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

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

1 **Technische Universität Berlin, Berlin, Germany**
archontia.giannopoulou@tu-berlin.de
2 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl
3 **Institute of Informatics, University of Warsaw, Warsaw, Poland; and AlGCo project team, CNRS, LIRMM, Montpellier, France**
jean-florent.raymond@mimuw.edu.pl
4 **AlGCo project team, CNRS, LIRMM, Montpellier, France; and Department of Mathematics, National and Kapodistrian University of Athens, Athens, Greece**
sedthilk@thilikos.info
5 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
m.wrochna@mimuw.edu.pl

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 an 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. Our findings mirror those of Fomin et al. [FOCS 2012], who obtained similar 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. 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.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

* The full version of this paper can be found as an arxiv preprint [19], https://arxiv.org/abs/1609.07780.

† This work was done while A. C. Giannopoulou was holding a post-doc position at Warsaw Center of Mathematics and Computer Science and she has also 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). Mi. Pilipczuk and M. Wrochna are supported by the Polish National Science Center grant UMO-2013/11/D/ST6/03073. J-F. Raymond is supported by the Polish National Science Center grant UMO-2013/11/N/ST6/02706. D. Thilikos is supported by project DEMOGRAPH (ANR-16-CE40-0028).

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 57; pp. 57:1–57:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases Kernelization, Approximation, Immersion, Protrusion, Tree-cut width

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.57

1 Introduction

On the $\mathcal{F}$-Minor Deletion problem. Let us fix a finite family of graphs $\mathcal{F}$. A graph is called *$\mathcal{F}$-minor-free* if it does not contain any graph from $\mathcal{F}$ as a minor. Given a class of graphs $\mathcal{G}$, we denote by $\mathbf{obs}_{\mathrm{mn}}(\mathcal{G})$ the set of minor-minimal graphs not in $\mathcal{G}$. The celebrated Graph Minors Theorem [32] states that for $\mathcal{G}$ closed under taking minors, the set $\mathbf{obs}_{\mathrm{mn}}(\mathcal{G})$ is finite. In other words, $\mathcal{G}$ is characterized by a finite set of minor-obstructions, as $\mathcal{G}$ is exactly the class of $\mathcal{F}$-minor-free graphs, for $\mathcal{F} = \mathbf{obs}_{\mathrm{mn}}(\mathcal{G})$. Hence, studying 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. [14] performed an in-depth study of the following parameterized[1] problem, named $\mathcal{F}$-Minor Deletion[2]: *Given a graph G and an integer parameter k, decide whether one can 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 $\mathcal{G}^{\mathrm{mn}}_{k,\mathcal{F}}$ consisting of the graphs in the YES-instances (G,k) of $\mathcal{F}$-Minor Deletion, is closed under taking of minors. By the meta-algorithmic consequences of the Graph Minors series of Robertson and Seymour [32, 30], 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 a stimulating project in parameterized algorithm design. So far, it has been focused on problems generated by minor-closed graph classes.

The goal of Fomin et al. [14] 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. [14] gives the following:

- a randomized constant-factor approximation running in time $\mathcal{O}(nm)$;
- 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 $\mathcal{F}$;
- an FPT-algorithm for $\mathcal{F}$-Minor Deletion in time $2^{\mathcal{O}(k)} \cdot n$ (note this originally required that all graphs from $\mathcal{F}$ be connected; Kim et al. [24] showed how to lift this assumption);
- a proof that every graph in $\mathbf{obs}_{\mathrm{mn}}(\mathcal{G}^{\mathrm{mn}}_{k,\mathcal{F}})$ has at most $k^{c_{\mathcal{F}}}$ vertices, for some constant $c_{\mathcal{F}}$ that depends (non-constructively) on $\mathcal{F}$.

