannopoulou et al. [23], such *uniform kernelization* can be achieved for some specific problem families, like vertex deletion to graphs of constant tree-depth.

Immersion problems. Recall that a graph H can be *immersed* into a graph G (or that H is an *immersion* of G) if there is a mapping from H to G that maps vertices of H to pairwise different vertices of G and edges of H to pairwise *edge-disjoint paths* connecting the images of the respective endpoints³. Such a mapping is called an *immersion model*. Just like the minor relation, the immersion relation imposes a partial order on the class of graphs. Alongside with the minor order, Robertson and Seymour [39] proved that graphs are also well-quasi-ordered under the immersion order, i.e., every set of graphs that are pair-wise non-comparable with respect to the immersion relation is finite. This implies that for every class of graphs \mathcal{G} that is closed under taking immersions the set $\mathbf{obs}_{\text{im}}(\mathcal{G})$, containing the immersion minimal graphs that do not belong in \mathcal{G} , is finite (we call $\mathbf{obs}_{\text{im}}\mathcal{G}$ *immersion obstruction set* of \mathcal{G}). Therefore \mathcal{G} can be characterized by a finite set of forbidden immersions. The general intuition is that immersion is a containment relation on graphs that corresponds to *edge cuts*, whereas the minor relation corresponds to *vertex cuts*. Also, the natural setting for immersions is the setting of *multigraphs*. Hence, from now on all the graphs considered in this paper may have parallel edges connecting the same pair of endpoints.

Recently, there has been a growing interest in immersion-related problems [2, 6, 14–16, 22, 24–26, 29, 33, 42] both from the combinatorial and the algorithmic point of view. Most importantly for us, Wollan proved in [42] an analog of the Excluded Grid Minor Theorem, which relates the size of the largest wall graph that is contained in a graph as an immersion with a new graph parameter called *tree-cut width*. By a *subcubic graph* we mean a graph of maximum degree at most 3. The following theorem follows from the work of Wollan [42] and summarizes the conclusions of this work that are important for us.

Theorem 1 ([24]). *For every graph H that is planar and subcubic there exists a constant a_H , such that every graph that does not contain H as an immersion has tree-cut width bounded by a_H .*

In other words, for any family \mathcal{F} of graphs that contains some planar subcubic graph, the tree-cut width of \mathcal{F} -immersion-free graphs is bounded by a universal constant depending on \mathcal{F} only. In Section 2 we discuss the precise definition of tree-cut width and how exactly Theorem 1 follows from the work of Wollan [42]. Also, note that if a family of graphs \mathcal{F} does not contain any planar subcubic graph, then there is no uniform bound on the tree-cut width of \mathcal{F} -immersion-free

³In this paper we consider *weak immersions* only, as opposed to *strong immersions* where the paths are forbidden to traverse images of vertices other than the endpoints of the corresponding edge.

graphs. Indeed, wall graphs are then \mathcal{F} -immersion-free, because all their immersions are planar and subcubic, and they have unbounded tree-cut width.

After the introduction of tree-cut width by Wollan [42], the new parameter gathered substantial interest from the algorithmic and combinatorial community [22, 24, 29, 33]. It seems that tree-cut width serves the same role for immersion-related problems as treewidth serves for minor-related problems and, in a sense, it can be seen as an “edge-analog” of treewidth. In particular, given the tree-cut width bound of Theorem 1 and the general approach of Fomin et al. [19] to \mathcal{F} -MINOR DELETION, it is natural to ask whether the same kind of results can be obtained for immersions where the considered modification is edge removal instead of vertex removal. More precisely, fix a finite family of graphs \mathcal{F} containing some planar subcubic graph and consider the following \mathcal{F} -IMMERSION DELETION problem: given a graph G and an integer k , determine whether it is possible to delete at most k edges of G in order to obtain a graph that does not admit any graph from \mathcal{F} as an immersion.

Parallel to the case of \mathcal{F} -MINOR DELETION, for every fixed k , the graph class $\mathcal{G}_{k,\mathcal{F}}^{\text{im}}$ consisting of the graphs in the YES-instances (G, k) of \mathcal{F} -IMMERSION DELETION is closed under taking of immersions⁴, therefore $\mathcal{O}_k^{\text{im}} = \text{obs}_{\text{im}}(\mathcal{G}_{k,\mathcal{F}}^{\text{im}})$ is a finite set, by the well-quasi-ordering of graphs under immersions [39]. Together with the immersion-testing algorithm of Grohe et al. [27], this implies that \mathcal{F} -IMMERSION DELETION admits (non-constructively) an FPT-algorithm. This naturally induces the parallel project of optimizing the performance of such FPT-algorithms for several instantiations of \mathcal{F} . More concretely, is it possible to extend the general framework of Fomin et al. [19] to obtain efficient approximation, kernelization, and FPT algorithms also for \mathcal{F} -IMMERSION DELETION? Theorem 1 suggests that the suitable analog of the assumption from the minor setting that \mathcal{F} contains a planar graph should be the assumption that at least one graph from \mathcal{F} is planar and subcubic.

Our results. In this work we give a definitive positive answer to this question. The following two theorems gather our main results; for a graph G , by $|G|$ and $\|G\|$ we denote the cardinalities of the vertex and edge sets of G , respectively.

Theorem 2 (Constant factor approximation). *Suppose \mathcal{F} is a finite family of connected graphs and at least one member of \mathcal{F} is planar and subcubic. Then there exists an algorithm that, given a graph G , runs in time $\mathcal{O}(\|G\|^3 \log \|G\| \cdot |G|^3)$ and outputs a subset of edges $F \subseteq E(G)$ such that $G - F$ is \mathcal{F} -immersion-free and the size of F is at most c_{apx} times larger than the optimum size of a subset of edges with this property, for some constant c_{apx} depending on \mathcal{F} only.*

In Section 8 (Conclusions) we comment on how the constant-factor approximation can be generalized to work for \mathcal{F} containing disconnected graphs as well, using the approach of Fomin et al. [18, 19].

Theorem 3 (Linear kernelization and obstructions). *Suppose \mathcal{F} is a finite family of connected graphs and at least one member of \mathcal{F} is planar and subcubic. Then there exists an algorithm that, given an instance (G, k) of \mathcal{F} -IMMERSION DELETION, runs in time $\mathcal{O}(\|G\|^4 \log \|G\| \cdot |G|^3)$ and outputs an equivalent instance (G', k) with $\|G'\| \leq c_{\text{ker}} \cdot k$, for some constant c_{ker} depending on \mathcal{F} only. Moreover, there exists a constant $c_{\mathcal{F}}$ (non-constructively depending on \mathcal{F}) such that every graph H in $\mathcal{O}_k^{\text{im}}$ has at most $c_{\mathcal{F}} \cdot k$ edges.*

⁴Notice that if we consider deletion of vertices instead of edges, then the graph class $\mathcal{G}_k^{\text{im}}$ is *not* closed under taking immersions (for example, in a star on 7 vertices with duplicated edges, deleting one vertex makes it K_3 -immersion-free, but this ‘duplicated’ star immerses $2K_3$, which has no such vertex). This is the main reason why we believe that *edge* deletion gives a more suitable counterpart to \mathcal{F} -MINOR DELETION for the case of immersions.

Thus, Theorems 2 and 3 mirror the approximation and kernelization results and the obstruction bounds of Fomin et al. [19]. However, this mirroring is not exact as we even show that, in the immersion setting, a stronger kernelization procedure can be designed. Namely, the size of the kernel given by Theorem 3 is *linear*, with only the multiplicative constant depending on the family \mathcal{F} , whereas in the minor setting, the exponent of the polynomial bound on the kernel size provably must depend on \mathcal{F} (under plausible complexity assumptions). This shows that the immersion and minor settings behave quite differently and in fact stronger results can be obtained in the immersion setting. Observe that using Theorem 3 it is trivial to obtain a decision algorithm for \mathcal{F} -IMMERSION DELETION working in time $\mathcal{O}(c_{\text{fpt}}^k + \|G\|^4 \log \|G\| \cdot |G|^3)$ for some constant c_{fpt} depending on \mathcal{F} only: one simply computes the kernel with a linear number of edges and checks all the subsets of edges of size k .

Our techniques. Our approach to proving Theorems 2 and 3 roughly follows the general framework of protrusion replacement of Fomin et al. [19] (see also [4, 5]). We first define protrusions suited for the problem of our interest. In fact, our protrusions can be seen as the edge-analog of those introduced in [19] (as in [7]). A protrusion for us is simply a vertex subset X that induces an \mathcal{F} -immersion-free subgraph (which hence has constant tree-cut width, by Theorem 1), and has a constant number of edges to the rest of the graph. When a large protrusion is localized, it can be replaced by a smaller gadget similarly as in the work of Fomin et al. [19]. However, we need to design a new algorithm for searching for large protrusions, mostly in order to meet the condition that the exponent of the polynomial running time of the algorithm *does not depend* on \mathcal{F} . For this, we employ the important cuts technique of Marx [32] and the randomized contractions technique of Chitnis et al. [9]. All of these yield an algorithm that exhaustively reduces all large protrusions.

Unfortunately, exhaustive protrusion replacement is still not sufficient for a linear kernel. However, we prove that in the absence of large reducible protrusions, the only remaining obstacles are large groups of parallel edges between the same two endpoints (called *thetas*), and, more generally, large “bouquets” of constant-size graphs attached to the same pair of vertices. Without these, the graph is already bounded linearly in terms of the optimum solution size. The approximation algorithm can thus delete *all* edges except for the copies included in bouquets and thetas, reducing the optimum solution size by a constant fraction of the deleted set. It then exhaustively reduces protrusions in the remaining edges, and repeats the process until the graph is \mathcal{F} -immersion-free.

To obtain a linear kernel we need more work, as we do not know how to reduce bouquets and thetas directly. Instead, we apply the following strategy based on the idea of *amortization*. After reducing exhaustively all larger protrusions, we compute a constant-factor approximate solution F_{apx} . Then we analyze the structure of the graph $G - F_{\text{apx}}$, which has constant tree-cut width. It appears that every bouquet and theta in G can be reduced up to size bounded linearly in the number of solution edges F_{apx} that “affect” it. After applying this reduction, we can still have large bouquets and thetas in the graph, but this happens only when they are affected by a large number of edges of F_{apx} . However, every edge of F_{apx} can affect only a constant number of bouquets and thetas and hence a simple amortization arguments shows that the total size of bouquets and theta is linear in $|F_{\text{apx}}|$, so also linear in terms of the optimum.

We remark that this part of the reasoning and in particular the amortization argument explained above, are fully new contributions of this work. These arguments deviate significantly from those needed by Fomin et al. [19], because they were aiming at a weaker goal of obtaining a polynomial kernel, instead of linear. Also, we remark that, contrary to the work of Fomin et al. [19], all our algorithms are deterministic.

For the second part of Theorem 3, we show that protrusions replacements can be done in

a way that the resulting graph is an immersion of the original one. This implies that, in the equivalent instance (G', k) produced by our kernelization algorithm, the graph G' is an immersion of G . Therefore if G is an immersion-obstruction of $\mathcal{G}_{k-1, \mathcal{F}}^{\text{im}}$, then it should already have a linear, on k , number of edges (see Section 7).

Application: immersion-closed parameters. Before we proceed to the proofs of Theorems 2 and 3, we would like to highlight one particular meta-algorithmic application of our results which was our original motivation. Suppose \mathbf{p} is a graph parameter, that is, a function that maps graphs to nonnegative integers. We shall say that \mathbf{p} is *closed under immersion* if whenever a graph H is an immersion of another graph G , then $\mathbf{p}(H) \leq \mathbf{p}(G)$. Furthermore, \mathbf{p} is *closed under disjoint union* if $\mathbf{p}(G_1 \uplus G_2) = \max(\mathbf{p}(G_1), \mathbf{p}(G_2))$, for any two graphs G_1 and G_2 ; here, \uplus denotes the disjoint union of two graphs. Finally, \mathbf{p} is *large on walls* if the set of integers $\{\mathbf{p}(W_{n,n})\}_{n \in \mathbb{N}}$ is infinite, where $W_{n,n}$ is the $n \times n$ wall, depicted in Figure 1, for $n = 4$. The following proposition follows easily from Theorem 1 and the fact that the immersion order is a well-quasi-order.

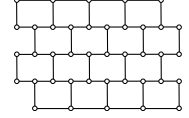


Figure 1: $W_{4,4}$.

Proposition 4. *Let \mathbf{p} be a graph parameter that is closed under immersion and under disjoint union and moreover is large on walls. Then for every $r \in \mathbb{N}$ there exists a finite family of graphs $\mathcal{F}_{\mathbf{p},r}$ with the following properties:*

- (a) every graph from $\mathcal{F}_{\mathbf{p},r}$ is connected;
- (b) $\mathcal{F}_{\mathbf{p},r}$ contains at least one planar subcubic graph; and
- (c) for every graph G , we have that $\mathbf{p}(G) \leq r$ if and only if G is $\mathcal{F}_{\mathbf{p},r}$ -immersion-free.

Proof. Denote by $\mathcal{G}_{\mathbf{p},r}$ the class of all graphs G for which $\mathbf{p}(G) \leq r$. Since \mathbf{p} is immersion-closed, $\mathcal{G}_{\mathbf{p},r}$ is closed under taking immersions. Since the immersion order is a well-quasi-order on graphs, we infer that there is a finite family $\mathcal{F}_{\mathbf{p},r}$ of graphs such that a graph G belongs to $\mathcal{G}_{\mathbf{p},r}$ if and only if G is $\mathcal{F}_{\mathbf{p},r}$ -immersion-free. Moreover, we can assume that $\mathcal{F}_{\mathbf{p},r}$ is minimal in the following sense: for each $H \in \mathcal{F}_{\mathbf{p},r}$ and each H' that can be immersed in H and is not isomorphic to H , we have that $H' \in \mathcal{G}_{\mathbf{p},r}$; equivalently, $\mathbf{p}(H') \leq r$. We need to argue that every member of $\mathcal{F}_{\mathbf{p},r}$ is connected and that $\mathcal{F}_{\mathbf{p},r}$ contains a planar subcubic graph.

For the first check, suppose that there is some disconnected graph H in $\mathcal{F}_{\mathbf{p},r}$. Then H has two proper subgraphs H_1 and H_2 such that $H = H_1 \uplus H_2$. Since H_1, H_2 are subgraphs of H , they can, in particular, be immersed in H . Both of them are strictly smaller than H , so we infer that $\mathbf{p}(H_1) \leq r$ and $\mathbf{p}(H_2) \leq r$. As \mathbf{p} is closed under disjoint union, we infer that $\mathbf{p}(H) = \mathbf{p}(H_1 \uplus H_2) = \max(\mathbf{p}(H_1), \mathbf{p}(H_2)) \leq r$. This is a contradiction to the fact that $H \notin \mathcal{G}_{\mathbf{p},r}$.

For the second check, since \mathbf{p} is unbounded on walls, there is some integer n such that $\mathbf{p}(W_{n,n}) > r$. Consequently, $W_{n,n} \notin \mathcal{G}_{\mathbf{p},r}$, so $W_{n,n}$ contains some graph H from $\mathcal{F}_{\mathbf{p},r}$ as an immersion. It can be easily seen that planar subcubic graphs are closed under taking immersions, so since $W_{n,n}$ is planar and subcubic, we infer that H is also planar and subcubic. \square

For a parameter \mathbf{p} and a constant r , define the \mathbf{p} -AT-MOST- r EDGE DELETION problem as follows: given a graph G and an integer k , determine whether at most k edges can be deleted from G to obtain a graph with the value of \mathbf{p} at most r . We also define the associated parameter \mathbf{p}_r such that

$$\mathbf{p}_r(G) = \min\{k \mid \exists S \subseteq E(G) : |S| \leq k \wedge \mathbf{p}(G \setminus S) \leq r\}.$$

We also define $\mathcal{G}_{k,\mathbf{p}_r} = \{G \mid \mathbf{p}_r(G) \leq k\}$. By combining Proposition 4 with Theorems 2 and 3 we obtain the following corollary.

Corollary 5. *Let \mathbf{p} be a graph parameter that is closed under immersion and under disjoint union and moreover is large on the class of walls⁵. Then, for every constant r , the \mathbf{p} -AT-MOST- r EDGE DELETION problem admits a constant-factor approximation and a linear kernel. Moreover, there is a constant c_r , depending (non-constructively) on r , such that for every k , every graph H in $\mathbf{obs}_{\text{im}}(\mathcal{G}_{k,\mathbf{p}_r})$ has at most $c_r \cdot k$ edges.*

Natural immersion-closed parameters that satisfy the prerequisites of Corollary 5 include cutwidth, carving width, tree-cut width, and edge ranking; see e.g. [28, 31, 40–42] for more details on these parameters. Corollary 5 mirrors the corollary given by Fomin et al. [19] for the TREEWIDTH- η DELETION problem, for which their results imply the existence of a constant-factor approximation, a polynomial-kernel, a polynomial bound for the corresponding minor-obstruction set $\mathbf{obs}_{\text{mn}}(\mathcal{G}_{k,\text{tw}_\eta})$, and a single-exponential FPT algorithm, for every constant η .

Organization of the paper. In Section 2 we introduce notation, recall known definitions and facts, and prove some easy observations of general usage. In Section 3 we provide several adjustments of the notions of tree-cut decompositions and tree-cut width. In particular, we provide a simpler definition of tree-cut width that we use throughout the paper, and show that an optimum-width tree-cut decomposition may be assumed to have some additional, useful properties. In Section 4 we discuss protrusions: finding them and replacing them. Section 5 contains the proof of Theorem 2 (constant factor approximation), while Section 6 contains the proof of Theorem 3 (linear kernelization). Section 7 is dedicated to the linear bound on the size of obstructions. We conclude with some finishing remarks in Section 8.

2 Preliminaries

For a positive integer p , we denote $[p] = \{1, 2, \dots, p\}$.

Graphs. In this work, all graphs are multigraphs without loops. That is, a graph G is a pair $(V(G), E(G))$, where $V(G)$ is the *vertex set* and $E(G)$ is a multiset of *edges*. An *edge* connects a pair of different vertices, called *endpoints*; we write $uv \in E(G)$ for an edge with endpoints $u, v \in V(G)$. Note that there might be several edges (called *parallel edges*) between two vertices. An edge is *incident* to a vertex if that vertex is one of its two endpoints.

We write $|G|$ for $|V(G)|$ and $\|G\|$ for $|E(G)|$ (counting edges with multiplicities). For a subset of vertices $X \subseteq V(G)$, $G[X]$ is the subgraph induced by X . For a subset $X \subseteq V(G)$ of vertices, we write $G - X$ for the induced subgraph $G[V(G) \setminus X]$. For a subset $F \subseteq E(G)$ of edges, we write $G - F$ for the graph obtained from G by removing all edges of F , with $V(G - F) = V(G)$ and $E(G - F) = E(G) \setminus F$.

For two subsets $X, Y \subseteq V(G)$, not necessarily disjoint, $E_G(X, Y)$ denotes the set of all edges $xy \in E(G)$ for which $x \in X$ and $y \in Y$. The *boundary* of X is $\delta_G(X) = E_G(X, V(G) \setminus X)$, while the set of edges *incident* to X is $E_G(X, V(G))$. For $v \in V(G)$, the *degree* of v is $\deg_G(v) = |\delta_G(\{v\})|$. We also define the set of neighbors of v : $N_G(v) = \{u \in V(G) : uv \in E(G)\}$. By $N_G(X)$ we denote the open neighborhood of X , that is, the set of all vertices outside X that have a neighbor in X . We drop the subscript G when it is clear from the context. A graph is *subcubic* if $\deg_G(v) \leq 3$ for every $v \in V(G)$.

⁵A graph parameter \mathbf{p} is *large on a graph class \mathcal{C}* if $\{\mathbf{p}(G) \mid G \in \mathcal{C}\}$ is not a bounded set.

Trees. A forest is a graph where every connected component is a tree. For a forest T and an edge $uv \in E(T)$, we denote by T_{uv} and T_{vu} the components of $T - uv$ containing u and v , respectively. Let T be a rooted tree. For every node $t \in V(T)$, we denote by $\pi(t)$ its unique parent on the tree T . A node t' is a *sibling* of t if $t \neq t'$ and t and t' have the same parent.

Tree-cut width. A *near-partition* of a set X is a family of (possibly empty) subsets X_1, \dots, X_k of X such that $\bigcup_{i=1}^k X_i = X$ and $X_i \cap X_j = \emptyset$ for every $i \neq j$.

A *tree-cut decomposition* of a graph G is a pair $\mathcal{T} = (T, \mathcal{X})$ such that T is a forest and $\mathcal{X} = \{X_t : t \in V(T)\}$ is a near-partition of the vertices of $V(G)$. Furthermore, we require that if T_1, \dots, T_r are the connected components of T , then $\bigcup_{t \in V(T_i)} X_t$ for $i \in [r]$ are exactly the vertex sets of connected components of G . In other words, the forest T has exactly one tree per each connected component of G , with this tree being a tree-cut decomposition of the connected component. We call the elements of $V(T)$ *nodes* and the elements of $V(G)$ *vertices* for clarity. The set X_t is called the *bag* of the decomposition corresponding to the node t , or just the *bag at t* . By choosing a root in each tree of T , thus making T into a rooted forest, we can talk about a *rooted tree-cut decomposition*.

Let G be a graph with a tree-cut decomposition $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$. For a subset $W \subseteq V(T)$, define X_W as $\bigcup_{t \in W} X_t$. For a subgraph T' of T we write $X_{T'}$ for $X_{V(T')}$. For an edge $uv \in E(T)$ we write X_{uv}^T for $X_{T_{uv}}$ to avoid multiple subscripts. Notice that, since \mathcal{X} is a near-partition, $\{X_{uv}^T, X_{vu}^T\}$ is a near-partition of the vertex set of a connected component of G . We will call the decomposition *connected* if for every edge $uv \in E(T)$, the graphs $G[X_{uv}^T]$ and $G[X_{vu}^T]$ are connected.