The assumption that $\mathcal{F}$ contains at least one planar graph is crucial for the approach of Fomin et al. [14]. Namely, from the Excluded Grid Minor Theorem of Robertson and Seymour [31] 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

[1] A *parameterized* problem can be seen as a subset of $\Sigma^* \times \mathbb{N}$. For graph problems, the string x in an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ usually encodes a graph G. An FPT-algorithm for the problem is then an algorithm working in $f(k) \cdot |x|^{O(1)}$ time. See [12, 8, 29] for more on parameterized algorithms and complexity.

[2] Fomin et al. used the name $\mathcal{F}$-Deletion. We write $\mathcal{F}$-Minor Deletion instead for clarity.

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. [14] apply protrusion-based techniques that originate in the work on *meta-kernelization* [3, 15]. 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 *protrusions*), which can then 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. [14] 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. [17] that in general this growth is unavoidable: unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, for every constant η, the Treewidth-η Deletion problem (delete k vertices to get a graph of treewidth at most η) has no kernel with $\mathcal{O}(k^{\eta/4-\epsilon})$ vertices for any $\epsilon > 0$. Since graphs of treewidth η can be characterized by a finite set of forbidden minors $\mathcal{F}_\eta$, 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. [17], such bounds can be achieved for some specific problems, 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 vertices of H to pairwise different vertices of G and from edges of H to pairwise *edge-disjoint paths* connecting the images of its endpoints[3]. 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 [33] proved that graphs are well-quasi-ordered under the immersion order as well. This implies that for every graph class $\mathcal{G}$ that is closed under taking immersions, the set $\mathbf{obs}_{\mathrm{im}}(\mathcal{G})$ containing immersion minimal graphs that do not belong in $\mathcal{G}$, is finite (we call $\mathbf{obs}_{\mathrm{im}}(\mathcal{G})$ the *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 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, 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 [28, 10, 16, 25, 18, 20, 36, 4, 9, 1, 21, 11] both from the combinatorial and the algorithmic point of view. Most importantly for us, Wollan proved in [36] an analogue 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 [36].

▶ **Theorem 1** ([18]). *For every planar subcubic graph H, there exists a constant a_H such that every graph not containing 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

[3] In 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.

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 [36]. 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 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 [36], the new parameter gathered substantial interest from the algorithmic and combinatorial community [16, 28, 18, 25]. 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-analogue" of treewidth. In particular, given the tree-cut width bound of Theorem 1 and the general approach of Fomin et al. [14] to $\mathcal{F}$-Minor Deletion, it is natural to ask whether the same kind of results can be obtained for immersions where the edge removals are considered instead of vertex removals. 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. Notice that when $\mathcal{F} = \{K_2\}$ and $\mathcal{F} = \{K_3\}$, the problem of computing the minimum size of such a set of edges can be solved in polynomial time, while it is NP-hard when $\mathcal{F} = \{K_4^-\}$ (see e.g. [5]).

Parallel to the case of $\mathcal{F}$-Minor Deletion, for every fixed k, the graph class $\mathcal{G}_{k,\mathcal{F}}^{\mathrm{im}}$ consisting of the graphs in the YES-instances (G, k) of $\mathcal{F}$-Immersion Deletion is closed under taking of immersions[4], therefore $\mathcal{O}_k^{\mathrm{im}} = \mathbf{obs}_{\mathrm{im}}(\mathcal{G}_{k,\mathcal{F}}^{\mathrm{im}})$ is a finite set, by the well-quasi-ordering of graphs under immersions [33]. Together with the immersion-testing algorithm of Grohe et al. [22], 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 various instantiations of $\mathcal{F}$. More concretely, is it possible to extend the general framework of Fomin et al. [14] to obtain efficient approximation, kernelization, and FPT algorithms also for $\mathcal{F}$-Immersion Deletion? Theorem 1 suggests that the suitable analogue 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). *Let $\mathcal{F}$ be a finite family of connected graphs with at least one member being planar and subcubic. Given a graph G, in $\mathcal{O}(\|G\|^3 \log \|G\| \cdot |G|^3)$ time one can output 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.*

The constant-factor approximation can be generalized to work when $\mathcal{F}$ contains disconnected graphs as well, using the approach of Fomin et al. [14, 13].

[4] Notice that if we consider deletion of vertices instead of edges, then the graph class $\mathcal{G}_k^{\mathrm{im}}$ 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.

▶ **Theorem 3** (Linear kernelization and obstructions). *Let $\mathcal{F}$ be a finite family of connected graphs with at least one member being planar and subcubic. Given an instance (G,k) of $\mathcal{F}$-*Immersion Deletion*, in time $\mathcal{O}(\|G\|^4 \log \|G\| \cdot |G|^3)$ one can output an equivalent instance (G',k) with $\|G'\| \leq c_{\mathrm{ker}} \cdot k$, for some constant c_{ker} depending on $\mathcal{F}$ only. Moreover, every graph in $\mathcal{O}_k^{\mathrm{im}}$ has at most $c_{\mathcal{F}} \cdot k$ edges (for a constant $c_{\mathcal{F}}$ non-constructively depending on $\mathcal{F}$).*

Thus, Theorems 2 and 3 mirror the approximation and kernelization results and the obstruction bounds of Fomin et al. [14]. However, this mirroring is not exact as we 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 $\mathcal{O}(c_{\mathrm{fpt}}^k + \|G\|^4 \log \|G\| \cdot |G|^3)$ 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. [14] (see also [3]). We first define protrusions suited for the problem of our interest. In fact, our protrusions can be seen as the edge-analogue of those introduced in [14] (as in [5]). 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. [14]. 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 [27] and the randomized contractions technique of Chitnis et al. [6]. 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 $G - F_{\mathrm{apx}}$, 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 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 hence a simple amortization arguments shows that the total size of bouquets is linear in $|F_{\mathrm{apx}}|$, so also linear in terms of the optimum.

We remark that this part of the reasoning (the above amortization argument in particular) are fully new contributions of this work. These deviate significantly from arguments by Fomin et al. [14], which aimed at a obtaining a polynomial kernel only, instead of linear. Also, we remark that, contrary to the work of Fomin et al. [14], all our algorithms are deterministic.

For the second part of Theorem 3, we show that protrusion replacements can be realized as immersions of the original graph. This implies that, in the equivalent instance (G', k) produced by our kernelization algorithm, the graph G' is an immersion of G. Therefore any immersion-obstruction of $\mathcal{G}^{\mathrm{im}}_{k-1,\mathcal{F}}$ must already have a linear, in k, number of edges.

Application: immersion-closed parameters. We would like to highlight one 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 $\mathbb{N}$. We shall say that $\mathbf{p}$ is *closed under immersion* if $\mathbf{p}(H) \leq \mathbf{p}(G)$ whenever H is an immersion of G; $\mathbf{p}$ is *closed under disjoint union* if $\mathbf{p}(G_1 \uplus G_2) = \max(\mathbf{p}(G_1), \mathbf{p}(G_2))$ for any graphs G_1, G_2; here, $\uplus$ denotes disjoint union.

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 $\mathbf{p}_r(G) = \min\{k \mid \exists S \subseteq E(G) : |S| \leq k \wedge \mathbf{p}(G \setminus S) \leq r\}$ and $\mathcal{G}_{k,\mathbf{p}_r} = \{G \mid \mathbf{p}_r(G) \leq k\}$. Then the following meta-result can be derived from Theorems 2 and 3 and the fact that immersion is a well-quasi-order; a proof can be found in the full version of the paper.

▶ **Theorem 4.** *Let $\mathbf{p}$ be a graph parameter that is closed under immersion and under disjoint union and moreover is large on the class of walls*[5]*. 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 $\mathbf{obs}_{\mathrm{im}}(\mathcal{G}_{k,\mathbf{p}_r})$ has at most $c_r \cdot k$ edges.*

Natural parameters that satisfy the prerequisites of Theorem 4 include cutwidth, carving width, tree-cut width, and edge ranking; see e.g. [34, 35, 36, 26, 23] for more details. Theorem 4 mirrors a corollary by Fomin et al. [14] for the Treewidth-η Deletion problem asserting a constant-factor approximation, a polynomial kernel, a polynomial bound for the corresponding minor-obstruction set, and a single-exponential FPT algorithm, for every constant η.