The *adhesion* of an edge $e = uv$ of T , denoted $\text{adh}_{\mathcal{T}}(e)$, is defined as the set $E_G(X_{uv}^T, X_{vu}^T)$. An adhesion is *thin* if it has at most 2 edges, and is *bold* otherwise. The *torso* at a node t of T is the graph $H_t^{\mathcal{T}}$ defined as follows. Let T' be the connected component of T that contains t , and let T_1, T_2, \dots, T_p be the components of $T' - t$ (note there might be no such components if t was an isolated node). Observe that $\{X_t, X_{T_1}, \dots, X_{T_p}\}$ is a near-partition of $X_{T'}$, whereas $X_{T'}$ induces a connected component of G . Then the torso $H_t^{\mathcal{T}}$ is the graph obtained from $G[X_{T'}]$ by identifying the vertices of X_{T_i} into a single vertex z_i , for each $i \in [p]$, and removing all the loops created in this manner. Note that, thus, every edge between a vertex of X_{T_i} and a vertex of X_{T_j} , for some $i \neq j$, becomes an edge between z_i and z_j ; similarly for edges between X_{T_i} and X_t . The vertices of X_t are called the *core* vertices of the torso, while the vertices z_i are called the *peripheral* vertices of the torso. Finally, the *3-center* of a node t of T , denoted by $\overline{H}_t^{\mathcal{T}}$, is the graph obtained from the torso $H_t^{\mathcal{T}}$ by repeatedly suppressing peripheral vertices of degree at most two and deleting any resulting loops. That is, any peripheral vertex of degree zero or one is deleted, while a peripheral vertex of degree two is replaced by an edge connecting its two neighbors; if the two neighbors are equal, the resulting loop is deleted, potentially allowing further suppressions. As Wollan [42] shows, any maximal sequence of suppressions leads to the same graph $\overline{H}_t^{\mathcal{T}}$. We omit the subscripts and superscripts \mathcal{T} when they are clear from the context.

The *width* of the decomposition $\mathcal{T} = (T, \mathcal{X})$, denoted $\text{width}(\mathcal{T})$, is

$$\max\left\{\max_{e \in E(T)} |\text{adh}(e)|, \max_{t \in V(T)} |V(\overline{H}_t^{\mathcal{T}})|\right\}.$$

The *tree-cut width* of G , denoted by $\text{tctw}(G)$, is the minimum width of a tree-cut decomposition of G .

Ganian et al. [22] showed that bounded tree-cut width implies bounded treewidth. Besides, Kim et al. [29] showed that the dependency cannot be improved to subquadratic.

Lemma 6 (see [22]). *For any graph G , $\text{tw}(G) \leq 2\text{tctw}(G)^2 + 3\text{tctw}(G)$.*

Finally, Kim et al. [29] proposed a 2-approximation FPT algorithm for computing the tree-cut width of a graph.

Theorem 7 (see [29]). *There is an algorithm that, given a graph G and an integer r , runs in time $2^{\mathcal{O}(r^2 \log r)} \cdot |G|^2$ and either concludes that $\text{tctw}(G) > r$, or returns a tree-cut decomposition of G of width at most $2r$.*

Immersion. For two graphs G and H we say that G contains H as an immersion, or H is immersed in G , if there exist an injective mapping $\mu_V: V(H) \rightarrow V(G)$, and a mapping μ_E from edges of H to paths in G such that:

- for any edge $uv \in E(H)$, $\mu_E(uv)$ is a path in G with endpoints $\mu_V(u)$ and $\mu_V(v)$; and
- for any pair of different edges $e, e' \in E(H)$, the paths $\mu_E(e)$ and $\mu_E(e')$ do not have common edges.

For a family of graphs \mathcal{F} , we say a graph G is \mathcal{F} -immersion-free, or \mathcal{F} -free for short, if for every $H \in \mathcal{F}$, G does not contain H as an immersion. We define the following parameterized problem:

\mathcal{F} -IMMERSION DELETION

Input: A graph G and a positive integer k .

Parameter: k

Question: Is there a set $F \subseteq E(G)$, such that $|F| \leq k$ and $G - F$ is \mathcal{F} -immersion-free?

By $\text{OPT}_{\mathcal{F}}(G)$ we denote the minimum size of a set $F \subseteq E(G)$ such that $G - F$ is \mathcal{F} -free. If the family \mathcal{F} is clear from the context, we omit the subscript.

The following result follows from the work of Wollan [42]; the improved bound is obtained using the polynomial Excluded Grid Minor Theorem by Chekuri and Chuzhoy [8, 10, 11]. In particular, note that Theorem 1 stated in the introduction follows from it.

Theorem 8. *Let \mathcal{F} be a family of graphs that contains at least one planar subcubic graph. Then if G is an \mathcal{F} -free graph, then $\text{tctw}(G) \leq a_{\mathcal{F}}$, where $a_{\mathcal{F}} = \mathcal{O}(\max_{H \in \mathcal{F}}(|H| + \|H\|)^{30})$ is a constant depending on \mathcal{F} only.*

Proof. Recall that the $r \times r$ wall $W_{r,r}$ is a grid-like graph with maximum degree three (see Figure 1). Theorem 17 of [42] states that if G is a graph with tree-cut width at least $4r^{10} \cdot w(r)$, then G admits an immersion of the $r \times r$ wall $W_{r,r}$. Here, $w(r)$ is the upper bound in the Excluded Grid Minor Theorem, that is, the maximum treewidth of a graph that excludes the $r \times r$ grid as a minor. The currently best upper bound for $w(r)$, given by Chuzhoy in [11], is $w(r) \leq \mathcal{O}(r^{19} \text{polylog}(r))$. It is easy to see (see e.g., [24]) that there is a constant d such that every planar subcubic graph H with at most r vertices and edges can be immersed into the $(dr) \times (dr)$ -wall $W_{dr,dr}$. Hence, by the results above it follows that excluding such a graph H as an immersion imposes an upper bound of $\mathcal{O}(r^{30})$ on the tree-cut width of a graph. \square

The above theorem is a starting point for our algorithms. In particular, it implies that we can test whether a graph is \mathcal{F} -free in linear time.

Lemma 9. *Let \mathcal{F} be a family of graphs that contains at least one planar subcubic graph. There is a linear-time algorithm that checks whether a given graph is \mathcal{F} -free.*

Proof. Let $a_{\mathcal{F}}$ be the constant given by Theorem 8 for the family \mathcal{F} , and let G be the input graph. Using Bodlaender’s algorithm [3] for computing treewidth, for $k = 2a_{\mathcal{F}}^2 + 3a_{\mathcal{F}}$ we either conclude that $\text{tw}(G) > k$, or compute a tree decomposition of G of width at most k . This takes time $f(k) \cdot |G|$ for some function f , hence linear time since k is a constant. In the first case, when $\text{tw}(G) > 2a_{\mathcal{F}}^2 + 3a_{\mathcal{F}}$, we may directly conclude that G is not \mathcal{F} -free, because from Lemma 6 it follows that $\text{tctw}(G) > a_{\mathcal{F}}$, and then Theorem 8 implies that G is not \mathcal{F} -free. Thus, we may now assume that we have constructed a tree decomposition of G of width at most $k = 2a_{\mathcal{F}}^2 + 3a_{\mathcal{F}}$.

Observe now that for any fixed graph H , the property of admitting H as an immersion can be expressed in MSO_2 (Monadic Second-Order logic on graphs with quantification over edge subsets). See, for example, [25]. Therefore, by applying Courcelle’s Theorem [12], for every graph $H \in \mathcal{F}$ we may decide whether G is H -free in time $g(k) \cdot \|G\|$ for some function g ; that is, in linear time since k is a constant. By verifying this for every graph $H \in \mathcal{F}$ we decide whether G is \mathcal{F} -free. \square

From now on, throughout the whole paper, we assume that \mathcal{F} is a fixed family containing only connected graphs, of which at least one is planar and subcubic. We define the constant $\text{MAX}_{\mathcal{F}} = \max_{H \in \mathcal{F}} \|H\|$ and let $a_{\mathcal{F}}$ be the bound from Theorem 8.

3 Adjusting tree-cut decompositions

3.1 Alternative definition of tree-cut width

To simplify many arguments, we give a simpler definition of tree-cut width and show it to be equivalent. Let G be a graph with a tree-cut decomposition $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$. For every $t \in V(T)$, we define

$$w_{\mathcal{T}}(t) = |X_t| + |\{t' \in N_T(t) : \text{adh}_{\mathcal{T}}(tt') \text{ is bold}\}|.$$

We drop the subscript \mathcal{T} when it is clear from the context. We then set

$$\text{width}'(\mathcal{T}) = \max\left\{\max_{e \in E(T)} |\text{adh}(e)|, \max_{t \in V(T)} w(t)\right\}$$

and define $\text{tctw}'(G)$ as the minimum of $\text{width}'(\mathcal{T})$ over all tree-cut decompositions \mathcal{T} of G .

Theorem 10. *For every graph G it holds that $\text{tctw}(G) = \text{tctw}'(G)$. Moreover, given a tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ of G , it always holds that $\text{width}(\mathcal{T}) \leq \text{width}'(\mathcal{T})$, and a tree-cut decomposition \mathcal{T}' such that $\text{width}'(\mathcal{T}') \leq \text{width}(\mathcal{T})$ can be computed in time $\mathcal{O}(\|G\| \cdot |G|^2 \cdot \text{width}(\mathcal{T}))$.*

Proof. Let G be a graph and $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$ be a tree-cut decomposition. We first show that $\text{width}(\mathcal{T}) \leq \text{width}'(\mathcal{T})$. Consider any $t \in V(T)$ and let $t_1, t_2, \dots, t_{\ell}$ be the neighbors of t in T . For $i \in [\ell]$, denote by z_i the peripheral vertex of the torso of $t, H_t^{\mathcal{T}}$, obtained after consolidating (i.e. identifying into one vertex) the set $X_{t_i}^{\mathcal{T}}$.

We claim that $|V(\overline{H_t^{\mathcal{T}}})| \leq w(t)$. Recall that $\overline{H_t^{\mathcal{T}}}$ is the 3-center at t : its vertices are core vertices X_t and peripheral vertices z_1, \dots, z_{ℓ} that were not suppressed. Since $\deg_{H_t^{\mathcal{T}}}(z_i) = |\text{adh}(tt_i)|$ for $i \in [\ell]$, the vertex z_i might not be suppressed (and thus, belong to the 3-center at t) only when $|\text{adh}(tt_i)| \geq 3$, which implies

$$|V(\overline{H_t^{\mathcal{T}}})| \leq |X_t| + |\{t' \in N_T(t) : \text{adh}(tt') \text{ is bold}\}| = w(t) \tag{1}$$

This holds for every $t \in V(T)$, hence $\text{width}(\mathcal{T}) \leq \text{width}'(\mathcal{T})$.

In particular, $\text{tctw}(G) \leq \text{tctw}'(G)$. We now proceed to showing that $\text{tctw}(G) = \text{tctw}'(G)$. Note that without loss of generality we may assume that G is connected, as we may consider each connected component separately. Hence, all the tree-cut decompositions considered in the sequel will consist of just one tree.

Let us choose a tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ of G as follows: \mathcal{T} has the optimum width (i.e. $\text{width}(\mathcal{T}) = \text{tctw}(G)$) and, among such optimum decompositions, $\sum_{e \in E(T)} |\text{adh}(e)|$ is minimum possible. We will now prove that $|V(\overline{H_t^{\mathcal{T}}})| = w(t)$ for every $t \in V(T)$, which implies that $\text{width}'(\mathcal{T}) = \text{width}(\mathcal{T}) = \text{tctw}(G)$ by definition, concluding the claim.

Towards a contradiction, let us assume that there exists $t \in V(T)$ with $|V(\overline{H_t^{\mathcal{T}}})| \neq w(t)$, and thus by (1), $|V(\overline{H_t^{\mathcal{T}}})| < w(t)$. We denote by t_1, t_2, \dots, t_ℓ the neighbors of t in T and define z_i for $i \in [\ell]$ as above.

Without loss of generality, let (z_1, z_2, \dots, z_k) be a maximal sequence of vertices whose suppression leads to $\overline{H_t^{\mathcal{T}}}$; see Figure 2. Observe that, since the inequality $|V(\overline{H_t^{\mathcal{T}}})| < w(t)$ is strict, at least one of the suppressed vertices originally has degree 3 or more in $H_t^{\mathcal{T}}$. Let z_p be the first such vertex, that is, $\deg_{H_t^{\mathcal{T}}}(z_p) \geq 3$ and $\deg_{H_t^{\mathcal{T}}}(z_i) \leq 2$ for $i < p$. Note that $p \neq 1$ as, by definition, the first vertex to get suppressed has degree at most 2 in $H_t^{\mathcal{T}}$. Let $C_1, \dots, C_{p'}$ be the connected components of the graph induced by the vertices z_1, \dots, z_{p-1} in $H_t^{\mathcal{T}}$. Since their degrees are at most two each component C_i is either an induced cycle or an induced path in $H_t^{\mathcal{T}}$. Let $\mathcal{C} = \{C_i : N_{H_t^{\mathcal{T}}}(z_p) \cap V(C_i) \neq \emptyset\}$, that is, let \mathcal{C} be the subset of the graphs $C_1, \dots, C_{p'}$ in which z_p has a neighbor.

First notice that the set \mathcal{C} is not empty. Indeed, if z_p would not have any neighbor in at least one the components $C_1, \dots, C_{p'}$ then it still would have degree at least 3 in $H_t^{\mathcal{T}}$ after suppressing the vertices of these graphs, z_1, \dots, z_{p-1} .

Second, for $C \in \mathcal{C}$, observe that if z is a neighbor of z_p in C , then z has degree at most 2 in $H_t^{\mathcal{T}}$, including its neighbor z_p , and hence z has degree at most 1 in C . Therefore, every $C \in \mathcal{C}$ is an induced path in $H_t^{\mathcal{T}}$, whose endpoints we henceforth denote by z_C and z'_C . We have $N_{H_t^{\mathcal{T}}}(z_p) \cap V(C) \subseteq \{z_C, z'_C\}$, and since by definition $N_{H_t^{\mathcal{T}}}(z_p) \cap V(C) \neq \emptyset$, without loss of generality we will assume that always $z_C z_p \in E(H_t^{\mathcal{T}})$.

Claim 11. *There exists $C \in \mathcal{C}$ such that $N_{H_t^{\mathcal{T}}}(V(C)) \subseteq \{z_p\}$.*

Proof. We prove the claim by contradiction. That is, suppose that for every $C \in \mathcal{C}$, C has a neighbor in $V(H_t^{\mathcal{T}}) \setminus \{z_p\}$. This must be a neighbor outside $\{z_i : i \in [p-1]\}$ (since C is a connected component of the subgraph induced by these vertices), thus for every $C \in \mathcal{C}$, there is a vertex $z \in C$ such that z has a neighbor z' in $V(H_t^{\mathcal{T}}) \setminus \{z_i : i \in [p]\}$.

Let $C \in \mathcal{C}$. Since the internal vertices of the path C have degree exactly 2 both in C and in $H_t^{\mathcal{T}}$, they have no neighbors outside C . Consider now two cases depending on $|V(C)|$.

- If $|V(C)| = 1$, then $z_C = z'_C$ has an edge to a neighbor z' in $V(H_t^{\mathcal{T}}) \setminus \{z_i : i \in [p]\}$ and an edge to z_p ; since $z_C = z'_C$ has degree 2 in $H_t^{\mathcal{T}}$, it has no other incident edges in $H_t^{\mathcal{T}}$.
- If $|V(C)| \geq 2$, then z_C has an edge to a neighbor in C and an edge to z_p , and again, no other incident edges. Thus it must be that z'_C has a neighbor z' in $V(H_t^{\mathcal{T}}) \setminus \{z_i : i \in [p]\}$. Then z'_C has an edge to z' , to a neighbor in C , and no other incident edges.

We conclude that in both cases, for every $C \in \mathcal{C}$, we have $|E(C, z_p)| = 1$; moreover, after suppressing the vertices z_1, \dots, z_{p-1} , including those of C , z_p has an edge to $V(H_t^{\mathcal{T}}) \setminus \{z_i : i \in [p]\}$, a different one for every $C \in \mathcal{C}$. Therefore, the degree of z_p in $H_t^{\mathcal{T}}$ does not drop after suppressing z_1, \dots, z_{p-1} . However, initially $\deg_{H_t^{\mathcal{T}}}(z_p) \geq 3$ by choice of z_p , and after suppressing z_1, \dots, z_{p-1} , the vertex z_p must have degree at most 2 to be itself suppressed, a contradiction. \square

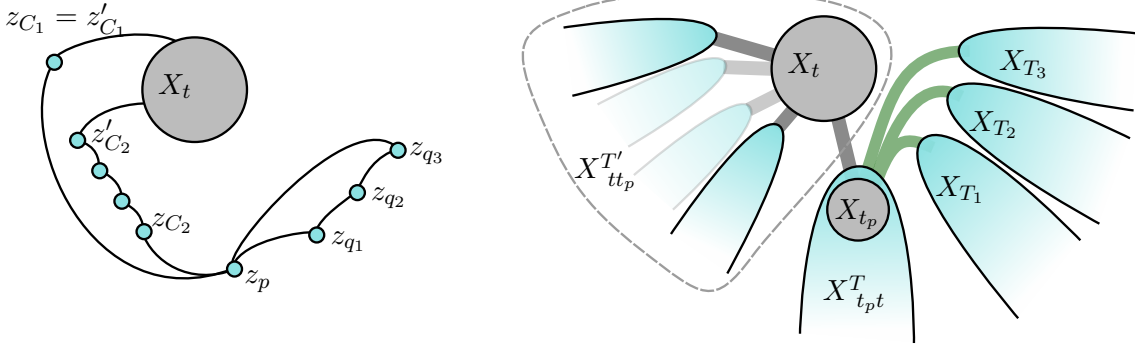


Figure 2: On the left, the torso H_t^T , including the peripheral vertices z_1, \dots, z_p suppressed in $\overline{H_t^T}$. On the right, the modified decomposition, with subtrees T_1, \dots, T_r attached to t_p instead of t . This improves the decomposition by making $\text{adh}(tt_p)$ strictly smaller (because of $z_p z_{q_1}, z_p z_{q_3}$).

Let now $C \in \mathcal{C}$ be such that $N_{H_t^T}(C) \subseteq \{z_p\}$. Let $z_{q_1}, z_{q_2}, \dots, z_{q_r}$ be the vertices of the path C . We construct a new tree-cut decomposition $\mathcal{T}' = (T', \mathcal{X})$ of G by removing the edges tt_{q_i} from T and adding the edges $t_p t_{q_i}$, for all $i \in [r]$. That is,

$$V(T') = V(T) \quad \text{and} \quad E(T') = (E(T) \setminus \{tt_{q_i} : i \in [r]\}) \cup \{t_p t_{q_i} : i \in [r]\}.$$

Note that the bags in \mathcal{T}' are exactly the same as in \mathcal{T} . We will show that $\mathcal{T}' = (T', \mathcal{X})$ has the minimum possible width, and that $\sum_{e \in E(T')} |\text{adh}_{\mathcal{T}'}(e)| < \sum_{e \in E(T)} |\text{adh}_{\mathcal{T}}(e)|$, a contradiction to the choice of \mathcal{T} . Notice that $T_{t_{q_i}t} = T'_{t_{q_i}t_p}$ for each $i \in [r]$; we will denote this subtree of T and T' by T_i from now on. The bags assigned to this subtree do not change either, thus for every $i \in [r]$ we have

$$\text{adh}_{\mathcal{T}'}(t_{q_i}t_p) = \text{adh}_{\mathcal{T}}(t_{q_i}t). \quad (2)$$

Similarly, notice that for every edge $e \in (E(T) \cap E(T')) \setminus \{tt_p\}$ we have

$$\text{adh}_{\mathcal{T}'}(e) = \text{adh}_{\mathcal{T}}(e). \quad (3)$$

Finally, let us consider the edge tt_p . By construction $V(T'_{tt_p}) = V(T_{tt_p}) \setminus \bigcup_{i \in [r]} V(T_i)$. In other words, $V(G)$ is partitioned into $X_{t_p}^T$, $X_{t_p}^{T'}$, and $\bigcup_{i \in [r]} X_{T_i}$. Observe that $\text{adh}_{\mathcal{T}'}(tt_p)$ is obtained from $\text{adh}_{\mathcal{T}}(tt_p)$ by deleting $E_G(X_{T_i}, X_{t_p}^T)$ and adding $E_G(X_{T_i}, X_{t_p}^{T'})$, for all $i \in [r]$.

Claim 12. *For every $i \in [r]$, $E_G(X_{T_i}, X_{t_p}^{T'}) = \emptyset$. Moreover, there exists $i \in [r]$, $E_G(X_{T_i}, X_{t_p}^T) \neq \emptyset$.*

Proof. By the choice of C , the only neighbor of $V(C) = \{z_{q_1}, \dots, z_{q_r}\}$ in H_t^T is z_p . That is, for each $i \in [r]$, the vertex z_{q_i} has neighbors only in z_{q_1}, \dots, z_{q_r} and z_p in H_t^T . Since H_t^T is constructed from G by consolidating X_{T_i} into z_{q_i} (for $i \in [r]$) and $X_{t_p}^T$ into z_p (among others), this means that the only edges in G leaving X_{T_i} go to $X_{T_1} \cup \dots \cup X_{T_r} \cup X_{t_p}^T$. Since $X_{t_p}^{T'}$ is obtained from $X_{t_p}^T = V(G) \setminus X_{t_p}^T$ by removing $X_{T_1} \cup \dots \cup X_{T_r}$, the first claim follows.

Similarly, by the choice of C , z_p has an edge to $V(C) = \{z_{q_1}, \dots, z_{q_r}\}$ in H_t^T . This means G has an edge between $X_{t_p}^T$ and X_{T_i} for some $i \in [r]$. \square

It follows that $\text{adh}_{\mathcal{T}'}(tt_p) \subsetneq \text{adh}_{\mathcal{T}}(tt_p)$. Together with (2), (3), this implies

$$\max_{e \in E(T')} |\text{adh}_{\mathcal{T}'}(e)| \leq \max_{e \in E(T)} |\text{adh}_{\mathcal{T}}(e)| \quad (4)$$

$$\sum_{e \in E(T')} |\text{adh}_{\mathcal{T}'}(e)| < \sum_{e \in E(T)} |\text{adh}_{\mathcal{T}}(e)|. \quad (5)$$

To prove our claim it remains to show that $\max_{t \in V(T)} |V(\overline{H_t^{\mathcal{T}}})| \leq \max_{t \in V(T)} |V(\overline{H_t^{\mathcal{T}'})}|$. Recall first that $V(T) = V(T')$. For every vertex $s \in V(T) \setminus \{t, t_p\}$ the torso at s in decomposition \mathcal{T} is the same as the torso at s in decomposition \mathcal{T}' . This implies that $\overline{H_s^{\mathcal{T}}} = \overline{H_s^{\mathcal{T}'}}$.

Consider now the torso at t in decomposition \mathcal{T}' , i.e. $H_t^{\mathcal{T}'}$. From the way T' was constructed, we have $N_{T'}(t) \subseteq N_T(t)$. Moreover, recall that for every vertex $s \in N_{T'}(t)$, $\text{adh}_{T'}(st) \subseteq \text{adh}_T(st)$. Therefore $H_t^{\mathcal{T}'}$ is a subgraph of $H_t^{\mathcal{T}}$, and thus $|V(\overline{H_t^{\mathcal{T}'})}|$ cannot be larger than $|V(\overline{H_t^{\mathcal{T}}})|$.