Organization. This extended abstract focuses on sketching the proofs of Theorems 2 and 3. The proofs of statements marked with $\star$ are omitted and can be found in the full version [19].

2 Preliminaries

For a positive integer p, we denote $[p] = \{1, 2, \ldots, p\}$. 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*. Each edge connects two different vertices, called the *endpoints* of the edge (we do not allow loops). Note that there might be several edges (called *parallel edges*) between two vertices. An edge is *incident* to a vertex if it 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 of edges

[5] A 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.

$F \subseteq E(G)$, $G - F$ denotes the graph G with all edges from F removed. For two subsets $X, Y \subseteq V(G)$, not necessarily disjoint, $E_G(X,Y)$ denotes the set of edges of $E(G)$ of the form xy for some $x \in X$ and $y \in Y$. The *boundary* of X is $\delta_G(X) = E_G(X, V(G) \setminus X)$.

Tree-cut width. A *near-partition* of a set X is a family of (possibly empty) subsets $X_1, \ldots, X_k$ 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 connected graph G is a pair $(T, \mathcal{X})$ where T is a tree and $\mathcal{X} = \{X_t : t \in V(T)\}$ is a near-partition of the vertices of G. Sets $\{X_t : t \in V(T)\}$ are called the *bags* of the decomposition. For a subset $W \subseteq V(T)$, define X_W as $\bigcup_{t \in W} X_t$. By rooting T at some vertex, we can talk about a *rooted tree-cut decomposition*. When G is disconnected, a tree-cut decomposition is a forest consisting of one tree for each connected component.

For each edge $e = uv$ of T, $T - uv$ has exactly two components which we call T_{uv} and T_{vu}, that contain u and v respectively. Since $\mathcal{X}$ is a near-partition, sets $X_{V(T_{uv})}$ and $X_{V(T_{vu})}$ form a near-partition of $V(G)$ (provided G is connected). We define the *adhesion* of an edge $e = uv$ of T, denoted $\mathsf{adh}_{\mathcal{T}}(e)$, as the set $E_G(X_{V(T_{uv})}, X_{V(T_{vu})})$. We omit the subscript if $\mathcal{T}$ is clear from the context. An adhesion is *thin* if it has at most 2 edges, *bold* otherwise.

We now move to the definition of tree-cut width. In fact, we do not give the original definition (it can be found in the full version of the paper), but we instead give an alternative definition that is easier to handle. Let G be a graph and $(T, \mathcal{X} = \{X_t : t \in V(T)\})$ be a tree-cut decomposition of G. For a node t of T, let $w(t)$ be the number of edges incident to t that have bold adhesions. The *width′* of the decomposition, denoted $\mathsf{width}'(T, \mathcal{X})$, is equal to $\max\{\max_{e \in E(T)} |\mathsf{adh}(e)|, \max_{t \in V(T)} |X_t| + w(t)\}$. The *tree-cut width′* of G, denoted $\mathsf{tctw}'(G)$, is the minimum width' of a tree-cut decomposition of G. The standard tree-cut width of G, denoted $\mathsf{tctw}(G)$, is similarly defined as the minimum width of a tree-cut decomposition of G, where width is defined slightly differently. We prove that both notions are equivalent.

▶ **Lemma 5** (⋆). *For every graph G, it holds that $\mathsf{tctw}(G) = \mathsf{tctw}'(G)$. Moreover, given a tree-cut decomposition $\mathcal{T}$ of G, it holds that $\mathsf{width}(\mathcal{T}) \leq \mathsf{width}'(\mathcal{T})$, and a tree-cut decomposition $\mathcal{T}'$ with $\mathsf{width}'(\mathcal{T}') \leq \mathsf{width}(\mathcal{T})$ can be computed in time $\mathcal{O}(\|G\| \cdot |G|^2 \cdot \mathsf{width}(\mathcal{T}))$.*

Ganian et al. [16] showed that bounded tree-cut width implies bounded treewidth. Besides, Kim et al. [25] showed that the dependency cannot be improved to subquadratic.

▶ **Lemma 6** (see [16]). *For any graph G, $\mathsf{tw}(G) \leq 2\mathsf{tctw}(G)^2 + 3\mathsf{tctw}(G)$.*

In our algorithms we need to adjust tree-cut decompositions to our needs, and hence we define the notion of a *neat tree-cut decomposition*. A neat tree-cut decomposition is a rooted tree-cut decomposition $(T, \mathcal{X})$ of a connected graph G that has the following properties:

- For every $e = uv \in E(T)$, the graphs $G[X_{V(T_{uv})}]$ and $G[X_{V(T_{vu})}]$ are connected.
- Suppose t is a node of T with parent s, such that the adhesion of st is thin. Then $E_G(X_{V(T_{ts})}, X_{V(T_{t's})}) = \emptyset$ for every sibling t' of t.

The second property was used by Ganian et al. [16] under the name *niceness*. It appears that any tree-cut decomposition can be made neat without increasing the width by much; the proof follows closely the lines of [16, Lemma 1].

▶ **Lemma 7** (⋆). *Given a tree-cut decomposition of $\mathsf{width}' \leq k$, a neat tree-cut decomposition of the same graph with $\mathsf{width}' \leq k^2 + 1$ can be constructed in time $\mathcal{O}(\|G\| \cdot |G|^2 \cdot k^2)$.*

The following result shows why neat tree-cut decompositions are useful: if a node of a neat decomposition has many neighboring nodes, then all but a constant number of them are connected to it via very simple adhesions.

▶ **Lemma 8** (⋆). *Let G be a graph with a neat tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ satisfying $\mathsf{width}'(\mathcal{T}) \leq r$. Let $t \in V(T)$. Then for all but at most $2r + 1$ of the edges e of T incident to t, $\mathsf{adh}(e)$ is thin and all of its edges have an endpoint in the bag at t.*

Finally, we need that the tree-cut width of a graph can be computed efficiently. For this, we can use the following 2-approximation algorithm of Kim et al. [25]. We remark that the width of the decomposition yielded is measured in terms of $\mathsf{width}(\cdot)$ and not $\mathsf{width}'(\cdot)$, but the decomposition can be adjusted to have also width' bounded by $2r$ using Lemma 5.

▶ **Theorem 9** (see [25]). *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 $\mathsf{tctw}(w) > r$, or returns a tree-cut decomposition of G of width at most $2r$.*

For the whole paper we fix a finite family of graphs $\mathcal{F}$ with the following properties: all graphs of $\mathcal{F}$ are connected, and at least one is planar and subcubic. A graph G will be called $\mathcal{F}$-immersion-free, or $\mathcal{F}$-free for short, if G contains no graph from $\mathcal{F}$ as an immersion. By Theorem 1, the tree-cut width of an $\mathcal{F}$-free graph is bounded by a constant depending on $\mathcal{F}$ only, so by Lemma 6 also its treewidth is bounded by a constant. By combining this with Bodlaender's algorithm [2] and Courcelle's Theorem [7], we obtain the following:

▶ **Lemma 10** (⋆). *It can be checked in linear-time whether a given graph is $\mathcal{F}$-free.*

Moreover, by combining the bound on tree-cut width of an $\mathcal{F}$-free graph with Lemmas 5, 7 and Theorem 9, we obtain the following.

▶ **Lemma 11** (⋆). *There exists an algorithm that, given an $\mathcal{F}$-free graph G, runs in time $\mathcal{O}(\|G\| \cdot |G|^2)$ and computes a neat tree-cut decomposition $\mathcal{T}$ of G with $\mathsf{width}'(\mathcal{T}) \leq b_{\mathcal{F}}$, for some constant $b_{\mathcal{F}}$ depending on $\mathcal{F}$ only.*

For a graph G, by $\mathsf{OPT}(G)$ we denote the minimum number of edges that need to be deleted from G in order to obtain an $\mathcal{F}$-free graph.