Consider finally the torso at the node t_p in decomposition \mathcal{T}' , i.e. $H_{t_p}^{\mathcal{T}'}$. Notice that $V(H_{t_p}^{\mathcal{T}'}) = V(H_{t_p}^{\mathcal{T}}) \cup \{z_{q_i} : i \in [r]\}$. Recall, however, that $\text{adh}_{\mathcal{T}'}(t_p t_{q_i}) = \text{adh}_{\mathcal{T}}(t t_{q_i})$ and therefore, $\deg_{H_{t_p}^{\mathcal{T}'}}(z_{q_i}) = \deg_{H_t^{\mathcal{T}}}(z_{q_i}) \leq 2$. This implies that the vertices z_{q_i} , for all $i \in [r]$, get suppressed in $\overline{H_{t_p}^{\mathcal{T}'}}$ (more precisely, we can start the procedure of obtaining $\overline{H_{t_p}^{\mathcal{T}'}}$ by suppressing them). For other vertices $s \in N_{T'}(t_p)$ we have $\text{adh}_{\mathcal{T}'}(st_p) \subseteq \text{adh}_{\mathcal{T}}(st_p)$. Hence $|V(\overline{H_{t_p}^{\mathcal{T}'})}|$ cannot be larger than $|V(\overline{H_{t_p}^{\mathcal{T}}})|$.

In any case we obtain that $|V(\overline{H_t^{\mathcal{T}'})}| \leq |V(\overline{H_t^{\mathcal{T}}})|$ for every $t \in V(T)$. Together with (4), we obtain that $\text{width}(\mathcal{T}') \leq \text{width}(\mathcal{T})$, that is, \mathcal{T}' has the minimum possible width (because we assumed \mathcal{T} has). But then (5) contradicts our choice of \mathcal{T} . Hence, $|V(\overline{H_t^{\mathcal{T}}})| = w(t)$, for every $t \in V(T)$, that is, $\text{width}(\mathcal{T}) = \text{width}'(\mathcal{T})$, which concludes the proof that $\text{tctw}(G) = \text{tctw}'(G)$.

For the algorithmic statement, note that in the proof, either we show that $\text{width}(\mathcal{T}) = \text{width}'(\mathcal{T})$, or we construct a decomposition \mathcal{T}' such that $\text{width}(\mathcal{T}') \leq \text{width}(\mathcal{T})$ and the sum of all adhesion sizes in \mathcal{T}' is strictly smaller than in \mathcal{T} (Equation (5)). Since the construction can be easily performed in time $\mathcal{O}(\|G\| \cdot |G|)$ by computing all torsos and 3-centers, and since the initial sum of all adhesion sizes can be at most $\mathcal{O}(|G| \cdot \text{width}(\mathcal{T}))$, the construction can be performed repeatedly until $\text{width}(\mathcal{T}) = \text{width}'(\mathcal{T})$, concluding the algorithm. \square

3.2 Making the decomposition connected

The goal of this section is to prove that one can make a tree-cut decomposition connected without increasing its width (more precisely, width') by much. For this, we will temporarily need a slight variation on tctw' . Let G be a graph and $\mathcal{T} = (T, \mathcal{X})$ be a tree-cut decomposition of G . For every $t \in V(T)$, recall that $\delta_T(t)$ denotes the set of edges of T incident to t . We define

$$z(t) = |X_t| + \sum_{\substack{e \in \delta(t) \\ |\text{adh}(e)| \geq 3}} |\text{adh}(e)| \quad , \quad \text{and}$$

$$\text{width}''(\mathcal{T}) = \max_{t \in V(T)} z(t).$$

Lemma 13. *For every graph G and every tree-cut decomposition \mathcal{T} of G ,*

$$\text{width}'(\mathcal{T}) - 1 \leq \text{width}''(\mathcal{T}) \leq (\text{width}'(\mathcal{T}))^2.$$

Proof. Let G be a graph and $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$ be a tree-cut decomposition of G . We first prove that $\text{width}'(\mathcal{T}) \leq \text{width}''(\mathcal{T}) + 1$, which is equivalent to the left inequality. Recall that $\text{width}'(\mathcal{T}) = \max\{\max_{e \in E(T)} |\text{adh}(e)|, \max_{t \in V(T)} w(t)\}$. Clearly for each $t \in V(T)$, from definitions of $w(t)$ and $z(t)$ we have

$$w(t) = |X_t| + |\{e \in \delta_T(t) : |\text{adh}(e)| \geq 3\}| \leq |X_t| + \sum_{\substack{e \in \delta(t) \\ |\text{adh}(e)| \geq 3}} |\text{adh}(e)| = z(t).$$

In particular, $\max_{t \in V(T)} w(t) \leq \max_{t \in V(T)} z(t)$.

Let $e^* \in E(T)$ be an edge of T that maximizes $|\text{adh}(e^*)|$. If $|\text{adh}(e^*)| \leq 2$, then trivially $|\text{adh}(e^*)| \leq |X_t| + 1 \leq z(t) + 1$ for some $t \in V(T)$. Otherwise, if $|\text{adh}(e^*)| \geq 3$, let t be an endpoint of e^* ; then $z(t) \geq |X_t| + |\text{adh}(e^*)|$, so in particular $|\text{adh}(e^*)| \leq z(t)$. Thus in any case, we conclude that

$$\text{width}'(\mathcal{T}) = \max\{|\text{adh}(e^*)|, \max_{t \in V(T)} w(t)\} \leq \max_{t \in V(T)} z(t) + 1 = \text{width}''(\mathcal{T}) + 1.$$

We now prove that $\text{width}''(\mathcal{T}) \leq (\text{width}'(\mathcal{T}))^2$. Notice that, by definition,

$$\max_{e \in E(T)} |\text{adh}(e)| \leq \text{width}'(\mathcal{T}).$$

Moreover, for every $t \in V(T)$,

$$|X_t| + |\{e \in \delta_T(t) : |\text{adh}(e)| \geq 3\}| = w(t) \leq \text{width}'(\mathcal{T}).$$

Therefore,

$$\begin{aligned} z(t) &= |X_t| + \sum_{\substack{e \in \delta(t) \\ |\text{adh}(e)| \geq 3}} |\text{adh}(e)| \\ &\leq |X_t| + \sum_{\substack{e \in \delta(t) \\ |\text{adh}(e)| \geq 3}} \text{width}'(\mathcal{T}) \\ &= |X_t| + |\{e \in \delta(t) : |\text{adh}(e)| \geq 3\}| \cdot \text{width}'(\mathcal{T}) \\ &\leq (\text{width}'(\mathcal{T}))^2. \end{aligned}$$

Thus, we conclude that $\text{width}''(\mathcal{T}) = \max_{t \in V(T)} z(t) \leq (\text{width}'(\mathcal{T}))^2$. \square

Recall that a tree-cut decomposition $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$ of G is *connected* if for every edge $uv \in E(T)$, the graphs $G[X_{uv}^T]$ and $G[X_{vu}^T]$ are connected. We now show that we may always find such a tree-cut decomposition.

Lemma 14. *Given a tree-cut decomposition of a graph G with width' at most k , a connected tree-cut decomposition of G with width' at most $k^2 + 1$ can be constructed in time $\mathcal{O}(\|G\| \cdot |G|^2 \cdot k^2)$.*

Proof. If a graph is disconnected, we may consider its connected components separately, find a connected tree-cut decomposition for each component and conclude the claim by taking the disjoint union of the decompositions. We will thus henceforth assume that G is a connected graph.

Suppose G has a tree-cut decomposition of width' at most k . Then by Lemma 13, the same decomposition has width'' at most k^2 . Let $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$ be a tree-cut decomposition of G such that $\text{width}''(\mathcal{T}) \leq k^2$ and, subject to that, $\sum_{e \in E(T)} |\text{adh}_{\mathcal{T}}(e)|^2$ is minimum possible.

We claim that \mathcal{T} is a connected decomposition. Towards a contradiction, assume that $G[X_{uv}^T]$ is not connected, for some $uv \in E(T)$. Let G_1, G_2, \dots, G_r be its connected components. Let $\mathcal{T}' = (T', \mathcal{X}')$ be the tree-cut decomposition of G obtained from (T, \mathcal{X}) in the following way. Let T_1, T_2, \dots, T_r be r distinct copies of T_{uv} where, for every $i \in [r]$,

$$\begin{aligned} V(T_i) &= \{z_i : z \in V(T_{uv})\} \\ E(T_i) &= \{f_i : f \in E(T_{uv})\}. \end{aligned}$$

To obtain T' , we remove T_{uv} from T , add the trees T_1, T_2, \dots, T_r instead, and for each $i = 1, 2, \dots, r$ we add a new edge e_i between v and $u_i \in V(T_i)$. Therefore,

$$V(T') = V(T_{vu}) \cup \bigcup_{i \in [r]} V(T_i) \quad (6)$$

$$E(T') = E(T_{vu}) \cup \bigcup_{i \in [r]} E(T_i) \cup \bigcup_{i \in [r]} \{e_i\}. \quad (7)$$

Notice then that $T_i = T'_{u_i v}$, for each $i \in [r]$. We define \mathcal{X}' in the following way.

$$X'_s = \begin{cases} X_s & \text{if } s \in V(T_{vu}) \\ X_z \cap V(G_i) & \text{if } s = z_i \text{ for some } z \in V(T_{uv}) \text{ and } i \in [r]. \end{cases} \quad (8)$$

We will obtain a contradiction by proving that

$$\text{width}''(\mathcal{T}') \leq \text{width}''(\mathcal{T}) \quad \text{and} \quad \sum_{e \in E(T')} |\text{adh}_{\mathcal{T}'}(e)|^2 < \sum_{e \in E(T)} |\text{adh}_{\mathcal{T}}(e)|^2.$$

However, towards our goal, we have to first show how adhesions in \mathcal{T}' correspond to adhesions in \mathcal{T} .

Notice that for every $f \in E(T) \cap E(T')$,

$$\text{adh}_{\mathcal{T}'}(f) = \text{adh}_{\mathcal{T}}(f). \quad (9)$$

The remaining edges of T' are of the form $p_j q_j \in E(T_j)$ or e_j , for some $j \in [r]$. In the latter case, let us write $p_j = u_j$ and $q_j = v$. We claim that $\{\text{adh}_{\mathcal{T}'}(p_j q_j) : j \in [r]\}$ is a near-partition of $\text{adh}_{\mathcal{T}}(pq)$ (here, if $p_j q_j = e_j = u_j v$, then $p = u$ and $q = v$). Indeed, $\text{adh}_{\mathcal{T}'}(p_j q_j)$ is by construction equal to the set of edges between $X_{pq}^T \cap V(G_j)$ and $X_{qp}^T \cup \bigcup_{i \neq j} V(G_i)$. Since, $E(V(G_j), V(G_i)) = \emptyset$, for $i \neq j$, it follows that

$$\text{adh}_{\mathcal{T}'}(p_j q_j) = E_G(X_{pq}^T \cap V(G_j), X_{qp}^T).$$

Since X_{qp}^T, X_{pq}^T is a near-partition of $V(G)$ and $\{X_{pq}^T \cap V(G_i) : i \in [r]\}$ is a near-partition of X_{pq}^T , we infer that $\{\text{adh}_{\mathcal{T}'}(p_j q_j) : j \in [r]\}$ is a near-partition of $E_G(X_{pq}^T, X_{qp}^T) = \text{adh}_{\mathcal{T}}(pq)$, as claimed. Therefore, for all edges $f \in E(T_{vu})$, as well as for $f = uv$ (in which case $f_i = e_i$), we have

$$\sum_{i \in [r]} |\text{adh}_{\mathcal{T}'}(f_i)| = |\text{adh}_{\mathcal{T}}(f)|. \quad (10)$$

We are now able to prove that $\text{width}''(\mathcal{T}') \leq \text{width}''(\mathcal{T})$. That is, we want to show that $\max_{t \in V(T')} z_{\mathcal{T}'}(t) \leq \max_{t \in V(T)} z_{\mathcal{T}}(t)$. Here $z_{\mathcal{T}}(\cdot)$ and $z_{\mathcal{T}'}(\cdot)$ are the $z(\cdot)$ -functions as in the definition of tctw'' , applied respectively in decompositions \mathcal{T} and \mathcal{T}' .

Let first $t \in V(T_{vu}) \setminus \{v\}$. From (8) we obtain that $|X'_t| = |X_t|$ and, from (9) we obtain that for every edge $y \in \delta_T(t)$, $|\text{adh}_{\mathcal{T}'}(y)| = |\text{adh}_{\mathcal{T}}(y)|$. Thus, $z_{\mathcal{T}'}(t) = z_{\mathcal{T}}(t)$.

Let now $t_i \in V(T_i)$, for some $i \in [r]$. From (8) we obtain that $|X'_{t_i}| \leq |X_{t_i}|$. Moreover, for every edge f_i incident to t_i , from (10) we obtain that $|\text{adh}_{\mathcal{T}'}(f_i)| \leq |\text{adh}_{\mathcal{T}}(f)|$. Thus, $z_{\mathcal{T}'}(t_i) \leq z_{\mathcal{T}}(t)$.

Finally, let $t = v$. From (8), we obtain that $|X'_v| = |X_v|$. Observe that $\delta_{T'}(v) = E_1 \uplus E_2$, where $E_1 = \delta_T(v) \setminus \{uv\}$ and $E_2 = \{e_i \mid i \in [r]\}$. Then from Equation (9), for every edge $y \in E_1$ we have $|\text{adh}_{\mathcal{T}'}(y)| = |\text{adh}_{\mathcal{T}}(y)|$, and from Equation (10), we also have $\sum_{i \in [r]} |\text{adh}_{\mathcal{T}'}(e_i)| = |\text{adh}_{\mathcal{T}}(e)|$. From this it follows that $z_{\mathcal{T}'}(v) \leq z_{\mathcal{T}}(v)$.

Thus for each $t' \in V(T')$, we have $z_{\mathcal{T}'}(t') \leq \max_{t \in V(T)} z_{\mathcal{T}}(t) = \text{width}''(\mathcal{T})$, and hence $\text{width}''(\mathcal{T}') \leq \text{width}''(\mathcal{T})$. Furthermore, from (9) and (10), we have

$$\sum_{e \in E(T')} |\text{adh}_{\mathcal{T}'}(e)| = \sum_{e \in E(T)} |\text{adh}_{\mathcal{T}}(e)|$$

Notice now that from (10), for each $f \in E(T_{vu}) \cup \{uv\}$ we have

$$\sum_{i \in [r]} |\text{adh}_{\mathcal{T}'}(f_i)|^2 \leq \left(\sum_{i \in [r]} |\text{adh}_{\mathcal{T}'}(f_i)| \right)^2 = |\text{adh}_{\mathcal{T}}(f)|^2. \quad (11)$$

Here, for $f = uv$ we consider $f_i = e_i$.

Since X_{uv}^T, X_{vu}^T is a near-partition of $V(G)$ and $V(G_1), \dots, V(G_r)$ is a partition of X_{uv}^T , we have that $X_{uv}^T, V(G_1), \dots, V(G_r)$ is a near-partition of all of $V(G)$. Furthermore, since G is connected and $E_G(V(G_i), V(G_j)) = \emptyset$ for $i \neq j$, it must be that $E_G(V(G_i), X_{uv}^T)$ is non-empty for each $i \in [r]$. This means $\text{adh}_{\mathcal{T}'}(e_i)$ is non-empty, and since $r \geq 2$, we infer that the inequality in (11) is strict for $f = uv$. We conclude that

$$\sum_{e \in E(T')} |\text{adh}_{\mathcal{T}'}(e)|^2 < \sum_{e \in E(T)} |\text{adh}_{\mathcal{T}}(e)|^2, \quad (12)$$

a contradiction to the choice of \mathcal{T} .

This concludes the proof that G has a connected tree-cut decomposition of width'' at most k^2 and hence, by Lemma 13, of width' at most $k^2 + 1$. Note that in the proof, we either proved that \mathcal{T} is already connected, or constructed a decomposition \mathcal{T}' with $\text{width}''(\mathcal{T}') \leq \text{width}''(\mathcal{T})$ and with a strictly smaller sum of squares of adhesion sizes (Equation (12)). Since the construction can be easily performed in $\mathcal{O}(\|G\| \cdot |G|)$ time, and since the sum of squares of adhesion sizes is initially bounded by $\mathcal{O}(|G| \cdot k^2)$ (as each adhesion has size bounded by width' of the decomposition, which is at most k), the construction can be performed repeatedly until \mathcal{T} is connected, concluding the algorithm. \square

3.3 Neat tree-cut decompositions

Recall that a tree-cut decomposition can be rooted by selecting a root in every its tree, which naturally imposes child-parent relation on the nodes, as well as the sibling relation. As already mentioned, the parent of a node t is denoted by $\pi(t)$. We now define additional properties of rooted tree-cut decompositions, and show that these properties can be achieved by simple modifications of the decomposition. This will help us in the next sections, where we will handle tree-cut decompositions combinatorially and algorithmically. The main notion that we will be interested in is called *neatness*.

Definition 15. Let G be a graph and $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$ be a rooted tree-cut decomposition of G . We say that \mathcal{T} is *neat* if it is connected and, furthermore, for every non-root node $t \in V(T)$ such that $\text{adh}(t\pi(t))$ is thin, and every sibling t' of t , there are no edges between $X_{t\pi(t)}^T$ and $X_{t'\pi(t)}^T$ in G .

The second condition was used by Ganian et al. [22] under the name *niceness*. We now show that every connected tree-cut decomposition can be made neat without increasing its width. The proof of this result follows closely the lines of the proof of [22, Lemma 1].

Theorem 16. *Given a connected tree-cut decomposition of a graph G with width' at most k , a neat tree-cut decomposition of G with width' at most k can be computed in time $\mathcal{O}(|G|^3)$.*

Proof. Similarly as before, if a graph is disconnected, we may consider its connected components separately, find a neat tree-cut decomposition for each component and conclude the claim by joining the decompositions into one forest. We will thus henceforth assume that G is a connected graph.

Let $\mathcal{T} = (T, \mathcal{X})$ be a connected tree-cut decomposition of G with $\text{width}'(\mathcal{T}) \leq k$. Let us arbitrarily choose a node $r \in V(T)$ to be the root of T . We will show that, after this rooting, \mathcal{T} can be transformed into a neat tree-cut decomposition of G .

Similarly to [22], we will call a node $t \in V(T) \setminus \{r\}$ *bad* if $|\text{adh}(t\pi(t))| \leq 2$ and there exists a sibling t' of t , such that there is an edge in G between $X_{t\pi(t)}^T$ and $X_{t'\pi(t)}$. Moreover, for a bad vertex t we say that a node b is a *bad neighbor* of t if $b \in V(T_{t'\pi(t)})$ for some sibling t' of t , and there is an edge between X_b and $X_{t\pi(t)}$ in G .

We define the following two procedures, similarly to [22], see Figure 3:

REROUTING(t): let t be a bad node and let b be a bad neighbor of t of maximum depth. Then remove the edge $t\pi(t)$ from T and add a new edge bt , thus making t a child of b .

TOP-DOWN REROUTING: as long as (T, \mathcal{X}) is not a neat tree-cut decomposition, pick a bad node t of minimum depth and perform REROUTING(t).

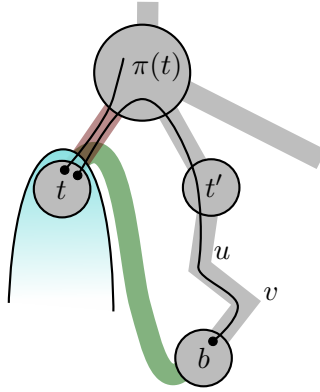


Figure 3: A decomposition with a bad node t and a bad neighbor b – the rerouting procedure will reattach the subtree $T_{t\pi(t)}$ below b . Two edges of G in $\text{adh}(tt_p)$ are shown.

We first make sure that rerouting does not spoil the connectivity of a decomposition.

Claim 17. *Let $\mathcal{T} = (T, \mathcal{X} = \{X_t : t \in V(T)\})$ be a connected rooted tree-cut decomposition and $t \in V(T)$ be a bad vertex of T . If $\mathcal{T}' = (T', \mathcal{X})$ is the rooted tree-cut decomposition obtained from \mathcal{T} after running REROUTING(t), then \mathcal{T}' is also connected.*

Proof. Notice first that, by construction, we have $E(T') = (E(T) \setminus \{t\pi(t)\}) \cup \{tb\}$. Let P denote the path in T' (and in T) that leads from $\pi(t)$ to b .

Observe that if uv is an edge of $E(T') \setminus (E(P) \cup \{tb\})$, then $V(T'_{uv}) = V(T_{uv})$ and $V(T'_{vu}) = V(T_{vu})$. Thus $G[X_{uv}^T]$ and $G[X_{vu}^T]$ are connected. Consider now the edge tb of T' . Notice that $V(T'_{tb}) = V(T_{t\pi(t)})$ and $V(T'_{bt}) = V(T_{\pi(t)t})$. This implies that the graphs $G[X_{tb}^{T'}$ and $G[X_{bt}^{T'}$ are connected as well.

Finally, consider an edge $e = uv$ of P and, without loss of generality, we assume that u is the parent of v . Notice then that

$$X_{vu}^{T'} = X_{vu}^T \cup X_{t\pi(t)}^T \quad \text{and} \quad X_{uv}^{T'} = X_{uv}^T \setminus X_{t\pi(t)}^T.$$

Recall that the subgraphs of G induced by X_{vu}^T and $X_{t\pi(t)}^T$ are connected. Moreover, since t is a bad vertex and b is a bad neighbor of t , there exists an edge between $X_{t\pi(t)}^T$ and X_b in G . Therefore, as $X_b \subseteq X_{vu}^T$, it follows that the graph $G[X_{vu}^{T'}]$ is connected.

To show that $X_{uv}^{T'} = X_{uv}^T \setminus X_{t\pi(t)}^T$ also induces a connected subgraph in G , recall that $\text{adh}_{\mathcal{T}}(t\pi(t))$ is thin. That is, $|\delta_G(X_{t\pi(t)}^T)| \leq 2$, and recall that there is an edge between $X_{t\pi(t)}^T$ and X_b in G . Observe then that $X_{uv}^T = X_{uv}^{T'} \cup X_{t\pi(t)}^T$, where there is at most one edge between $X_{uv}^{T'}$ and $X_{t\pi(t)}^T$. Since $G[X_{uv}^T]$ is connected, this implies that after removing $X_{t\pi(t)}^T$ it remains connected, because any path that connected two vertices of $X_{uv}^{T'}$ and went through $X_{t\pi(t)}^T$ would imply at least two edges between these parts. Thus, $G[X_{uv}^{T'}]$ is also connected. \lrcorner

Next, we verify that rerouting does not increase the width.

Claim 18. *Let $\mathcal{T} = (T, \mathcal{X})$ be a rooted tree-cut decomposition and $t \in V(T)$ be a bad vertex of T . If $\mathcal{T}' = (T', \mathcal{X})$ is the rooted tree-cut decomposition obtained from \mathcal{T} after running $\text{REROUTING}(t)$, then $\text{width}'(\mathcal{T}') \leq \text{width}'(\mathcal{T})$.*

Proof. As before, notice first that we have $E(T') = (E(T) \setminus \{t\pi(t)\}) \cup \{tb\}$. Let P denote the path in T' (and in T) that leads from $\pi(t)$ to b .

Observe that if e is an edge of $E(T') \setminus (E(P) \cup \{tb\})$ then $\text{adh}_{\mathcal{T}'}(e) = \text{adh}_{\mathcal{T}}(e)$. Notice also that $\text{adh}_{\mathcal{T}'}(tb) = \text{adh}_{\mathcal{T}}(t\pi(t))$.

Finally, let e be an edge of P . Notice that the edge between $X_{t\pi(t)}^T$ and X_b belongs to $\text{adh}_{\mathcal{T}}(e)$ but not to $\text{adh}_{\mathcal{T}'}(e)$. Moreover, since $|\text{adh}_{\mathcal{T}}(t\pi(t))| \leq 2$, there exists at most one edge that belongs to $\text{adh}_{\mathcal{T}'}(e)$ but not to $\text{adh}_{\mathcal{T}}(e)$. We conclude that $|\text{adh}_{\mathcal{T}'}(e)| \leq |\text{adh}_{\mathcal{T}}(e)|$, for every $e \in E(P)$.

In particular, $\max_{e \in E(T')} |\text{adh}(e)| \leq \max_{e \in E(T)} |\text{adh}(e)|$. Furthermore, since the bag at every node is unchanged, $w_{\mathcal{T}'}(v)$ can be larger than $w_{\mathcal{T}}(v)$ only for $v = b$. (Here, $w_{\mathcal{T}}(\cdot)$ and $w_{\mathcal{T}'}(\cdot)$ are functions $w(\cdot)$ as in the definition of width' , applied to decompositions \mathcal{T} and \mathcal{T}' , respectively.) However, even in this case the only additional edge incident to b is tb , whose adhesion has size at most 2. We conclude that $w_{\mathcal{T}'}(v) \leq w_{\mathcal{T}}(v)$ for each $v \in V(T)$, and hence $\text{width}'(\mathcal{T}') \leq \text{width}'(\mathcal{T})$. \lrcorner

The above two claims show that it is safe to apply the REROUTING procedure. We now show that applying it exhaustively, as described in procedure $\text{TOP-DOWN REROUTING}$, always terminates within a polynomial number of steps.

Claim 19. $\text{TOP-DOWN REROUTING}$ *terminates after $\mathcal{O}(|T|^2)$ invocations of $\text{REROUTING}(t)$.*

Proof. We note that the proof is again similar to the one in [22]. However, we include it for the sake of completeness. For a tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ and a node $v \in V(T)$ let $\text{depth}(v, T) = \text{dist}_T(v, r)$, where r is the root of T . Notice, that for every $v \in V(T)$ we have $\text{depth}(v, T) \leq |T|$, and hence

$$\sum_{v \in V(T)} \text{depth}(v, T) \leq |T|^2.$$

Let $\mathcal{T} = (T, \mathcal{X})$ be a rooted tree-cut decomposition of G and t be a bad node of \mathcal{T} such that the distance of t from r is minimum. Let $\mathcal{T}' = (T', \mathcal{X})$ be the tree-cut decomposition of G obtained after performing $\text{REROUTING}(t)$. Notice that, for every $v \in V(T_{t\pi(t)})$, $\text{depth}(v, T') \geq \text{depth}(v, T) + 1$. Moreover, for every $v \in V(T_{\pi(t)t})$, $\text{depth}(v, T') = \text{depth}(v, T)$. This implies that

$$\sum_{v \in V(T)} \text{depth}(v, T) < \sum_{v \in V(T')} \text{depth}(v, T') \leq |T|^2.$$

Therefore, since the sum of depths of nodes increases at each step, and it is always upper bounded by $|T|^2$, the TOP-DOWN REROUTING procedure terminates after at most $|T|^2$ steps. \square

This concludes the proof that TOP-DOWN REROUTING produces a neat tree-cut decomposition of width' bounded by $k^2 + 1$. Since we always assume $|T| = \mathcal{O}(|G|)$, REROUTING is invoked $\mathcal{O}(|G|^2)$ times. Finding a bad node and a bad neighbor can be done in $\mathcal{O}(|G|)$ time by inspecting edges of thin adhesions and computing the least-common-ancestor in T of the two bags containing their endpoints, using e.g. Gabow and Tarjan's classical algorithm [21]. Since REROUTING can be performed in $\mathcal{O}(|G|)$ time, the algorithm runs in $\mathcal{O}(|G|^3)$ total time. \square

We can now combine all tools developed so far to prove the following statement, which will later serve as an abstraction for getting tree-cut decompositions of \mathcal{F} -free graphs with good properties.

Corollary 20. *Given an \mathcal{F} -free graph G , a neat tree-cut decomposition of width' at most $b_{\mathcal{F}}$ of G can be computed in time $\mathcal{O}(\|G\| \cdot |G|^2)$, where $b_{\mathcal{F}} = 4(a_{\mathcal{F}})^2 + 1$ is a constant depending on \mathcal{F} only. Here, $a_{\mathcal{F}}$ is the constant given by Theorem 8.*

Proof. We have $\text{tctw}(G) \leq a_{\mathcal{F}}$ by Theorem 8. By Theorem 7, a tree-cut decomposition of width at most $2a_{\mathcal{F}}$ can be computed in time $\mathcal{O}(|G|^2)$. From it, by Theorem 10, a decomposition of width' at most $2a_{\mathcal{F}}$ can be computed in time $\mathcal{O}(\|G\| \cdot |G|^2)$. Then, by Lemma 14, a connected decomposition of width' at most $4(a_{\mathcal{F}})^2 + 1$ can be computed in time $\mathcal{O}(\|G\| \cdot |G|^2)$. Finally, by Theorem 16, a neat decomposition of width' at most $4(a_{\mathcal{F}})^2 + 1$ can be computed in time $\mathcal{O}(|G|^3)$. \square

Let G be a graph with a neat tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$, and let $p \in V(T)$. For $t \in N_T(p)$, we say the component T_{tp} of $T - p$ is connected *with a neat adhesion to p* if the adhesion of tp is thin, and moreover all of its edges have an endpoint in X_p . We now prove a result that shows what the neat decompositions are useful for: provided some node has many neighbors, all but a constant number of them is connected to it via neat adhesions.

Corollary 21. *Let G be a graph with a neat tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ with $\text{width}'(\mathcal{T}) \leq b$, for some integer b . Then for every $p \in V(T)$, at most $2b + 1$ of the connected components of $T - p$ are not connected with a neat adhesions to p .*

Proof. Let $p \in V(T)$. By the definition of width', at most b of the edges in $\delta_T(p)$ have adhesions containing more than two edges of G ; all other edges in $\delta_T(p)$ have thin adhesions. Additionally, at most one edge in $pt \in \delta_T(p)$ has the property that $\pi(p) = t$; all other edges $pt \in \delta_T(p)$ satisfy $\pi(t) = p$. Consider then the remaining edges in $\delta_T(p)$, say pt_1, \dots, pt_r , for some $r \geq |\delta_T(p)| - b - 1$. They have thin adhesions and satisfy $\pi(t_i) = p$.

Recall that $\text{adh}(pt_i)$ contains precisely the edges of G with one endpoint in $X_{t_i p}^T$ and the other in $X_{pt_i}^T$. By definition of a neat decomposition (and since $\text{adh}(t_i p) = \text{adh}(t_i \pi(t_i))$ is thin), the edges of G contained in $\text{adh}(pt_i)$ cannot have an endpoint in $X_{t' p}^T$ for any sibling t' of t_i . This means that they have one endpoint in $X_{t_i p}^T$ and one in $X_{\pi(p)p}^T \cup X_p$ (if p is the root, assume $X_{\pi(p)p}^T = \emptyset$). However, the number of edges between $X_{t_i p}^T$ and $X_{\pi(p)p}^T$ is bounded by $|\text{adh}(\pi(p)p)| \leq b$. Therefore, at least $r - b$ of the decomposition edges $t_i p$ ($i \in [r]$) have adhesions containing only edges of G that have an endpoint in X_p . This means that for at least $r - b \geq |\delta_T(p)| - 2b - 1$ indices $i \in [r]$, the component $T_{t_i p}$ of $T - p$ is connected with a neat adhesion to p . \square

4 Protrusions

We now introduce the notion of a protrusion that is suitable for our problem. Namely, protrusions are \mathcal{F} -free parts of the graph with a constant-size boundary.

Definition 22. An r -protrusion of a graph G is a set $X \subseteq V(G)$ such that $|\delta(X)| \leq r$ and $G[X]$ is \mathcal{F} -free.

Recall that by Corollary 20, the subgraph induced by a protrusion, as an \mathcal{F} -free graph, always has a neat tree-cut decomposition of width' bounded by a constant $b_{\mathcal{F}}$. In the sequel, we will only deal with $2b_{\mathcal{F}}$ - and 2-protrusions.

4.1 Replacing protrusions

As in [19], the base for our kernelization algorithm is *protrusion replacement*. That is, the algorithm iteratively finds a protrusion X that is large but has small $\delta(X)$, and replaces it with a gadget X' that has the same behaviour, but is smaller. The following lemma, whose proof is the main goal of this section, formalizes this intuition.

Lemma 23. *There is a constant $c_{\mathcal{F}}$ and algorithm that, given a graph G and a $2b_{\mathcal{F}}$ -protrusion X in it with $\|G[X]\| > c_{\mathcal{F}}$, outputs in linear time a graph G' with $\text{OPT}(G) = \text{OPT}(G')$ and $\|G'\| < \|G\|$.*

Moreover, there is a linear-time algorithm working as follows: given a subset F' of edges of G' such that $G' - F'$ is \mathcal{F} -free, the algorithm computes a subset F of edges of G such that $G - F$ is \mathcal{F} -free and $|F| \leq |F'|$ (and is called a solution-lifting algorithm).

The proof of Lemma 23 follows closely the strategy used by Fomin et al. [19]: Every $2b_{\mathcal{F}}$ -protrusion can be assigned a type, where the number of types is bounded by a function depending on \mathcal{F} only. The type of a protrusion can be computed efficiently due to protrusions having constant treewidth. Protrusions with the same type behave in the same way with respect to the problem of our interest, and hence can be replaced by one another. Therefore, we store a replacement table consisting of the smallest protrusion of each type, so that every larger protrusion can be replaced by a smaller representative stored in the table. The lifting algorithm finds, using dynamic programming, a partial solution in the large protrusion that has the same behaviour as the given partial solution in the replacement protrusion, while being not larger.

We now proceed with implementing this plan formally. We start with defining *boundaried graphs*.

Definition 24. An r -boundaried graph consists of an underlying graph G and an r -tuple (u_1, \dots, u_r) of (not necessarily different) vertices of G , called the *boundary*. Given two r -boundaried graphs

$$\mathbb{G} = (G, (u_1, \dots, u_r)) \quad \text{and} \quad \mathbb{H} = (H, (v_1, \dots, v_r)),$$

we define their *gluing*, denoted $\mathbb{G} \oplus \mathbb{H}$, to be the following graph: take the disjoint union of G and H , and for each $i \in [r]$ add one edge $u_i v_i$. Finally, we define $\|\mathbb{G}\|$ to be $\|G\|$.

We extend all notation for graphs to boundaried graphs, always applying it to the underlying graph. Thus, we can talk about, e.g., \mathcal{F} -free boundaried graphs.

Boundaried graphs can be naturally equipped with a Myhill-Nerode-like equivalence relation concerning the problem of our interest.

Definition 25. Two r -boundaried graphs \mathbb{G}_1 and \mathbb{G}_2 are called \mathcal{F} -equivalent if for every r -boundaried graph \mathbb{H} , the following holds:

$$\text{OPT}(\mathbb{G}_1 \oplus \mathbb{H}) = \text{OPT}(\mathbb{G}_2 \oplus \mathbb{H}).$$

Obviously, \mathcal{F} -equivalence is an equivalence relation on r -boundaried graphs. We now introduce a condition that implies \mathcal{F} -equivalence, which will be combinatorially easier to handle.

Suppose \mathbb{G} is an r -boundaried graph, with boundary (u_1, u_2, \dots, u_r) . Define the *extended graph* $\text{ext}(\mathbb{G})$ as follows: for each $i \in [r]$, introduce a new vertex $\text{copy}(u_i)$ that is adjacent only to u_i . Suppose further that Q is some graph. If ϕ is a partial function from $V(Q)$ to $[r]$, then by a ϕ -rooted immersion model of Q in $\text{ext}(\mathbb{G})$ we mean an immersion model of Q in $\text{ext}(\mathbb{G})$ that is faithful w.r.t. ϕ in the following sense: for each vertex v of Q that has defined image under ϕ , v is mapped to $\text{copy}(u_{\phi(v)})$ in the immersion model.

Fix a positive integer r and recall that $\text{MAX}_{\mathcal{F}} = \max_{H \in \mathcal{F}} \|H\|$. Consider a graph Q and a partial function ϕ from $V(Q)$ to $[r]$. We call the pair (Q, ϕ) *relevant* if the following conditions hold:

- $\|Q\| \leq (r + 1)\text{MAX}_{\mathcal{F}}$ and Q has no isolated vertices; and
- ϕ is non-empty, i.e., it assigns a value to at least one argument.

The set of relevant pairs will be denoted by $\mathcal{R}_{r, \mathcal{F}}$. Observe that

$$|\mathcal{R}_{r, \mathcal{F}}| \leq 2^{\text{poly}(r, \text{MAX}_{\mathcal{F}})}. \quad (13)$$

Indeed, there are at most $2^{\text{poly}(r, \text{MAX}_{\mathcal{F}})}$ graphs with at most $(r + 1) \cdot \text{MAX}_{\mathcal{F}}$ edges and no isolated vertices, and for each of them there are at most $(r + 1)^{\mathcal{O}((r+1)\text{MAX}_{\mathcal{F}})}$ possible partial functions ϕ .

Definition 26. Let r be a positive integer and let \mathbb{G} be an r -boundaried graph. For a set $\mathcal{S} \subseteq \mathcal{R}_{r, \mathcal{F}}$ of relevant pairs, the *deletion number of \mathbb{G} w.r.t. \mathcal{S}* is the minimum number of edges that need to be deleted from $\text{ext}(\mathbb{G})$ so that it does not admit a ϕ -rooted immersion model of Q , for each $(Q, \phi) \in \mathcal{S}$. Note that the edges $u_i \text{copy}(u_i)$, for $i \in [r]$, may also be deleted in this definition. The *signature* of \mathbb{G} , denoted $\sigma[\mathbb{G}]$, is the function from subsets of $\mathcal{R}_{r, \mathcal{F}}$ to nonnegative integers defined as follows:

$$\sigma[\mathbb{G}](\mathcal{S}) = \text{deletion number of } \mathbb{G} \text{ w.r.t. } \mathcal{S}.$$

The following lemma explains the relation between \mathcal{F} -equivalence and signatures.

Lemma 27. *If two \mathcal{F} -free r -boundaried graphs have the same signatures, then they are \mathcal{F} -equivalent.*

Proof. Let $\mathbb{G}_1, \mathbb{G}_2$ be a pair of \mathcal{F} -free r -boundaried graphs that have the same signature. For $t = 1, 2$, let (u_1^t, \dots, u_r^t) be the boundary of \mathbb{G}_t . Take any r -boundaried graph \mathbb{H} , and let (v_1, \dots, v_r) be its boundary.

We need to prove that $\text{OPT}(\mathbb{G}_1 \oplus \mathbb{H}) = \text{OPT}(\mathbb{G}_2 \oplus \mathbb{H})$. It suffices to prove that $\text{OPT}(\mathbb{G}_1 \oplus \mathbb{H}) \leq \text{OPT}(\mathbb{G}_2 \oplus \mathbb{H})$, because then the converse inequality will follow by symmetry. Throughout the proof, we implicitly identify $\mathbb{G}_1, \mathbb{G}_2$, and \mathbb{H} with their copies in the gluings $\mathbb{G}_1 \oplus \mathbb{H}$ and $\mathbb{G}_2 \oplus \mathbb{H}$. We also use the extended graphs $\text{ext}(\mathbb{G}_1)$ and $\text{ext}(\mathbb{G}_2)$, with the notation $\text{copy}(\cdot)$, and injective mappings

$$\begin{aligned} \iota_1: E(\text{ext}(\mathbb{G}_1)) &\rightarrow E(\mathbb{G}_1 \oplus \mathbb{H}) \\ \iota_2: E(\text{ext}(\mathbb{G}_2)) &\rightarrow E(\mathbb{G}_2 \oplus \mathbb{H}) \end{aligned}$$

defined as follows. If $e \in E(\mathbb{G}_1)$, then $\iota_1(e) = e$, and if $e = u_i^1 \text{copy}(u_i^1)$ for some $i = 1, \dots, r$, then $\iota_1(e) = u_i^1 v_i$. Mapping ι_2 is defined in the same way.

Suppose F_1 is an optimum-size subset of edges of $\mathbb{G}_1 \oplus \mathbb{H}$ such that $(\mathbb{G}_1 \oplus \mathbb{H}) - F_1$ is \mathcal{F} -free; that is, $|F_1| = \text{OPT}(\mathbb{G}_1 \oplus \mathbb{H})$. Let $L_1 = \iota_1^{-1}(F_1 \setminus E(\mathbb{H}))$; that is, L_1 consists of all edges of $\text{ext}(\mathbb{G}_1)$ that correspond to edges of F_1 under mapping ι_1 . Let \mathcal{S} be the set of all relevant pairs $(Q, \phi) \in \mathcal{R}_{r, \mathcal{F}}$ for which $\text{ext}(\mathbb{G}_1) - L_1$ does not admit a ϕ -rooted immersion model of Q . Since \mathbb{G}_1 and \mathbb{G}_2 have the same

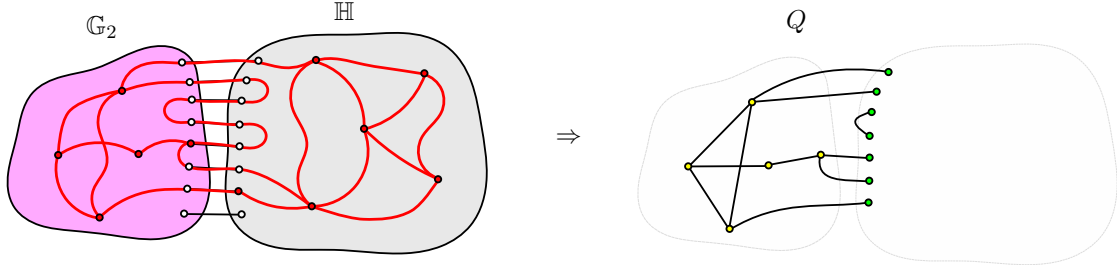


Figure 4: Construction of graph Q from the immersion model \mathcal{I} . The model \mathcal{I} is depicted on the left panel; the red vertices are the images of the vertices of H . The obtained graph Q is on the right panel. The yellow vertices are the images of vertices of H that lie within $V(\mathbb{G}_2)$, whereas the green vertices are the copies of boundary vertices that are included in the vertex set of Q . The former graphs \mathbb{G}_2 and \mathbb{H} are depicted in very light grey in order to show from where the different parts of Q come from.

signatures, there is a subset of edges $L_2 \subseteq E(\text{ext}(\mathbb{G}_2))$ with $|L_2| \leq |L_1|$ such that $\text{ext}(\mathbb{G}_2) - L_2$ also does not admit a ϕ -rooted immersion model of Q , for every $(Q, \phi) \in \mathcal{S}$. We define $F_2 \subseteq E(\mathbb{G}_2 \oplus \mathbb{H})$ as follows:

$$F_2 = \iota_2(L_2) \cup (F_1 \cap E(\mathbb{H})).$$

Since $|L_2| \leq |L_1|$, we also have that $|F_2| \leq |F_1|$. Hence it suffices to show that $(\mathbb{G}_2 \oplus \mathbb{H}) - F_2$ is \mathcal{F} -free.

For the sake of contradiction suppose that $(\mathbb{G}_2 \oplus \mathbb{H}) - F_2$ contains an immersion model \mathcal{I} of some graph $H \in \mathcal{F}$. Clearly \mathcal{I} must use at least one edge outside $E(\mathbb{H})$, because otherwise \mathcal{I} would be also an immersion model of H in $(\mathbb{G}_1 \oplus \mathbb{H}) - F_1$, which is \mathcal{F} -free by assumption. Also, \mathcal{I} must use at least one edge outside $E(\mathbb{G}_2)$, because otherwise it would be an immersion model of H in \mathbb{G}_2 , which is \mathcal{F} -free by the supposition of the lemma.

Take any edge e of H , and let P_e be the path in the model \mathcal{I} that is the image of e . Each vertex traversed by P_e belongs either to $V(\mathbb{G}_2)$ or to $V(\mathbb{H})$. For each maximal interval I on P_e of vertices belonging to $V(\mathbb{G}_2)$, consider the path in $\text{ext}(\mathbb{G}_2)$ constructed as follows: take all edges of P_e incident to the vertices of I (so including the edge preceding and succeeding I on the path), and map them to the edges of $\text{ext}(\mathbb{G}_2)$ using ι_2^{-1} . This image is a path in $\text{ext}(\mathbb{G}_2)$ whose endpoints are either copies $\text{copy}(u_i^2)$ of some boundary vertices, or the original endpoints of P_e .

Starting from H , construct a graph Q as follows (see Fig. 4 for reference). The vertex set of Q consists of all the vertices of H that are mapped to $V(\mathbb{G}_2)$ in the model \mathcal{I} , plus the set of all the vertices $\text{copy}(u_i^2)$ for which the edge $u_i^2 v_i$ is used in the model \mathcal{I} . The edges of Q are defined by the construction of the previous paragraph: every path R constructed for some maximal interval on some path P_e gives rise to an edge in Q connecting the endpoints of R . It is now easy to see that the above paths define a ϕ -rooted immersion model of Q in $\text{ext}(\mathbb{G}_2) - L_2$, where ϕ assigns each vertex $\text{copy}(u_i^2)$ its index i .

We now verify that (Q, ϕ) is a relevant pair. First, since every graph of \mathcal{F} is connected and has at least one edge, it is immediate that Q has no isolated vertices. For every edge e of H , the path P_e can alternate between $V(\mathbb{G}_2)$ and $V(\mathbb{H})$ at most r times, and hence e can give rise to at most $r + 1$ edges in Q ; it follows that

$$\|Q\| \leq (r + 1)\|H\| \leq (r + 1)\text{MAX}_{\mathcal{F}}.$$

Finally, since \mathcal{I} uses at least one edge outside $E(\mathbb{H})$ and at least one edge outside $E(\mathbb{G}_2)$, we conclude that neither Q nor ϕ is empty.

Since (Q, ϕ) is a relevant pair for which there is a ϕ -rooted immersion model of Q in $\text{ext}(\mathbb{G}_2) - L_2$, we have that $(Q, \phi) \notin \mathcal{S}$. By the way we defined \mathcal{S} , it follows that there is a ϕ -rooted immersion model of Q in $\text{ext}(\mathbb{G}_1) - L_1$. Take the edges of this model, map them according to ι_1 to edges of $\mathbb{G}_1 \oplus \mathbb{H}$, and add all the edges used by model \mathcal{I} within $E(\mathbb{H})$. It can be now easily seen that all these edges form an immersion model of H in $(\mathbb{G}_1 \oplus \mathbb{H}) - F_1$, which is a contradiction with $(\mathbb{G}_1 \oplus \mathbb{H}) - F_1$ being \mathcal{F} -free. \square

It is not hard to see that the deletion numbers in fact cannot be too large.

Lemma 28. *If \mathbb{G} is an r -boundaried graph and $\mathcal{S} \subseteq \mathcal{R}_{r, \mathcal{F}}$ is a subset of relevant pairs, then the deletion number of \mathbb{G} w.r.t. \mathcal{S} is at most r .*

Proof. Let (u_1, \dots, u_r) be the boundary of \mathbb{G} . Since ϕ is non-empty for each $(Q, \phi) \in \mathcal{S}$, in order to make $\text{ext}(\mathbb{G})$ not admit ϕ -rooted minor of Q one can always remove all the edges $u_i \text{copy}(u_i)$, for $i \in [r]$. Hence, the deletion number of \mathbb{G} w.r.t. \mathcal{S} is upper bounded by the number of these edges, that is, by r . \square

Lemmas 27 and 28, together with (13), immediately yield the following.

Corollary 29. *The number of possible signatures of r -boundaried graphs is at most $2^{2^{\text{poly}(r, \text{MAX}_{\mathcal{F}})}}$. Consequently, \mathcal{F} -equivalence has at most this many equivalence classes.*

Finally, we need the algorithmic tractability of signatures.

Lemma 30. *For every positive integer r , there exists a linear-time algorithm that, given an \mathcal{F} -free r -boundaried graph \mathbb{G} , computes its signature.*

Proof. For each such subset \mathcal{S} of relevant pairs, the deletion number of \mathbb{G} w.r.t. \mathcal{S} can be computed in linear time as follows. First, observe that, due to \mathbb{G} being \mathcal{F} -free, by Proposition 6 and Theorem 8 we infer that the treewidth of $\text{ext}(\mathbb{G})$ is bounded by a constant depending on \mathcal{F} only. Hence, using Bodlaender's algorithm [3] we can compute in linear time a tree decomposition of $\text{ext}(\mathbb{G}_1)$ of constant width. Then, on this tree decomposition we apply the optimization variant of Courcelle's theorem, due to Arnborg et al. [1] (see also [13, Theorem 7.12] for a modern presentation). For this, we observe that finding the minimum cardinality of an edge subset of $\text{ext}(\mathbb{G})$ that hits all ϕ -rooted immersion model of Q , for each $(Q, \phi) \in \mathcal{S}$, can be expressed in a straightforward way as an MSO_2 optimization problem; the formula's length depends only on r and \mathcal{F} . Thus, the algorithm of Arnborg et al. [1] solves this optimization problem in linear-time, yielding the deletion number of \mathbb{G} w.r.t. \mathcal{S} . By applying this procedure to all subsets \mathcal{S} of relevant pairs, whose number is bounded by a constant depending on r and \mathcal{F} , we obtain the whole signature of \mathbb{G} . \square

We are ready to prove the basic protrusion replacement lemma, i.e., Lemma 23.

Proof of Lemma 23. We first describe the algorithm that computes G' . Recall that, by Corollary 29, for any $r \leq 2b_{\mathcal{F}}$ the number of possible signatures of r -boundaried graphs is bounded by a constant depending \mathcal{F} only. Define the following table T : for each $r \leq 2b_{\mathcal{F}}$ and each possible signature ρ of r -boundaried graphs, we store in T the smallest, in terms of the number of edges, \mathcal{F} -free r -boundaried graph \mathbb{G}_ρ for which $\sigma[\mathbb{G}_\rho] = \rho$. If no \mathcal{F} -free r -boundaried graph has signature ρ , a marker \perp is stored instead. Note that table T depends only on family \mathcal{F} , and hence can be hardcoded in the algorithm. Define $c_{\mathcal{F}}$ to be the largest number of edges among the graphs stored in T ; then $c_{\mathcal{F}}$ is a constant depending on \mathcal{F} only.

Let $r = |\delta(X)|$. Based on $G[X]$ and $G - X$, define r -boundaried graphs \mathbb{G}_X and \mathbb{H} as follows. The underlying graph of \mathbb{G}_X is $G[X]$, and of \mathbb{H} is $G - X$. Fix an arbitrary ordering e_1, e_2, \dots, e_r of the edges of $\delta(X)$. Then the i -th boundary vertex of \mathbb{G}_X is the endpoint of e_i that lies in X , and the i -th boundary vertex of \mathbb{H} is the second endpoint of e_i , the one that lies outside of X . It follows that $G = \mathbb{G}_X \oplus \mathbb{H}$.

Using the algorithm of Lemma 30, compute the signature $\rho := \sigma[\mathbb{G}_X]$. Note here that \mathbb{G}_X is \mathcal{F} -free by the supposition that X is a protrusion. Since \mathbb{G}_X has signature ρ , it follows that table T stores some r -boundaried graph \mathbb{G}_ρ with the same signature. As $\|G[X]\| = \|\mathbb{G}_X\| > c_{\mathcal{F}}$ and $\|\mathbb{G}_\rho\| \leq c_{\mathcal{F}}$, we have that $\|\mathbb{G}_\rho\| < \|\mathbb{G}_X\|$.

Define

$$G' := \mathbb{G}_\rho \oplus \mathbb{H}.$$

As $\|\mathbb{G}_\rho\| < \|\mathbb{G}_X\|$, we have that $\|G'\| < \|G\|$. Since \mathbb{G}_X and \mathbb{G}_ρ have the same signatures and are both \mathcal{F} -free, by Lemma 27 we have that they are \mathcal{F} -equivalent. Hence

$$\text{OPT}(G) = \text{OPT}(\mathbb{G}_X \oplus \mathbb{H}) = \text{OPT}(\mathbb{G}_\rho \oplus \mathbb{H}) = \text{OPT}(G'),$$

and we conclude that G' can be output by the algorithm.

We now describe the solution lifting algorithm. Suppose we are given a subset F' of edges of G' such that $G' - F'$ is \mathcal{F} -free. Recall that $G' = \mathbb{G}_\rho \oplus \mathbb{H}$. Let $F'_{\mathbb{H}} = F' \cap E(\mathbb{H})$ and let $F'_\rho = F' \setminus F'_{\mathbb{H}}$; here, we implicitly identify \mathbb{G}_ρ and \mathbb{H} with their copies in the gluing $G' = \mathbb{G}_\rho \oplus \mathbb{H}$. Consider the extended graph $\text{ext}(\mathbb{G}_\rho)$, and let \widetilde{F}'_ρ be the image of F'_ρ under the mapping ι^{-1} defined as in the proof of Lemma 27: the edges of \mathbb{G}_ρ are mapped to themselves, while the edges between \mathbb{G}_X and \mathbb{H} are mapped to the corresponding edges between the boundary vertices and their copies in $\text{ext}(\mathbb{G}_\rho)$.

Since $\text{ext}(\mathbb{G}_\rho)$ is a graph of constant size, we can compute in constant time the subset $\mathcal{S} \subseteq \mathcal{R}_{r,\mathcal{F}}$ of those relevant pairs $(Q, \phi) \in \mathcal{R}_{r,\mathcal{F}}$, for which $\text{ext}(\mathbb{G}_\rho)$ does not admit a ϕ -rooted immersion model of Q . Observe that since the signatures of \mathbb{G}_ρ and \mathbb{G}_X are the same, the deletion numbers of \mathbb{G}_ρ and \mathbb{G}_X w.r.t \mathcal{S} are equal. Hence, there exists a subset \widetilde{F}'_X of edges of $\text{ext}(\mathbb{G}_X)$ with $|\widetilde{F}'_X| \leq |\widetilde{F}'_\rho|$, such that also in $\text{ext}(\mathbb{G}_X) - \widetilde{F}'_X$ there is no ϕ -rooted immersion model of Q , for each $(Q, \phi) \in \mathcal{S}$.

Observe that such set \widetilde{F}'_X can be computed in linear time using the algorithm of Arnborg et al. [1] as follows. Just as in Lemma 30, the fact that \mathbb{G}_X is \mathcal{F} -free implies that $\text{ext}(\mathbb{G}_X)$ has constant treewidth. Hence, we can compute its tree decomposition of constant width using Bodlaender's algorithm [3]; this takes linear time. Then, on this decomposition we run the algorithm of Arnborg et al. [1] for the MSO_2 optimization problem defined as follows: find the smallest subset of edges whose removal leaves no ϕ -rooted immersion model of Q , for each $(Q, \phi) \in \mathcal{S}$. The algorithm of Arnborg et al. [1] can within the same linear running time also reconstruct the solution, so we are indeed able to construct \widetilde{F}'_X .

Let now $F_X \subseteq E(G)$ be the image of \widetilde{F}'_X under the mapping ι defined as in the proof of Lemma 27: the edges of \mathbb{G}_X are mapped to themselves, while the edges between the boundary vertices and their copies are mapped to the corresponding edges between \mathbb{G}_X and \mathbb{H} . Define $F := F_X \cup F'_{\mathbb{H}}$. By the construction of F_X we have that for every relevant pair $(Q, \phi) \in \mathcal{R}_{r,\mathcal{F}}$, if $\text{ext}(\mathbb{G}_X) - \widetilde{F}'_X$ admits a ϕ -rooted immersion model of Q , then so does $\text{ext}(\mathbb{G}_\rho) - \widetilde{F}'_\rho$. A simple replacement argument, essentially the same as in the proof of Lemma 27, shows that the \mathcal{F} -freeness of $G' - F'$ implies that $G - F$ is also \mathcal{F} -free. Also, $|F| \leq |F'|$ due to $|\widetilde{F}'_X| \leq |\widetilde{F}'_\rho|$, so the solution F can be returned by the solution-lifting algorithm. \square

We henceforth define a *replaceable protrusion* in G as a $2b_{\mathcal{F}}$ -protrusion X with $\|G[X]\| > c_{\mathcal{F}}$, where $c_{\mathcal{F}}$ is the constant given by Lemma 23.

Note that the proof of Lemma 23 a priori does not give any concrete upper bound on the sizes of the replacement graphs stored in table T , and on the obtained constant $c_{\mathcal{F}}$. Also, it is unclear how to compute the table T based on the knowledge of \mathcal{F} . Obviously, we do not need the computability of T or any concrete upper bound on $c_{\mathcal{F}}$, because we design algorithms for a family \mathcal{F} fixed in advance, so objects that depend on \mathcal{F} only may be hard-coded in the algorithms. However, we find it instructive to discuss the matter of computability of T and the bound on $c_{\mathcal{F}}$, at least intuitively.

First of all, the algorithm of Arnborg et al. [1] is based on constructing an automaton that traverses the given tree decomposition of a graph. In our case, we have a constant upper bound on the minimum size of the sought set, so we are actually working with a finite-state tree automaton. By tracing the standard translation from MSO_2 to tree automata, one can estimate the number of states in the tree automaton constructed by the algorithm. This number is bounded by a tower function of constant height applied to $\text{MAX}_{\mathcal{F}}$ and the width of the decomposition; this is because the formula expressing the problem has constant quantifier rank. This also gives an upper bound on the minimum size of a tree decomposition that the automaton evaluates to a given state, which directly corresponds to the sizes of graphs stored in table T . The automaton can be explicitly constructed by the algorithm of Arnborg et al. [1], and from the automaton one can retrieve small candidates for graphs stored in T .

The above strategy roughly shows that the graphs that are stored in T are of size bounded by a tower function of constant height applied to $\text{MAX}_{\mathcal{F}}$ and the width of the decomposition. However, based on this idea one can also give a direct proof, as follows. Take any graph \mathbb{G} stored in T , and let \mathcal{T} be its tree-cut decomposition of width at most $b_{\mathcal{F}}$. With every node x of \mathcal{T} one can associate a boundaried graph \mathbb{G}_x , which corresponds to the subtree rooted at x ; the adhesion between x and its parent forms the boundary. Assume for a moment that \mathbb{G} is very large. Suppose first that the depth of \mathcal{T} is large, more precisely larger than the total number of signatures of r -boundaried graphs, for $r \leq b_{\mathcal{F}}$. Then there is some root-to-leaf path in \mathcal{T} that contains two nodes x and y , say x being an ancestor of y , for which \mathbb{G}_x and \mathbb{G}_y have the same signatures. It can be easily seen that the part between \mathbb{G}_x and \mathbb{G}_y can be “unpumped”: we can replace \mathbb{G}_x with \mathbb{G}_y , obtaining a smaller graph \mathbb{G}' with the same signature as \mathbb{G} . If this unpumping cannot be applied, then the depth of \mathcal{T} is bounded by the number of signatures, and \mathbb{G}_x can be large only if some node has a large number of children. But then again, a similar unpumping strategy can be applied if the number of children is larger than some constant depending on the number of possible signatures. Thus we obtain an explicit upper bound on the size of a graph that can be stored in T instead of \mathbb{G} .

Once all these arguments are formalized, one can prove the following result that gives an upper bound on the sizes of graphs that are stored in T .

Lemma 31. *Suppose r is a positive integer and ρ is a signature of r -boundaried graphs. If there exists an \mathcal{F} -free r -boundaried graph with signature ρ , then there is also one with at most $4\text{exp}(\text{poly}(r, \text{MAX}_{\mathcal{F}}))$ vertices and edges, where $4\text{exp}(\cdot)$ is the 4-times folded exponential function.*

Note that once we have a computable upper bound, table T may be constructed from \mathcal{F} in constant time by brute force. We would like to remark that the same unpumping strategy was recently applied by Chatzidimitriou et al. [7] for the parameter *tree-partition width*, which is similar to treewidth. The full proof of Lemma 31 will appear in the journal version of this paper.

4.2 Finding excessive protrusions

Recall that a replaceable protrusion in a graph G is a $2b_{\mathcal{F}}$ -protrusion X with $\|G[X]\| > c_{\mathcal{F}}$. To find replaceable protrusions in the input graph, we need to assume some additional connectivity constraint (which will be implied from a connected tree-cut decomposition) – this is captured by

the following definition. The larger protrusion size is needed to make any connected component of the protrusion replaceable.

Definition 32. A $2b_{\mathcal{F}}$ -protrusion B in a connected graph G is called *excessive* if $\|G[B]\| > 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$ and $G - B$ has at most two connected components.

Replaceable protrusions could be found easily if we allowed a (far worse) running time of the form $\|G\|^{\mathcal{O}(b_{\mathcal{F}})}$, but this would affect the running times in both our main results. With the above definition in hand, we use the following two techniques instead.

The first is *important cuts*, introduced by Marx [32], see also the exposition in [13, Chapter 8.2]. Intuitively, we consider (S, T) -cuts (i.e., edge sets whose removal separates the vertex sets S and T) that are ‘pushed’ towards T , meaning that we make the set of vertices reachable from S inclusion-wise maximal, without increasing the cut size. Such cuts can be effectively enumerated, allowing us to find a protrusion’s boundary.

Definition 33. Consider a graph G and disjoint vertex sets $S, T \subseteq V(G)$. Let $\Delta \subseteq E(G)$ be an (S, T) -cut and let R be the set of vertices reachable from S in $G - \Delta$. We say Δ is an *important cut* if it is inclusion-wise minimal and there is no (S, T) -cut Δ' with $|\Delta'| \leq |\Delta|$ such that $R' \supset R$, where R' is the set of vertices reachable from S in $G - \Delta'$.

Lemma 34 ([32]). *Let $S, T \subseteq V(G)$ be two disjoint sets of vertices in a graph G and let $k \geq 0$. The set of all important (S, T) -cuts of size at most k can be enumerated in time $\mathcal{O}(4^k \cdot k \cdot \|G\|)$.*

The second technique we use is *randomized contractions* by Chitnis et al. [9]. While *randomized* refers to the intuition behind this technique, following [9] we use the technique of *splitters* of Naor et al. [34] to make its usage deterministic. A convenient black-box access to splitters is given by the following lemma.

Lemma 35 ([9]). *Given a set U of size m together with integers $0 \leq a, b \leq m$, one can in time $2^{\mathcal{O}(\min(a,b) \log(a+b))} \cdot m \log m$ construct a family \mathfrak{F} of at most $2^{\mathcal{O}(\min(a,b) \log(a+b))} \cdot \log m$ subsets of U , such that the following holds: for any sets $A, B \subseteq U$ with $A \cap B = \emptyset$, $|A| \leq a$, $|B| \leq b$, there exists a set $T \in \mathfrak{F}$ with $A \subseteq T$ and $B \cap T = \emptyset$.*

These two techniques allow us to reduce excessive protrusions: we use the randomized contractions technique to find a large enough subset of a presumed excessive protrusion, after which important cuts allow us to find a boundary that makes this subset a replaceable protrusion.

Lemma 36. *There is an algorithm that, given a connected graph G , runs in time $\mathcal{O}(\|G\| \log \|G\| \cdot |G|^2)$ and either correctly concludes that G does not contain any excessive protrusion, or it outputs some replaceable protrusion in G .*

Proof. We describe the algorithm under the assumption that G contains some excessive protrusion; in this case, we show that the algorithm can compute some replaceable protrusion. If the algorithm fails to find some replaceable protrusion, then this certifies that G has no excessive protrusions, and this conclusion can be reported by the algorithm.

Let B be an excessive protrusion in G . Since G is connected and B is a $2b_{\mathcal{F}}$ -protrusion, B induces at most $2b_{\mathcal{F}}$ connected components in G . Let B' be the largest one (in the number of edges). Then clearly B' is a $2b_{\mathcal{F}}$ -protrusion with $\|B'\| > c_{\mathcal{F}}$ and with $G[B']$ connected. Furthermore, $G - B'$ has at most two components, because $G - B$ has, and every connected component of $G[B]$ is adjacent to at least one of the components of $G - B$, due to the connectivity of G . We consider B' instead of B from now on.

Let T be a tree spanning a subset of B' with $\min(|B'|, c_{\mathcal{F}} + 2)$ vertices. Then $\|T\| \leq c_{\mathcal{F}} + 1$ and $\|G[V(T)]\| > c_{\mathcal{F}}$. Let s_1, s_2 be arbitrary vertices in the two components of $G - B'$ (set $s_1 = s_2$ if it has only one component) and set $S = \{s_1, s_2\}$.

To find T , we now apply Lemma 35 for universe $U := E(G)$ and constants $a := \|T\| \leq c_{\mathcal{F}} + 1$ and $b := |\delta(B')| \leq 2b_{\mathcal{F}}$. Thus, in time $\mathcal{O}(\|G\| \log \|G\|)$ we construct a family \mathfrak{F} of $\mathcal{O}(\log \|G\|)$ subsets of $E(G)$ with the following guarantee: for at least one $F \in \mathfrak{F}$, we have $E(T) \subseteq F$ and $\delta(B') \cap F = \emptyset$. The algorithm guesses this set $F \in \mathfrak{F}$ and the vertices of S (by iterating over $|\mathfrak{F}| \cdot |G|^2$ possibilities); we shall consider the guess *successful* if F indeed has the above property and S indeed intersects each component of $G - B'$.

Make the edges of F undeletable by considering the graph \bar{G} obtained from G by contracting all edges in F (we use the same vertex labels in \bar{G} by abuse of notation). Observe that $\delta(B')$ is an $(S, V(T))$ -cut in G of size at most $2b_{\mathcal{F}}$. If the guess was successful, it is an $(S, V(T))$ -cut of size at most $2b_{\mathcal{F}}$ in \bar{G} too, and furthermore by choice of S , the set of vertices reachable from S in $\bar{G} - \delta(B')$ is precisely $V(\bar{G}) \setminus B'$.

Consider a corresponding important cut, that is, let $\Delta \subseteq E(\bar{G})$ be an important $(S, V(T))$ -cut of size at most $2b_{\mathcal{F}}$ such that the set of vertices reachable from S in $\bar{G} - \Delta$ contains $V(\bar{G}) \setminus B'$ (the existence of such a cut is easily proved, see [13, 32]). Let \bar{X} be the set of vertices reachable from T in $\bar{G} - \Delta$; then $\bar{X} \subseteq B'$ and $\delta(\bar{X}) \subseteq \Delta$.

Let X be the set of vertices in G that gets contracted to \bar{X} in \bar{G} . Then also $X \subseteq B'$ and $\delta(\bar{X}) \subseteq \Delta$ (as a subset of $E(G) \setminus F$). That is, X is \mathcal{F} -free (because B' is) and $|\delta(X)| \leq 2b_{\mathcal{F}}$, meaning X is a $2b_{\mathcal{F}}$ -protrusion. As X contains $V(T)$, we have $\|G[X]\| \geq \|G[V(T)]\| > c_{\mathcal{F}}$, meaning X is a replaceable protrusion.

Since Δ is an important cut of size at most $2b_{\mathcal{F}}$, we can use Lemma 34 to find it, and thus to find X , in $\mathcal{O}(\|G\|)$ time. Therefore, for at least one of $\mathcal{O}(|G|^2 \log \|G\|)$ guesses, the algorithm will find a replaceable protrusion. To handle unsuccessful guesses, the algorithm checks if the obtained set X is in fact a replaceable protrusion; this takes $\mathcal{O}(\|G\|)$ time for each guess, by Proposition 9. \square

We remark that we only defined excessive protrusions in connected graphs. Note that if B is an excessive protrusion in a connected component H of G , it would not necessarily be an excessive protrusion in G , since $G - B$ may have more components than $H - B$ (they are however not adjacent to B). We will thus consider the property that no component of G has an excessive protrusion. By this we mean that for each connected component H of G , there is no excessive protrusion in H .

By exhaustively (at most $\|G\|$ times) executing the algorithm of Lemma 36 and replacing any obtained protrusion using Lemma 23, we can get rid of all excessive protrusions. We formalize this in the following lemma, which will serve as the abstraction of protrusion replacement in the sequel.

Lemma 37 (Exhaustive Protrusion Replacement). *There is an algorithm that, given a graph G , runs in time $\mathcal{O}(\|G\|^2 \log \|G\| \cdot |G|^2)$ and computes a graph G' such that $\text{OPT}(G) = \text{OPT}(G')$, $\|G'\| \leq \|G\|$, and no connected component of G' has an excessive protrusion.*

Moreover, there exists a solution-lifting algorithm that works as follows: given a subset F' of edges of G' for which $G' - F'$ is \mathcal{F} -free, the algorithm runs in time $\mathcal{O}(\|G\|^2)$ and outputs a subset F of edges of G such that $|F| \leq |F'|$ and $G - F$ is \mathcal{F} -free.

Proof. Inspect every connected component H of G , and to each of them apply the algorithm of Lemma 36, which runs in time $\mathcal{O}(\|G\| \cdot \log \|G\| \cdot |G|^2)$. This algorithm either concludes that H has no excessive protrusion, or finds some replaceable protrusion X in H . Then X is also a replaceable protrusion in G , so by applying Lemma 23 to X we can compute in linear time a new graph G' with $\text{OPT}(G') = \text{OPT}(G)$ and $\|G'\| < \|G\|$. Having found G' , we can restart the whole algorithm on G' . Eventually, the algorithm of Lemma 36 concludes that each component has no excessive

protrusions and can hence output G . The solution-lifting algorithm follows by iteratively applying the solution-lifting algorithm of Lemma 23 for all the consecutive replacements performed above.

Since the number of edges strictly decreases in each iteration, the number of iterations is bounded by the number of edges of the original graph G . Therefore, the claimed running time follows. \square

5 Constant-factor approximation

It would be ideal if just applying the Exhaustive Protrusion Replacement (Lemma 37) reduced the size of the graph to linear in OPT . Then, we would already have a linear kernel, and taking all its edges would yield a constant-factor approximation. Unfortunately, there are graphs with no excessive protrusions, where the size is not bounded linearly in OPT . To see this, observe that even an arbitrarily large group of parallel edges is not a protrusion, so our current reduction rules will not reduce their multiplicity, even if they amount to 99% of the graph. Hence, we need to find a way to discover and account for such groups (we remark here that reducing each to $\mathcal{O}(\text{OPT})$ would be relatively easy, giving a quadratic kernel only, though). More generally, the structures that turn out to be problematic are large groups of constant-size 2-protrusions attached to the same pair of vertices; a group of parallel edges is a degenerated case of this structure. To describe the problematic structures formally, we introduce the notion of a *bouquet*.

Pruning bouquets and sets of parallel edges to constant size does not give an equivalent graph (because a larger bouquet may always require a larger number of edge deletions). However, the edge set of the resulting pruned graph intersects some optimal solution of the original graph; this is because for any deleted element, pruning preserves some number of isomorphic elements. Moreover, since the intersection is a solution for the pruned graph and the pruned graph has no bouquets, we can show that the number of edges of the pruned graph is linear in the size of the intersection.

Pruning thus gives a procedure that finds a subset of edges which intersects an optimal solution and such that the size of the subset is at most a constant factor larger than the size of this intersection. By iteratively finding such a set and removing it, we obtain a solution that is at most a constant factor larger than the optimum. In the next section we will leverage the obtained constant-factor approximation to reduce all bouquets at once, thus achieving a linear kernel.

5.1 Bouquets

Let us define the following constant (recall that $\text{MAX}_{\mathcal{F}} = \max_{H \in \mathcal{F}} \|H\|$)

$$d_{\mathcal{F}} := \max\{2b_{\mathcal{F}} \cdot c_{\mathcal{F}} + 2b_{\mathcal{F}}, 3\text{MAX}_{\mathcal{F}}\} + 1$$

We now introduce the notions of *bouquets* and *thetas*. Intuitively, a bouquet is a family of at least $d_{\mathcal{F}}$ isomorphic 2-protrusions, while a theta is a set of at least $d_{\mathcal{F}}$ parallel edges.

Definition 38. Consider a graph G , a set $U \subseteq V(G)$ and a family of 2-protrusions $\{S_i\}_{i \in I}$ such that for each $i \in I$:

- $N(S_i) = U$ (implying $|U| \leq 2$);
- $G[S_i]$ is connected; and
- $G[U \cup S_i]$ is isomorphic to $G[U \cup S_j]$ for all $i, j \in I$, with an isomorphism that maps each vertex of U to itself.

We call such a family a *bouquet attached to U* if it is maximal under inclusion (i.e. there is no proper superfamily which is also a bouquet) and has at least $d_{\mathcal{F}}$ elements. The edge set of the bouquet is the set of all edges incident to some S_i .

Definition 39. For two vertices $u, v \in V(G)$, a *theta attached to $\{u, v\}$* is a set of edges between u and v that is maximal under inclusion and has at least $d_{\mathcal{F}}$ elements.

The constant $d_{\mathcal{F}}$ is chosen so that a protrusion containing a set to which a bouquet (or theta) is attached is large enough to be excluded as an excessive protrusion, and so that any immersion of a graph of \mathcal{F} cannot simultaneously intersect all elements of a bouquet. Indeed, in any immersion of some $H \in \mathcal{F}$ in a graph G , the image of an edge of H is a path in G , which visits every vertex of the bouquet's attachment at most once, and hence intersects at most three elements of the bouquet. Thus in total, the immersion model intersects at most $\max_{H \in \mathcal{F}} 3\|H\| = 3\text{MAX}_{\mathcal{F}}$ elements of the bouquet or theta, which is less than $d_{\mathcal{F}}$.

We now show that the number of edges of a graph with no excessive protrusions, no bouquets and no thetas is linearly bounded in the optimum solution size, which formalizes the intuition that these structures are the only obstacles preventing the graph from being a linear kernel.

The following well-known notion and lemma are useful for proving such bounds. For a rooted forest T and a set $M \subseteq V(T)$, the *least common ancestor closure (lca-closure)* of M is the set $\text{lca}(M) \subseteq V(T)$ obtained from M by repeatedly adding to it the least common ancestor of every pair of nodes in the set (unless the nodes are in different connected components of the forest T).

Lemma 40 ([19]). *Let T be a rooted forest and $M \subseteq V(T)$. Then $|\text{lca}(M)| \leq 2|M|$ and every connected component C of $T - \text{lca}(M)$ has at most two neighbors in T .*

Lemma 41. *Let G be a connected graph with no excessive protrusions, no bouquets and no thetas. Then either G is \mathcal{F} -free, or $\|G\| \leq c \cdot \text{OPT}(G)$ for some constant c depending on \mathcal{F} only.*

Proof. Denote $k := \text{OPT}(G)$ and suppose G is not \mathcal{F} -free, that is, $k \geq 1$. Let $F \subseteq E(G)$ be a set of k edges such that $G - F$ is \mathcal{F} -free. By Corollary 20, $G - F$ has a neat tree-cut decomposition $(T, \mathcal{X} = \{X_t, : t \in V(T)\})$ with $\text{width}'(T, \mathcal{X}) \leq b_{\mathcal{F}}$. Let $M \subseteq V(T)$ be the lca-closure of the set of the nodes of T that correspond to the bags which contain some vertices incident to edges of F . Since $|F| \leq k$ there are at most $2k$ such bags and, by Lemma 40, we have that $|M| \leq 4k$ and that every connected component of $T - M$ has at most two neighbors in T ; see Figure 5. Recall that by $X_{T'}$ we denote the union of bags at the nodes of a subtree T' of T .

Claim 42. *For any connected component T' of $T - M$, $\|G[X_{T'}]\| \leq 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$.*

Proof. Suppose to the contrary that $\|G[X_{T'}]\| > 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$. We verify that then $X_{T'}$ is an excessive protrusion. Indeed, $X_{T'}$ has no vertices incident to F , so $G[X_{T'}]$ is \mathcal{F} -free. Moreover, T' has at most two neighbors in T , so $|\delta_G(X_{T'})| \leq 2b_{\mathcal{F}}$ and $T - V(T')$ has at most two components adjacent to T' . By the properties of neat decompositions, the unions of bags of these two components of $T - V(T')$ induce at most two connected components in $G - F$; in other words, $G - F - X_{T'}$ has at most two connected components adjacent to $X_{T'}$, say C_1, C_2 . Then $N_{G-F}(X_{T'}) \subseteq C_1 \cup C_2$ and since $X_{T'}$ has no vertices incident to F , also $N_G(X_{T'}) \subseteq C_1 \cup C_2$. Since G is connected, this implies $G - X_{T'}$ has at most two connected components (because every vertex of $G - X_{T'}$ has a path connecting it to $X_{T'}$ in G , which must visit $N_G(X_{T'})$). This shows that $X_{T'}$ is an excessive protrusion, contradicting assumptions. \square

We set $c' := 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$.

Each connected component of $T - M$ has exactly one or exactly two neighbors in M ; it cannot have zero neighbors in M , as it would then induce a component in $G - F$ with no vertices incident to F , contradicting that G is connected and not \mathcal{F} -free. Observe that the number of components that have exactly two neighbors in M is at most $|M| - 1$, because replacing each such component with an edge connecting its neighbors yields a forest with vertex set M . It remains to bound the number of components in $T - M$ with exactly one neighbor in M .

Suppose T_1, \dots, T_p are those connected components of $T - M$ for which the neighborhood in T is exactly t , for some $t \in M$. Again, by X_{T_i} we denote the union of the bags at the nodes of T_i . By Corollary 21, at most $2b_{\mathcal{F}} + 1$ of them are not connected with neat adhesions to t , so assume w.l.o.g. that $T_1, \dots, T_{p-2b_{\mathcal{F}}-1}$ are. That is, $N_G(X_{T_i})$ is a non-empty subset of X_t of size at most two, for all $i = 1, \dots, p - 2b_{\mathcal{F}} - 1$. Since there are at most $b_{\mathcal{F}}^2$ such subsets of size at most 2, at least $\frac{p-2b_{\mathcal{F}}-1}{b_{\mathcal{F}}^2} \geq p/b_{\mathcal{F}}^2 - 3$ of the sets X_{T_i} have the same neighborhood U in G . By the neatness of the decomposition, each $G[X_{T_i}]$ is connected and moreover $\|G[X_{T_i}]\| \leq c'$ by Claim 42. This implies in particular that $|X_{T_i} \cup U| \leq c' + 3$. This means that there are at most $(c' + 4)^{2c'}$ possible isomorphism types for $G[X_{T_i} \cup U]$. If there were at least $d_{\mathcal{F}}$ components with the same isomorphism type, they would form a bouquet. Hence $p/b_{\mathcal{F}}^2 - 3 \leq (c' + 4)^{2c'} \cdot d_{\mathcal{F}}$, meaning that

$$p \leq ((c' + 4)^{2c'} \cdot d_{\mathcal{F}} + 3) \cdot b_{\mathcal{F}}^2.$$

We define $c'' := ((c' + 4)^{2c'} \cdot d_{\mathcal{F}} + 3) \cdot b_{\mathcal{F}}^2$.

Therefore, $T - M$ is partitioned into at most $|M| - 1 + c'' \cdot |M| \leq (c'' + 1) \cdot |M|$ connected components. By Claim 42, for each of these components, the vertices contained in its bags induce a subgraph with at most c' edges. In addition to these edges, the edge set of G contains only:

- k edges of the deletion set F ;
- $d_{\mathcal{F}} \cdot b_{\mathcal{F}}^2$ edges in $G[X_t]$ for each $t \in M$ ($G[X_t]$ has at most $b_{\mathcal{F}}$ vertices and every pair has less than $d_{\mathcal{F}}$ edges in between, as G has no thetas); and
- up to $((c'' + 1) \cdot |M| + |M|) \cdot b_{\mathcal{F}} = (c'' + 2) \cdot |M| \cdot b_{\mathcal{F}}$ edges between parts of the partition of $V(T)$ given by individual elements of M and connected components of $T - M$ (each edge of T between different parts yields at most $b_{\mathcal{F}}$ edges).

Since $|M| \leq 4k$, we infer that the number of edges in G is at most

$$4(c'' + 1)k \cdot c' + k + 4k \cdot d_{\mathcal{F}} \cdot b_{\mathcal{F}}^2 + 4(c'' + 2)k \cdot b_{\mathcal{F}} = c''' \cdot k,$$

for a constant $c''' := 4c'(c'' + 1) + 1 + 4d_{\mathcal{F}} \cdot b_{\mathcal{F}}^2 + 4b_{\mathcal{F}}(c'' + 2)$. \square

5.2 Finding a constant-factor approximation piece by piece

To handle bouquets and thetas algorithmically, we first show that they are disjoint, as otherwise they would constitute a large protrusion. In this subsection, we frequently use the observation that, in a connected graph, a 2-protrusion with more than $2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$ edges is an excessive protrusion. Indeed, if X is a 2-protrusion in a connected graph G , then it is always the case that $G - X$ has at most two connected components, due to $|\delta(X)| \leq 2$.

Lemma 43. *Let G be a connected graph with no excessive protrusions. Then every two bouquets and/or thetas in G have disjoint edge sets. Furthermore, if a bouquet or theta is attached to $U \subseteq V(G)$, then U is disjoint with all elements of any bouquet.*

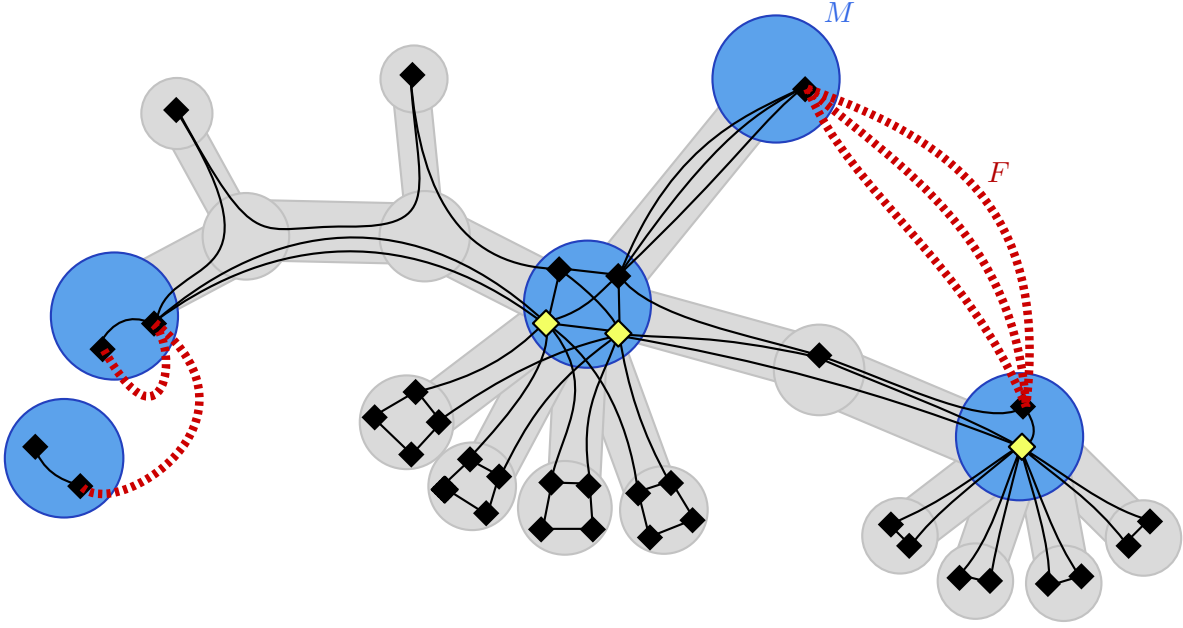


Figure 5: A graph G with a solution F (five red dotted edges), in a tree-cut decomposition of $G - F$. The set M used in Lemma 41 is highlighted in blue: three bags incident to F , a fourth central one in their lca-closure. Remaining components turn out to have sizes bounded by a constant (light gray). Two bouquets are present, their attachments visible as light yellow vertices.

Proof. Suppose a bouquet or theta \mathcal{X} is attached to U_X and a bouquet $\mathcal{Y} = \{Y_j\}_{j \in J}$ is attached to U_Y . Observe that either $|U_X| = 1$ or the two vertices of U_X are joined by at least $d_{\mathcal{F}} > 2$ edge-disjoint paths in G ; so in any case all of U_X is on one side of the cut $\delta(Y_j)$, for each $j \in J$. If $U_X \subseteq Y_j$ for some j , then all but possibly two elements of \mathcal{X} would be contained in the side Y_j of this cut. Hence Y_j would be a 2-protrusion with $\|G[Y_j]\| \geq d_{\mathcal{F}} - 2 > 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$ and thus an excessive protrusion, contradicting assumptions. We infer that U_X is disjoint with the elements of any bouquet, which concludes the second part of the claim.

To show the first part of the claim, first notice that two thetas cannot have intersecting edge sets, as they are maximal sets of parallel edges. Secondly, if a theta contained an edge from the edge set of a bouquet $\mathcal{Y} = \{Y_j\}_{j \in J}$, then either it would be an edge of $\delta(Y_j)$ for some $j \in J$, contradicting $|\delta(Y_j)| \leq 2$; or it would be an edge of $G[Y_j]$, again implying that $\|G[Y_j]\| > 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$ and that thus Y_j is an excessive protrusion, contradicting assumptions.

Finally, consider the case $\mathcal{X} = \{X_i\}_{i \in I}$ is a bouquet attached to U_X and $\mathcal{Y} = \{Y_j\}_{j \in J}$ is a bouquet attached to U_Y . Let $X = \bigcup_{i \in I} X_i$ and $Y = \bigcup_{j \in J} Y_j$. We already showed that $U_X \cap Y = \emptyset$ and symmetrically $U_Y \cap X = \emptyset$. Therefore $U_X, U_Y \subseteq V(G) \setminus (X \cup Y)$. We infer that the edge-sets of the two bouquets can intersect only if the sets X and Y intersect. Hence $X_i \cap Y_j \neq \emptyset$ for some $i \in I, j \in J$. If there was an edge between some $u \in X_i \cap Y_j$ and some $v \in Y_j \setminus X_i$, then $v \in N(X_i) = U_X$ and $v \in Y_j$, contradicting $U_X \cap Y = \emptyset$. Hence there is no edge between $X_i \cap Y_j$ and $Y_j \setminus X_i$. Since Y_j is connected and $X_i \cap Y_j$ is non-empty, this implies $Y_j \subseteq X_i$. A symmetric reasoning yields that $X_i \subseteq Y_j$, so $X_i = Y_j$. But then $U_X = U_Y$ and all the elements of both bouquets are pairwise isomorphic with an isomorphism that fixes $U_X = U_Y$. We infer that the union of the two bouquets is also a bouquet. Hence, by maximality of bouquets, we conclude that $\{X_i\}_{i \in I}$ and $\{Y_j\}_{j \in J}$ are in fact the same bouquet. \square

We now proceed to formalizing the procedure of pruning all bouquets and thetas to constant size. We show that the remaining edge set, named Δ , has the property that its removal from G strictly decreases $\text{OPT}(G)$ and its size is linear in that decrement.

Lemma 44. *Given a connected graph G with no excessive protrusion that is not \mathcal{F} -free, one can find in time $\mathcal{O}(|G|^3)$ a set $\Delta \subseteq E(G)$ such that (for some c depending on \mathcal{F} only):*

$$\text{OPT}(G - \Delta) < \text{OPT}(G) \quad \text{and} \quad |\Delta| \leq c \cdot (\text{OPT}(G) - \text{OPT}(G - \Delta)).$$

Proof. The algorithm finds all bouquets and thetas and deletes all but $d_{\mathcal{F}} - 1$ elements from each. More precisely, the algorithm first deletes all but $d_{\mathcal{F}} - 1$ edges from each theta in G , resulting in a subgraph G' . Note that since $d_{\mathcal{F}} - 1 > 2b_{\mathcal{F}}$, this reduction of thetas cannot introduce excessive protrusions in the graph and hence remaining bouquets are disjoint in the sense of Lemma 43. The algorithm then sets $V' := V(G') = V(G)$, finds all bouquets in G' and deletes from V' all vertices of all but $d_{\mathcal{F}} - 1$ elements of each bouquet. The algorithm then outputs $\Delta := E(G'[V'])$.

Bouquets in G' can be found by checking all possible attachments U of size at most 2 and all components of $G - U$ containing at most $2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$ edges (2-protrusions cannot have more edges, as they would form excessive protrusions otherwise). There are $\mathcal{O}(|G|^2)$ possible attachments U and checking all components of $G - U$ for any U takes time $\mathcal{O}(|G|)$, hence the running time follows.

To prove that $\Delta = E(G'[V'])$ has the claimed properties, let us first show that $G'[V']$ is not \mathcal{F} -free; we use the following slightly more general statement later.

Claim 45. *Let $S \subseteq V(G)$. If $G'[V' \cap S]$ is \mathcal{F} -free, then so is $G'[S]$. If $G'[S]$ is \mathcal{F} -free, then so is $G[S]$. In particular, $G'[V']$ is not \mathcal{F} -free.*

Proof. Suppose $G[S]$ (or $G'[S]$) is not \mathcal{F} -free. Then there is an immersion model of a graph from \mathcal{F} in $G[S]$ (or $G'[S]$). Since such a model intersects at most $d_{\mathcal{F}} - 1$ elements of any theta or bouquet, we can find an immersion model that intersects only the elements that were not deleted from G when constructing G' , nor from V' when constructing $G'[V']$. This means $G'[S]$ and $G'[V' \cap S]$ also contain an immersion of a graph in \mathcal{F} , that is, they are not \mathcal{F} -free. \lrcorner

Consider now an optimal solution $F \subseteq E(G)$ for G . Then $F \cap \Delta$ is a solution (not necessarily optimal) for the subgraph $G'[V']$, meaning $F \cap \Delta$ is non-empty (as $G'[V']$ is not \mathcal{F} -free) and $\text{OPT}(G'[V']) \leq |F \cap \Delta|$.

Observe that since $\text{OPT}(G) = |F|$, we have $\text{OPT}(G) - \text{OPT}(G - F \cap \Delta) = |F \cap \Delta|$. Hence

$$\text{OPT}(G) - \text{OPT}(G - \Delta) \geq |F \cap \Delta|, \quad \text{and in particular,} \quad \text{OPT}(G - \Delta) < \text{OPT}(G).$$

We show in the three claims below that $G'[V']$ is connected and has no excessive protrusions, no bouquets and no thetas. Therefore, by Lemma 41, $G'[V']$ has at most $c \cdot \text{OPT}(G'[V'])$ edges, for some constant c depending on \mathcal{F} only. Using the above inequalities, we reach the desired conclusion:

$$|\Delta| = |E(G'[V'])| \leq c \cdot \text{OPT}(G'[V']) \leq c \cdot |F \cap \Delta| \leq c \cdot (\text{OPT}(G) - \text{OPT}(G - \Delta)).$$

It remains to show that $G'[V']$ is indeed connected, has no excessive protrusions, no bouquets and no thetas. Clearly G' has no theta and thus $G'[V']$ has no theta either. Any edge or path deleted in the construction can be replaced with one that was not deleted, hence $G'[V']$ is connected.

Claim 46. *G' has no excessive protrusions.*

Proof. Suppose $S \subseteq V(G') = V(G)$ is an excessive protrusion in G' . If there was a theta in G attached to a set U with one vertex in S and the other outside of S , then in G' there would still be least $d_{\mathcal{F}} - 1$ edges connecting S and $V(G') \setminus S$, contradicting that $|\delta_{G'}(S)| \leq 2b_{\mathcal{F}} < d_{\mathcal{F}} - 1$. Hence there is no such theta and, in particular, $\delta_{G'}(S) = \delta_G(S)$.

Since S is not an excessive protrusion in G , there must be a theta in G' with an attachment U contained in S . Thus $G[S]$ has at least $d_{\mathcal{F}}$ edges, which is more than $2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$. As S is an excessive protrusion in G' , $G' - S$ has at most two connected components and thus so does $G - S$. Note that since S is a $2b_{\mathcal{F}}$ -protrusion in G' , we have $|\delta_G(S)| = |\delta_{G'}(S)| \leq 2b_{\mathcal{F}}$ and since $G'[S]$ is \mathcal{F} -free, so is $G[S]$ (Claim 45). Therefore S is an excessive protrusion in G , a contradiction. \lrcorner

Claim 47. $G'[V']$ has no excessive protrusions.

Proof. Suppose S is an excessive protrusion in $G'[V']$. Since it is not an excessive protrusion in G' , it must be the case that S had more incident edges in G' than in $G'[V']$, i.e. $\delta_{G'}(S) \supsetneq \delta_{G'[V']}(S)$. By the construction of G' , we infer that there is a bouquet $\{Y_j\}_{j \in J}$ in G' attached to some U_Y and a $j \in J$ such that Y_j was removed when constructing V' and was adjacent to S . Since $N(Y_j) = U_Y$, we have that $U_Y \cap S \neq \emptyset$.

We claim that $U_Y \subseteq S$. If U_Y has one vertex this is clear. Otherwise, if U_Y has two vertices, then at least $d_{\mathcal{F}} - 1$ elements of the bouquet connect them in $G'[V']$, yielding a family of more than $2b_{\mathcal{F}}$ edge-disjoint paths between them. Thus U_Y lies entirely on one side of the cut $\delta_{G'[V']}(S)$, because $|\delta_{G'[V']}(S)| \leq 2b_{\mathcal{F}}$. Since $U_Y \cap S \neq \emptyset$, we conclude that $U_Y \subseteq S$.

Similarly, we deduce that every other bouquet in G' has an attachment fully contained in either S or in $V' \setminus S$. Recall that V' is obtained from $V(G')$ by removing vertices of some elements from each bouquet. It follows that $\delta_{G'[V']}(S)$ is a cut in G' too. More precisely, let $S^* \subseteq V(G')$ be the set consisting of S and those elements of bouquets deleted when constructing V' that were attached to vertices in S . Then $\delta_{G'}(S^*) = \delta_{G'[V']}(S)$.

Note that as S is a $2b_{\mathcal{F}}$ -protrusion in $G'[V']$, we have $|\delta_{G'}(S^*)| = |\delta_{G'[V']}(S)| \leq 2b_{\mathcal{F}}$ and since $G'[S] = G'[V' \cap S^*]$ is \mathcal{F} -free, so is $G'[S^*]$ (Claim 45). Thus S^* is a $2b_{\mathcal{F}}$ -protrusion in G' .

Since $U_Y \subseteq S \subseteq S^*$, S^* must contain all elements of $\{Y_j\}_{j \in J}$ except for at most $|\delta_{G'}(S^*)| \leq 2b_{\mathcal{F}}$. Consequently,

$$\|G'[S^*]\| \geq d_{\mathcal{F}} - 2b_{\mathcal{F}} > 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}.$$

As S is an excessive protrusion in $G'[V']$, we have that $G'[V'] - S$ has at most two connected components. The graph $G' - S^*$ can be obtained from $G'[V'] - S$ by reintroducing elements of bouquets (which induced connected subgraphs) with attachments in $V' \setminus S$, hence $G - S^*$ also has at most two components. Therefore, S^* is an excessive protrusion in G' , a contradiction. \lrcorner

Claim 48. $G'[V']$ has no bouquet.

Proof. Suppose $G'[V']$ has a bouquet $\{X_i\}_{i \in I}$ attached to some $U_X \subseteq V'$. Since it was not removed when constructing V' , it was not a bouquet in G' , so it must be that $\delta_{G'[V']}(X_i) \subsetneq \delta_{G'}(X_i)$ for some $i \in I$. By the construction of V' , there must be some bouquet $\{Y_j\}_{j \in J}$ attached to some U_Y in G with some $Y_j, j \in J$ adjacent to X_i . Again, U_Y either consists of one vertex, or of two vertices that are still connected in $G'[V']$ by $d_{\mathcal{F}} - 1 > 2$ edge-disjoint paths (namely paths contained in the elements of $\{Y_j\}_{j \in J}$ that did not get deleted). Hence all of U_Y lies on the same side of the cut $\delta(X_i)$ in $G'[V']$. Since Y_j is adjacent to X_i , $N_{G'}(Y_j) = U_Y$ intersects X_i and thus $U_Y \subseteq X_i$. As $|\delta_{G'}(X_i)| \leq 2$, we have that $\delta_{G'[V']}(X_i)$ can intersect at most two of the $d_{\mathcal{F}} - 1$ elements of the bouquet $\{Y_j\}_{j \in J}$ that survive in $G'[V']$. All the other elements of this bouquet must lie on the same side of the cut $\delta(X_i)$ as U_Y , that is, they must be contained in X_i . So in fact X_i is a 2-protrusion in $G'[V']$ containing at least $d_{\mathcal{F}} - 3 > 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$ edges and thus an excessive protrusion, a contradiction. \lrcorner

With the above claims, we conclude that $G'[V']$ has no excessive protrusions, no bouquets and no thetas. Therefore, it satisfies the conditions of Lemma 41. \square

We extend Lemma 44 to disconnected graphs by simply considering each connected component separately.

Corollary 49. *Suppose we are given a graph G that is not \mathcal{F} -free and in which every connected component does not have any excessive protrusion. Then one can find in time $\mathcal{O}(|G|^3)$ a set $\Delta \subseteq E(G)$ such that (for some constant c depending on \mathcal{F} only):*

$$\text{OPT}(G - \Delta) < \text{OPT}(G) \quad \text{and} \quad |\Delta| \leq c \cdot (\text{OPT}(G) - \text{OPT}(G - \Delta)).$$

Proof. Let C_1, \dots, C_r be the connected components of G and let without loss of generality $C_1, \dots, C_{r'}$ be those that are not \mathcal{F} -free, for some $r' \leq r \in \mathbb{N}$. Since G is not \mathcal{F} -free and all graphs of \mathcal{F} are connected, we have $r' \geq 1$. Since every component C_i has no excessive protrusion, for each $1 \leq i \leq r'$ we compute a set $\Delta_i \subseteq E(C_i)$ by invoking Lemma 44 on C_i . Then, for some constant c depending on \mathcal{F} only, we have that:

$$\text{OPT}(C_i - \Delta_i) < \text{OPT}(C_i) \quad \text{and} \quad |\Delta_i| \leq c \cdot (\text{OPT}(C_i) - \text{OPT}(C_i - \Delta_i))$$

Let $\Delta = \bigcup_{i=1}^{r'} \Delta_i$. Since all the graph in \mathcal{F} are connected, $\text{OPT}(G) = \sum_{i=1}^r \text{OPT}(C_i)$. Components C_i that are \mathcal{F} -free have $\text{OPT}(C_i) = 0$, thus

$$\text{OPT}(G) = \sum_{i=1}^{r'} \text{OPT}(C_i) \quad \text{and similarly} \quad \text{OPT}(G - \Delta) = \sum_{i=1}^{r'} \text{OPT}(C_i - \Delta_i)$$

Since $r' \geq 1$, we have $\text{OPT}(C_1 - \Delta_1) < \text{OPT}(C_1)$ and thus $\text{OPT}(G - \Delta) < \text{OPT}(G)$. Finally

$$|\Delta| = \sum_{i=1}^{r'} |\Delta_i| \leq c \cdot (\text{OPT}(G) - \text{OPT}(G - \Delta)).$$

Therefore, the algorithm can in $\mathcal{O}(\sum_{i=1}^{r'} |C_i|^3) \leq \mathcal{O}(|G|^3)$ time output Δ as a result. \square

To get a constant-factor approximation algorithm, we invoke the above corollary iteratively. Intuitively, we maintain a set of edges F , initially empty, and invoke the corollary on $G - F$ to find a set Δ such that adding it to F decreases $\text{OPT}(G - F)$, while increasing $|F|$ by only a constant factor more. We then run the algorithm of Lemma 37 to remove excessive protrusions from $G - F$, reducing in a sense those parts of the graph where no more edges need to be deleted. Eventually, we reach $\text{OPT}(G - F) = 0$, meaning F is a solution of size linear in $\text{OPT}(G)$. The proof is straightforward, but requires reconstructing at every step a solution to the original graph given to Lemma 37.

Theorem 50 (Theorem 2, reformulated). *There is an algorithm running in time $\mathcal{O}(\|G\|^3 \log \|G\| \cdot |G|^3)$ that given a graph G , outputs a set $F \subseteq E(G)$ of size at most $c_{\text{apx}} \cdot \text{OPT}(G)$ such that $G - F$ is \mathcal{F} -free, for some constant c_{apx} depending on \mathcal{F} only.*

Proof. Let $G_0 = G$ and $\Delta_0 = \emptyset$. The algorithm computes a sequence of graphs G_i with $\|G_i\| \leq \|G\|$ and sets $\Delta_i \subseteq E(G_i)$ as follows.

For $i \geq 0$, G_{i+1} is computed from $G_i - \Delta_i$ by invoking Lemma 37 on this graph. That is, in time $\mathcal{O}(\|G\|^2 \log \|G\| \cdot |G|^3)$ we compute a graph G_{i+1} in which no connected component contains any excessive protrusions and moreover

$$\|G_{i+1}\| \leq \|G_i - \Delta_i\| \leq \|G\| \quad \text{and} \quad \text{OPT}(G_{i+1}) = \text{OPT}(G_i - \Delta_i).$$

For $i \geq 1$, provided G_i is not \mathcal{F} -free, Δ_i is computed from G_i by invoking Corollary 49. That is, in time $\mathcal{O}(|G|^3)$ we find a set $\Delta_i \subseteq E(G_i)$ such that

$$\text{OPT}(G_i - \Delta) < \text{OPT}(G_i) \quad \text{and} \quad |\Delta_i| \leq c \cdot (\text{OPT}(G_i) - \text{OPT}(G_i - \Delta_i)).$$

Here, c is the constant given by Corollary 49.

Eventually, since $\text{OPT}(G_{i+1}) < \text{OPT}(G_i)$, there is an $1 \leq r \leq \text{OPT}(G)$ such that G_r is \mathcal{F} -free. We reconstruct a sequence of solutions $F_i \subseteq E(G_i)$ for G_i as follows. Clearly $F_r := \emptyset$ is a solution for G_r . If F_{i+1} is a solution for G_{i+1} , then using the solution-lifting algorithm of Lemma 37, a solution F'_i for $G_i - \Delta_i$ can be constructed in time $\mathcal{O}(\|G\|^2)$ such that $|F'_i| \leq |F_{i+1}|$. Then $F_i := F'_i \cup \Delta_i$ is a solution for G_i . This way, we reconstruct a solution F_0 for $G_0 = G$ in at most $\text{OPT}(G)$ iterations, that is, in total time $\mathcal{O}(\|G\|^2 \cdot \text{OPT}(G))$. Constructing the graphs G_i took $\mathcal{O}(\|G\|^2 \log \|G\| \cdot |G|^3 \cdot \text{OPT}(G))$ total time, so since $\text{OPT}(G) \leq \|G\|$, the time bound follows.

To bound the size of the solution, we show inductively that $|F_i| \leq c \cdot \text{OPT}(G_i)$ for $i = r, \dots, 0$. Clearly this holds for $i = r$. If it holds for $i + 1$, then it holds for i , because:

$$|F_i| \leq |F'_i| + |\Delta_i| \leq |F_{i+1}| + |\Delta_i| \leq c \cdot \text{OPT}(G_{i+1}) + c \cdot (\text{OPT}(G_i) - \text{OPT}(G_{i+1})) = c \cdot \text{OPT}(G_i). \quad \square$$

6 Linear kernel

In the previous section we have already observed (Lemma 41) that the only structures in the graph that prevent it from being a linear kernel are excessive protrusions, bouquets, and thetas. Using the Exhaustive Protrusion Replacement (Lemma 37) we can get rid of excessive protrusions, but bouquets and thetas can still be present in the graph.

It would be ideal if we could reduce the size of every bouquet or theta to a constant, but unfortunately we are so far unable to do this. Instead, we employ the following strategy based on the idea of *amortization*. First, we reduce all excessive protrusions using Lemma 37. Second, using Theorem 50, we compute an approximate solution F that is larger than the optimum only by a constant multiplicative factor. Then, we investigate every bouquet in the graph and we estimate the number of edges of F that, in some sense, “affect” the bouquet. It can be then shown that the size of the bouquet can be reduced to linear in terms of the number of edges that affect it. Thus, after performing this reduction there still might be large bouquets in the graph, but only because a large number of edges of F affect them. However, every edge of F will affect at most a constant number of bouquets, so the total size of the bouquets will amortize to linear in terms of $|F|$, hence also linear in terms of OPT . The same amortization reasoning also enables us to bound the total sum of sizes of thetas.

We first show that if we know a local solution that isolates a bouquet into an \mathcal{F} -free part, then this bouquet can be proportionally bounded without changing $\text{OPT}(G)$.

Lemma 51. *Let $\{X_i\}_{i \in I}$ be a bouquet attached to U in G . Suppose $\Delta \subseteq E(G)$ is such that all the connected components of $G - \Delta$ that intersect $U \cup \bigcup_{i \in I} X_i$ are \mathcal{F} -free. Then $\text{OPT}(G) = \text{OPT}(G')$, where G' is obtained from G by removing vertices of all except $d_{\mathcal{F}} + |\Delta|$ elements of the bouquet.*

Proof. Suppose that, to the contrary, G' admits an edge subset $F \subseteq E(G')$ of size k such that $G' - F$ is \mathcal{F} -free, but G does not. Let $C \subseteq V(G)$ be the union of the vertex sets of those connected components of $G - \Delta$ that contain some vertices of $U \cup \bigcup_{i \in I} X_i$. By assumption, $G[C] - \Delta$ is \mathcal{F} -free. Note that $\delta(C) \subseteq \Delta$.

Let E_C be the set of all the edges incident to vertices of C in G . We claim that $|\Delta| > |F \cap E_C|$. To show this, define $F' := (F \setminus E_C) \cup \Delta$. Observe that $G - F'$ is \mathcal{F} -free: as all graphs in \mathcal{F} are

connected, an immersion model of one of them in $G - F' \subseteq G - \delta(C)$ would either be contained in C or disjoint from it. The first case would contradict the assumption that $G[C] - F' \subseteq G[C] - \Delta$ is \mathcal{F} -free. In the second case, the immersion model would be contained in $V(G) \setminus C$, which is equal to $V(G') \setminus C$, because C contains all the vertices of the bouquet. That is, it would be contained in $G[V(G) \setminus C] - F' \subseteq G'[V(G') \setminus C] - (F \setminus E_C) = G'[V(G') \setminus C] - F$, which would contradict the assumption that $G' - F$ is \mathcal{F} -free. By our supposition that G does not admit a solution of size k , we infer that $|F'| > k \geq |F|$, and hence $|\Delta| > |F \cap E_C|$, as claimed.

Since $G - F$ cannot be \mathcal{F} -free, by our assumption that G has no solution of size k , it contains an immersion model of some graph in \mathcal{F} . As argued after the definition of a bouquet, this immersion model can intersect at most $d_{\mathcal{F}}$ elements of the bouquet $\{X_i\}_{i \in I}$. Since G' still has $d_{\mathcal{F}} + |\Delta|$ isomorphic elements of this bouquet, $G' - F$ has at least $d_{\mathcal{F}} + |\Delta| - |F \cap E_C| > d_{\mathcal{F}}$ isomorphic elements that are not intersected by F . Therefore, even if the immersion model in $G - F$ intersected any elements removed from G' , they can be replaced by not intersected elements that remained unchanged in $G' - F$, thus yielding an immersion model of the same graph in $G' - F$. This means that $G' - F$ is not \mathcal{F} -free, a contradiction. \square

The same reasoning can also be applied to limit the sizes of thetas. The proof is exactly the same and hence we leave it to the reader.

Lemma 52. *Let $\{e_i\}_{i \in I}$ be a theta attached to $\{u, v\} = U$ in G . Suppose $\Delta \subseteq E(G)$ is such that all the connected components of $G - \Delta$ that contain some vertex of U are \mathcal{F} -free. Then $\text{OPT}(G) = \text{OPT}(G')$, where G' is obtained from G by removing all edges of $\{e_i\}_{i \in I}$ except for $d_{\mathcal{F}} + |\Delta|$.*

The above lemmas allow us to reduce bouquets and sets of parallel edges effectively, given a local part of an approximate solution. By appropriately amortizing bounds with the total size of the approximate solution, we finally get a linear bound on an irreducible equivalent instance.

Lemma 53. *Let G be a connected graph with no excessive protrusions and let $F \subseteq E(G)$ be such that $G - F$ is \mathcal{F} -free. Then either $\|G\| \leq c \cdot |F|$ for some constant c depending on \mathcal{F} only, or given G and F , one can compute in time $\mathcal{O}(\|G\| \cdot |G|^2)$ a subgraph G' of G such that $\text{OPT}(G) = \text{OPT}(G')$ and $\|G'\| < \|G\|$.*

Proof. We begin as in the proof of Lemma 41, except that given F we can now do the same effectively. That is, using Corollary 20, we compute a neat tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ of $G - F$ with $\text{width}'(\mathcal{T}) \leq b_{\mathcal{F}}$ in time $\mathcal{O}(\|G\| \cdot |G|^2)$. For a node t of T , by X_t we denote the bag at t .

Let $M \subseteq V(T)$ be the lca-closure of the set of bags containing a vertex incident to F ; M can be easily computed in linear time. By Lemma 40, $|M| \leq 4|F|$ and every connected component of $T - M$ has at most two neighbors in T . Then Claim 42 from the proof of Lemma 41 can be argued exactly in the same manner. We recall it for convenience and refer the reader to the proof of Claim 42 for the argumentation.

Claim 54 (Claim 42, restated). *Take any connected component T' of $T - M$ and let $X_{T'}$ be the union of the bags at the nodes of T' . Then*

$$\|G[X_{T'}]\| \leq 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}.$$

We denote $c_1 := 2b_{\mathcal{F}} \cdot c_{\mathcal{F}}$.

For a node $t \in M$, let $F(t)$ be the set of those edges of F that are incident to some vertex of X_t . A standard hand-shaking argument shows that

$$\sum_{t \in M} |F(t)| \leq 2|F|. \tag{14}$$

Let $T_1, \dots, T_{p(t)}$ be the connected components of $T - t$ which contain some other nodes of M and let $S_1, \dots, S_{q(t)}$ be those with none. Again, by X_{T_i} , resp. X_{S_i} , we denote the union of bags at the nodes of T_i , respectively S_i .

Consider the forest with vertex set M defined as follows: put an edge between two nodes of M if there is a component of $T - M$ that neighbors both of them. Observe that $p(t)$ is the degree of t in this forest. Therefore, as there are at most $|M| - 1$ edges in any forest on $|M|$ vertices, we infer that

$$\sum_{t \in M} p(t) \leq 2(|M| - 1). \quad (15)$$

Define $f(t) := |F(t)| + b_{\mathcal{F}} \cdot p(t) + d_{\mathcal{F}}$. By (14) and (15) we conclude that

$$\sum_{t \in M} f(t) \leq 2|F| + b_{\mathcal{F}} \cdot 2(|M| - 1) + |M| \cdot d_{\mathcal{F}} \leq d_1 \cdot |F|, \quad \text{for } d_1 := 2 + 8b_{\mathcal{F}} + 4d_{\mathcal{F}}. \quad (16)$$

We proceed similarly as in the proof of Lemma 41. Consider an arbitrary $t \in M$. By Corollary 21, at least $q(t) - (2b_{\mathcal{F}} + 1)$ of the trees $S_1, \dots, S_{q(t)}$ are connected to t via a neat adhesion; let $I_1 \subseteq \{1, 2, \dots, q(t)\}$ be the set of their indices. That is, for each $i \in I_1$ we have that $|\delta(X_{S_i})| \leq 2$ and $N(X_{S_i})$ is a subset of X_t of size at most 2. Since there are at most $b_{\mathcal{F}}^2$ subsets of X_t of size at most 2, at least $(q(t) - 2b_{\mathcal{F}} - 1)/b_{\mathcal{F}}^2$ of subtrees $\{S_i\}_{i \in I_1}$ have the same neighborhood $N(X_{S_i}) = U$, for some $U \subseteq X_t$ of size at most 2; let $I_2 \subseteq I_1$ be the set of their indices. By Claim 54, for each $i \in I_2$ we have that X_{S_i} induces in G a subgraph with at most c_1 edges. It follows that there are at most $c_2 := (2c_1 + 1)^{c_1+2}$ possible isomorphism types for graphs $G[X_{S_i} \cup U]$ for $i \in I_2$ (considering isomorphisms that fix U). Therefore, if $(q(t) - 2b_{\mathcal{F}} - 1)/(b_{\mathcal{F}}^2 \cdot c_2) \geq f(t)$, then there is a subset $I_3 \subseteq I_2$ of size at least $f(t)$ for which sets $\{X_{S_i}\}_{i \in I_3}$ are elements of a single bouquet \mathcal{X} attached to U .

Let $\Delta \subseteq E(G)$ be the set comprising of $F(t)$ and the adhesions corresponding to the edges connecting subtrees $T_1, \dots, T_{p(t)}$ with t in T . Then Δ separates in G the vertices of

$$Z := X_t \cup X_{S_1} \cup \dots \cup X_{S_{q(t)}}$$

from the rest of the graph; that is, all the edges between Z and $V(G) \setminus Z$ are contained in Δ . Since Δ contains $F(t)$ and none of the sets X_{S_i} is incident to any edge of F , we infer that $G[Z] - \Delta$ is \mathcal{F} -free. Hence all the components of $G - \Delta$ that contain some vertex of the bouquet \mathcal{X} are \mathcal{F} -free. By applying Lemma 51, we infer that either

$$|\mathcal{X}| \leq d_{\mathcal{F}} + |\Delta| \leq d_{\mathcal{F}} + |F(t)| + b_{\mathcal{F}} \cdot p(t) = f(t), \quad (17)$$

or all except $f(t)$ elements of the bouquet can be deleted to obtain a strictly smaller subgraph $G' \subsetneq G$ with $\text{OPT}(G') = \text{OPT}(G)$. Hence, if (17) does not hold, then the algorithm can output G' and terminate. We can thus conclude the proof, unless for all $t \in M$ the following holds:

$$(q(t) - 2b_{\mathcal{F}} - 1)/(b_{\mathcal{F}}^2 \cdot c_2) \leq f(t),$$

or equivalently

$$q(t) \leq f(t) \cdot c_2 \cdot b_{\mathcal{F}}^2 + 2b_{\mathcal{F}} + 1. \quad (18)$$

We henceforth assume that this is the case.

Similarly, if there are more than $f(t)$ edges with the same pair of endpoints $U \subseteq X_t$, by Lemma 52 all but $f(t)$ of them can be deleted to obtain a strictly smaller subgraph $G' \subsetneq G$ with $\text{OPT}(G') = \text{OPT}(G)$. We can thus conclude the proof unless, for each $t \in M$, there is no group of

more than $f(t)$ parallel edges connecting the same pair of endpoints in X_t . We henceforth assume that the latter alternative is the case. Since $|X_t| \leq \text{width}'(\mathcal{T}) \leq b_{\mathcal{F}}$, we have the following for each $t \in M$:

$$\|G[X_t]\| \leq b_{\mathcal{F}}^2 \cdot f(t). \quad (19)$$

We proceed with analyzing the size of the instance, with the goal of showing that it is bounded linearly in $|F|$. By (16) and (18) we have

$$\sum_{t \in M} q(t) \leq c_2 \cdot b_{\mathcal{F}}^2 \cdot d_1 \cdot |F| + |M| \cdot (2b_{\mathcal{F}} + 1) \leq c_3 \cdot |F|, \quad \text{for } c_3 := c_2 \cdot b_{\mathcal{F}}^2 \cdot d_1 + 4(2b_{\mathcal{F}} + 1).$$

That is, the total number of components of $T - M$ with exactly one neighbor in T is at most $c_3 \cdot |F|$. As we argued before, the number of components of $T - M$ with exactly two neighbors in T is at most $|M| - 1$. Hence,

$$T - M \text{ has at most } c_3 \cdot |F| + |M| - 1 \text{ connected components in total.} \quad (20)$$

We now examine the edges of G . Every edge of G is either:

- (i) in F , or
- (ii) in $G[X_t]$ for some $t \in M$, or
- (iii) in $G[X_{T'}]$ for a component T' of $T - M$, or
- (iv) in an adhesion corresponding to an edge of T connecting a connected component of $T - M$ with a node of M .

The total number of edges of each of these types is respectively bounded by:

- (i) $|F|$,
- (ii) $b_{\mathcal{F}}^2 \cdot d_1 \cdot |F|$ (by (16) and (19)),
- (iii) $c_1 \cdot (c_3 \cdot |F| + |M| - 1)$ (by Claim 54 and (20)), and
- (iv) $2b_{\mathcal{F}} \cdot (c_3 \cdot |F| + |M| - 1)$ (by $\text{width}'(\mathcal{T}) \leq b_{\mathcal{F}}$, (20), and the fact that each component of $T - M$ neighbors at most two nodes of M).

Therefore, we conclude that

$$\|G\| \leq c \cdot |F| \quad \text{for } c := 1 + (c_1 + 2b_{\mathcal{F}}) \cdot (c_3 + 4) + b_{\mathcal{F}}^2 \cdot d_1$$

as required. \square

We are ready to conclude the description of our kernelization algorithm, that is, to prove Theorem 3. For convenience, we recall its statement and adjust it to the current notation.

Theorem 55 (Theorem 3, reformulated). *There is an algorithm that, given an instance (G, k) of \mathcal{F} -IMMERSION DELETION, runs in time $\mathcal{O}(\|G\|^4 \log \|G\| \cdot |G|^3)$ and either correctly concludes that (G, k) is a NO-instance, or outputs an equivalent instance (G', k) such that G' has at most $c_{\text{ker}} \cdot k$ edges, where c_{ker} is a constant depending on \mathcal{F} only.*

Proof. We fix the constant c_{ker} as

$$c_{\text{ker}} := c_{\text{apx}} \cdot c,$$

where c is the constant given by Lemma 53. Let (G, k) be the input instance.

We first apply the algorithm of Lemma 37, which runs in time $\mathcal{O}(\|G\|^2 \log \|G\| \cdot |G|^3)$ and yields a new graph G' such that $\|G'\| \leq \|G\|$, $\text{OPT}(G') = \text{OPT}(G)$ and each connected component of G' has no excessive protrusion. From now on, we work on the graph G' instead of G . If we already have that $\|G'\| \leq c \cdot k$, then we can simply output (G', k) , so assume that this is not the case.

Apply the approximation algorithm of Lemma 50 to G' , yielding in time $\mathcal{O}(\|G\|^3 \log \|G\| \cdot |G|^3)$ a subset of edges F such that $G' - F$ is \mathcal{F} -free and $|F| \leq c_{\text{apx}} \cdot \text{OPT}(G')$. If $|F| > c_{\text{apx}} \cdot k$, then we can infer that $\text{OPT}(G) = \text{OPT}(G') > k$ and thus we terminate the algorithm by concluding that (G, k) is a NO-instance. Hence, assume otherwise, that $|F| \leq c_{\text{apx}} \cdot k$. Because $\|G'\| > c_{\text{apx}} \cdot c \cdot k$, we have that

$$\|G'\|/|F| > c. \tag{21}$$

For each connected component H of G' , let $F_H = F \cap E(H)$. Obviously $H - F_H$ is \mathcal{F} -free, as it is an induced subgraph of $G' - F$. By (21), there exists some connected component H of G' for which $\|H\|/|F_H| > c$, that is, $\|H\| > c \cdot |F_H|$. Therefore, as H is connected and has no excessive protrusions (as a connected component of G'), we can apply the algorithm of Lemma 53 to H . This application takes $\mathcal{O}(\|H\| \cdot |H|^2)$ time and outputs a subgraph H' of H with $\|H'\| < \|H\|$ and $\text{OPT}(H') = \text{OPT}(H)$. We can now replace H with H' in G' , thus obtaining a new graph G'' , and restart the whole algorithm on the instance (G'', k) . Since $\text{OPT}(H') = \text{OPT}(H)$ and every graph of \mathcal{F} is connected, it follows that also $\text{OPT}(G') = \text{OPT}(G)$ and, hence, the instance (G'', k) is equivalent to (G, k) . Also, $\|H'\| < \|H\|$ implies $\|G''\| < \|G'\| \leq \|G\|$, so the number of edges is strictly smaller in the instance (G'', k) than in the original instance (G, k) .

We conclude that the algorithm will either terminate by concluding that (G, k) is a NO instance, or it will output (G', k) , provided $\|G'\| \leq c_{\text{ker}} \cdot k$, or it will restart on an equivalent instance (G'', k) with $\|G''\| < \|G\|$. The number of iterations is bounded by the number of edges in the original graph and each iteration takes $\mathcal{O}(\|G\|^3 \log \|G\| \cdot |G|^3)$ time, so the running time bound follows. \square

7 Bounding the size of the obstructions

In order to prove the second part of Theorem 3 it is enough to prove that, in the statement of the first part, the graph of the equivalent instance (G', k) is an immersion of G . To see this, assume that $H \in \mathcal{O}_k^{\text{im}} = \text{obs}_{\text{im}}(\mathcal{G}_{k, \mathcal{F}}^{\text{im}})$. Clearly, (H, k) is a NO-instance of \mathcal{F} -IMMERSION DELETION. If we run the kernelization algorithm on (H, k) the result should be a NO-instance (H', k) where H' is an immersion of H . As H is an immersion obstruction of $\mathcal{G}_{k, \mathcal{F}}^{\text{im}}$, for every proper immersion of H , the pair (H', k) should be a YES-instance. Therefore $H' = H$ and, according to the first statement of Theorem 3, H has a linear, on k , number of edges.

It remains now to modify the kernelization algorithm of Theorem 3 so that, when it runs with input (G, k) , it outputs a pair (G', k) where G' is an immersion of G . Recall that the algorithm, during its execution, either applies replacements of replaceable protrusions (i.e., $2b_{\mathcal{F}}$ -protrusions with more than $c_{\mathcal{F}}$ edges) with smaller ones (Lemma 23), or removes edges from thetas and buckets (Lemma 53). Therefore we need to modify the protrusion replacement in Lemma 23 so that G' is an immersion of G . Before we explain this modification, we first need some definitions.

Given two r -boundaried graphs $\mathbb{G} = (G, (u_1, \dots, u_r))$ and $\mathbb{H} = (H, (v_1, \dots, v_r))$, we say that \mathbb{H} is a *rooted immersion* of \mathbb{G} if H is an immersion of G where the corresponding mapping μ_V maps v_i to u_i for every $i \in \{1, \dots, r\}$. We next argue that for every r , rooted r -boundaried graphs are

well-quasi-ordered with respect to the rooted-immersion relation. Indeed, Robertson and Seymour proved in [39] that the set of all colored (by a bounded number of colors) graphs is well-quasi-ordered with respect to the colored immersion relation (here the function μ_V should additionally map vertices to vertices of the same color). By assigning to the boundaries of the r -boundaried graphs r different colors and using one more color for their non-boundary vertices, we deduce that r -boundaried graphs are well-quasi-ordered under the rooted immersion relation.

Let \mathcal{B}_r be the set of all r -boundaried graphs. The set \mathcal{B}_r has a partition $\mathcal{C}^{(r)} = \{\mathcal{C}_1^{(r)}, \dots, \mathcal{C}_{q_r}^{(r)}\}$ such that two r -boundaried graphs belong in the same set if and only if they have the same signature. According to Corollary 29, $q_r \leq 2^{2^{2^{\text{poly}(r, \text{MAX}_{\mathcal{F}})}}}$. For every $i \in \{1, \dots, q_r\}$, let $\bar{\mathcal{C}}_i$ be the set of rooted immersion-minimal elements of \mathcal{C}_i . As r -rooted graphs are well-quasi-ordered under rooted-immersions, we have that $\bar{\mathcal{C}}_i^{(r)}$ is a finite set. We now consider the following set

$$\mathcal{R}_{\mathcal{F}} = \bigcup_{r \leq 2b_{\mathcal{F}}} \bigcup_{i \in \{1, \dots, q_r\}} \bar{\mathcal{C}}_i^{(r)}.$$

Notice that, for each $r \in \{1, \dots, 2b_{\mathcal{F}}\}$, each r -boundaried graph \mathbb{H} belongs in some, say $\mathcal{C}_i^{(r)}$, of the classes in $\mathcal{C}^{(r)}$, therefore it should contain as a rooted immersion some of the rooted graphs in $\bar{\mathcal{C}}_i^{(r)}$. Let $c_{\mathcal{F}}^*$ be the maximum number of edges in an r -boundaried graph of $\mathcal{R}_{\mathcal{F}}$. This implies that every r -boundaried graph \mathbb{H} of more than $c_{\mathcal{F}}^*$ edges can be replaced by an equivalent (i.e., one with the same signature) r -boundaried graph \mathbb{H}' that belongs in $\mathcal{R}_{\mathcal{F}}$ and is a rooted immersion of \mathbb{H} . Therefore, if (G, k) is an instance of \mathcal{F} -IMMERSION DELETION, \mathbb{H} is an $2b_{\mathcal{F}}$ -protrusion of G of more than $c_{\mathcal{F}}^*$ edges, and $G = \mathbb{F} \oplus \mathbb{H}$, then (G', k) is an instance equivalent to (G, k) where $G' = \mathbb{F} \oplus \mathbb{H}'$. Moreover, as \mathbb{H}' is a rooted immersion of \mathbb{G} , it follows that G' is an immersion of G as required.

Notice that the above argumentation does not give any way to compute $c_{\mathcal{F}}^*$ as the, inherently non-constructive, proof in [39] does not provide any way to compute a bound to the size of the graphs in $\bar{\mathcal{C}}_i^{(r)}$ (it only says that $|\bar{\mathcal{C}}_i^{(r)}|$ is a finite number). We wish to report that it is actually possible to prove a constructive version of the second statement of Theorem 3. This proof is postponed in later versions of this paper, as it resides on results that are currently under development.

8 Conclusions

In this work we have proved that the protrusion machinery, introduced in [4, 5, 19], can be applied to immersion-related problems in a similar manner as to minor-related problems. In particular, we have given a constant-factor approximation algorithm and a linear kernel for the \mathcal{F} -IMMERSION DELETION problem, which on one hand mirrors and on the other surpasses the results of Fomin et al. [19] for \mathcal{F} -MINOR DELETION. Namely, while the exponent of the polynomial bounding the kernel size provably has to depend on the family \mathcal{F} in the minor setting [23], in the immersion setting we were able to give a linear kernel, with only the multiplicative constant depending on \mathcal{F} . We consider this apparent difference of complexity interesting and worth studying further.

The immediate next goal is to lift the technical assumption that all graphs from \mathcal{F} are connected. While this assumption plays an important role in several of our proofs, we expect that it is not necessary and can be lifted using the techniques of Kim et al. [30] or Fomin et al. [18, 19] that worked in the minor setting. In fact, a constant-factor approximation, without the assumption on the connectivity of \mathcal{F} , can easily be obtained in the following way, as in the full version of the work of Fomin et al. [18]. Since Theorem 8 works just as well in the case of \mathcal{F} containing disconnected graphs, a set of edges whose deletion makes a graph \mathcal{F} -free also makes it a graph tree-cut width bounded

by some constant $a_{\mathcal{F}}$. Thus, $\text{OPT}_{\mathcal{F}}(G)$ is not smaller than the optimum size of a set of edges whose deletion turns G into a graph of tree-cut width at most $a_{\mathcal{F}}$. Tree-cut width is a graph parameter satisfying the conditions of Corollary 5, hence given a graph G , we can construct in polynomial time a set of edges $F \subseteq E(G)$ of size at most a constant factor larger than $\text{OPT}_{\mathcal{F}}(G)$, such that $G - F$ has tree-cut width at most $a_{\mathcal{F}}$. A standard application of the optimization variant of Courcelle’s theorem, due to Arnborg et al. [1], then gives a set $F' \subseteq E(G - F)$ such that $G - F - F'$ is \mathcal{F} -free and $|F'| = \text{OPT}_{\mathcal{F}}(G - F) \leq \text{OPT}_{\mathcal{F}}(G)$. Hence by outputting $F \cup F'$, one achieves a constant factor approximation for \mathcal{F} -IMMERSION DELETION. For the linear kernel for \mathcal{F} -IMMERSION DELETION, we so far do not see how to avoid the assumption on the connectivity of \mathcal{F} .

We believe that an important conceptual insight that is given by this paper is the confirmation of usefulness of the notions of tree-cut width and tree-cut decompositions. Our work, together with a few other recent ones [22, 29, 33, 42], shows that tree-cut width is often the right parameter to study in the context of problem related to immersions and edge-disjointness, and plays a similar role as treewidth for minors and vertex-disjointness. We expect that more results of this kind will appear in future.

Clearly, the remaining insisting problem on the study of \mathcal{F} -IMMERSION DELETION problem is to consider cases where none of the graphs in \mathcal{F} is planar subcubic. This comes as an analogue to instantiations of the \mathcal{F} -MINOR DELETION problem where \mathcal{F} contain only non-planar graphs. In both cases the existence of a polynomial kernel can be seen as a major challenge in parameterized algorithms. Especially, for \mathcal{F} -IMMERSION DELETION, further advances are necessary on the structure of graphs excluding non-planar or non-subcubic immersions. While some results in this direction have appeared in [2, 15, 33, 42], it is still unclear whether the current combinatorial insight can produce general algorithmic results on \mathcal{F} -IMMERSION DELETION.

Acknowledgements. The authors wish to thank an anonymous referee for suggesting a more direct approach to finding excessive protrusions, as well as Ignasi Sau, Petr Golovach, Eun Jung Kim, and Christophe Paul for preliminary discussions on the \mathcal{F} -IMMERSION DELETION problem.

References

- [1] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.
- [2] R. Belmonte, A. Giannopoulou, D. Lokshtanov, and D. M. Thilikos. The Structure of W_4 -Immersion-Free Graphs. *ArXiv e-prints 1602.02002*, Feb. 2016.
- [3] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.
- [4] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *Proceedings of FOCS 2009*, pages 629–638. IEEE Computer Society, 2009.
- [5] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (meta) kernelization. *ArXiv e-prints 0904.0727*, 2009.
- [6] H. D. Booth, R. Govindan, M. A. Langston, and S. Ramachandramurthi. Fast algorithms for K_4 immersion testing. *J. Algorithms*, 30(2):344–378, 1999.

- [7] D. Chatzidimitriou, J.-F. Raymond, I. Sau, and D. M. Thilikos. An $O(\log OPT)$ -approximation for covering and packing minor models of θ_r . *ArXiv e-prints 1510.03945*, Oct. 2015.
- [8] C. Chekuri and J. Chuzhoy. Polynomial bounds for the grid-minor theorem. In *Proceedings of STOC 2014*, pages 60–69. ACM, 2014.
- [9] R. H. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *Proceedings of FOCS 2012*, pages 460–469. IEEE Computer Society, 2012.
- [10] J. Chuzhoy. Excluded grid theorem: Improved and simplified. In *Proceedings of STOC 2015*, pages 645–654. ACM, 2015.
- [11] J. Chuzhoy. Improved Bounds for the Excluded Grid Theorem. *ArXiv e-prints 1602.02629*, Feb. 2016.
- [12] B. Courcelle. The Monadic Second-Order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.
- [13] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.
- [14] M. Devos, Z. Dvořák, J. Fox, J. McDonald, B. Mohar, and D. Scheide. A minimum degree condition forcing complete graph immersion. *Combinatorica*, 34(3):279–298, 2014.
- [15] Z. Dvořák and P. Wollan. A structure theorem for strong immersions. *Journal of Graph Theory*, 2015. To appear.
- [16] Z. Dvořák and L. Yepremyan. Complete graph immersions and minimum degree. *ArXiv e-prints 1512.00513*, Dec. 2015.
- [17] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.
- [18] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh. Planar F-Deletion: Approximation and optimal FPT algorithms. *ArXiv e-prints 1204.4230*, Oct. 2012.
- [19] F. V. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh. Planar \mathcal{F} -deletion: Approximation, kernelization and optimal FPT algorithms. In *Proceedings of FOCS 2012*, pages 470–479. IEEE Computer Society, 2012.
- [20] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proceedings of SODA 2010*, pages 503–510. SIAM, 2010.
- [21] H. N. Gabow and R. E. Tarjan. A linear-time algorithm for a special case of disjoint set union. In *Proceedings of STOC 1983*, pages 246–251. ACM, 1983.
- [22] R. Ganian, E. J. Kim, and S. Szeider. Algorithmic applications of tree-cut width. In G. F. Italiano, G. Pighizzini, and D. Sannella, editors, *Proceedings of MFCS 2015*, volume 9235 of *Lecture Notes in Computer Science*, pages 348–360. Springer, 2015.
- [23] A. C. Giannopoulou, B. M. P. Jansen, D. Lokshtanov, and S. Saurabh. Uniform kernelization complexity of hitting forbidden minors. In *Proceedings of ICALP 2015*, volume 9134 of *Lecture Notes in Computer Science*, pages 629–641. Springer, 2015.

- [24] A. C. Giannopoulou, O.-j. Kwon, J.-F. Raymond, and D. M. Thilikos. Packing and Covering Immersion Models of Planar Subcubic Graphs. *ArXiv e-prints 1602.04042*, Feb. 2016.
- [25] A. C. Giannopoulou, I. Salem, and D. Zoros. Effective computation of immersion obstructions for unions of graph classes. *J. Comput. Syst. Sci.*, 80(1):207–216, 2014.
- [26] R. Govindan and S. Ramachandramurthi. A weak immersion relation on graphs and its applications. *Discrete Mathematics*, 230(1–3):189 – 206, 2001.
- [27] M. Grohe, K. Kawarabayashi, D. Marx, and P. Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of STOC 2011*, pages 479–488. ACM, 2011.
- [28] A. V. Iyer, H. D. Ratliff, and G. Vijayan. On an edge ranking problem of trees and graphs. *Discrete Applied Mathematics*, 30(1):43–52, 1991.
- [29] E. Kim, S. Oum, C. Paul, I. Sau, and D. M. Thilikos. An FPT 2-approximation for tree-cut decomposition. In *Proceedings of WAOA 2015*, pages 35–46, 2015.
- [30] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. In *Proceedings of ICALP 2013*, volume 7965 of *Lecture Notes in Computer Science*, pages 613–624. Springer, 2013.
- [31] T. W. Lam and F. L. Yue. Edge ranking of graphs is hard. *Discrete Applied Mathematics*, 85(1):71–86, 1998.
- [32] D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.
- [33] D. Marx and P. Wollan. Immersions in highly edge connected graphs. *SIAM J. Discrete Math.*, 28(1):503–520, 2014.
- [34] M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of FOCS 1995*, pages 182–191. IEEE Computer Society, 1995.
- [35] R. Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.
- [36] N. Robertson and P. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.
- [37] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.
- [38] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner’s conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.
- [39] N. Robertson and P. D. Seymour. Graph minors XXIII. Nash-Williams’ immersion conjecture. *J. Comb. Theory, Ser. B*, 100(2):181–205, 2010.
- [40] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.
- [41] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *J. Algorithms*, 56(1):1–24, 2005.

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