3 Protrusions

Replacing protrusions. We now introduce the notion of a protrusion suited to the considered problem. In the sequel, we will only deal with $2b_{\mathcal{F}}$- and 2-protrusions, where $b_{\mathcal{F}}$ is the constructive bound on tctw' guaranteed by Lemma 11.

▶ **Definition 12.** 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.

As in [14], the base for our kernelization algorithm is *protrusion replacement*. That is, we iteratively find a protrusion X that is large but has small $\delta(X)$, and replace it with a smaller gadget X' that has the same behaviour. The following lemma formalizes this intuition.

▶ **Lemma 13** (⋆). *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 $\mathsf{OPT}(G) = \mathsf{OPT}(G')$ 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 $|F| \leq |F'|$.*

The proof of Lemma 13 follows closely the strategy used by Fomin et al. [14]: 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. Note that while a replacement table exists and can be hard-coded into the algorithm (as it depends on the fixed family $\mathcal{F}$ only), giving an explicit bound on the size of the graphs in it (and thus on $c_{\mathcal{F}}$ in Lemma 13) would require additional arguments. The details can be found in the full version of the paper.

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 13.

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 14.** 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 techniques of *important cuts*, introduced by Marx [27] (see also the exposition in [8, Chapter 8.2]) and of *randomized contractions* by Chitnis et al. [6] instead. 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 15** ($\star$). *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.*

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 say that *no component of G has an excessive protrusion* if 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 15 and replacing any obtained protrusion using Lemma 13, 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 16** (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 $\mathsf{OPT}(G) = \mathsf{OPT}(G')$, $\|G'\| \leq \|G\|$, 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.

4 Constant-factor approximation

It would be ideal if just applying the Exhaustive Protrusion Replacement (Lemma 16) 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 a 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 that reducing each to $\mathcal{O}(\mathsf{OPT})$ would be relatively easy, giving a quadratic kernel). 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*.

Bouquets. Let us define the following constant $d_{\mathcal{F}} := \max\{2b_{\mathcal{F}} \cdot c_{\mathcal{F}} + 2b_{\mathcal{F}},\ 3\mathsf{MAX}_{\mathcal{F}}\} + 1$ (where $\mathsf{MAX}_{\mathcal{F}} = \max_{H \in \mathcal{F}} \|H\|$). 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 17.** 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 18.** 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 $3 \cdot \max_{H \in \mathcal{F}} \|H\|$ 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 only those structures prevent the graph from being a linear kernel.

▶ **Lemma 19** ($\star$). *Let G be a connected graph without excessive protrusions, bouquets, or thetas. Then G is $\mathcal{F}$-free, or $\|G\| \leq c \cdot \mathsf{OPT}(G)$, for some constant c depending on $\mathcal{F}$ only.*

The proof of Lemma 19 goes roughly as follows. Take some optimum solution F. Then $G - F$ is $\mathcal{F}$-free, so, by Theorem 11, it has a neat tree-cut decomposition $(T, \mathcal{X})$ of width at most $b_{\mathcal{F}}$. For simplicity suppose that $G - F$ is connected, so that T is a tree (the proof in the general case is essentially the same). Let M_0 be the set of all vertices of T whose bags contain a vertex incident to an edge of F; then $|M_0| \leq 2|F|$. Compute the *lowest common ancestor closure* M of M_0: start with $M := M_0$, and iteratively add to M any lowest common

ancestor of two nodes of M that is not yet included. As in Fomin et al. [14], we have that $|M| \leq 2|M_0| \leq 4|F|$, and each component of $T - M$ is adjacent to at most two nodes of M.

Let us consider some connected component T' of $T - M$, and let $X_{T'}$ be the union of the bags at the nodes of T'. Suppose first that T' is adjacent to exactly two nodes of M; note that there are at most $|M| - 1$ such components T'. Then one can easily see that $\|G[X_{T'}]\| \leq 2b_{\mathcal{F}}c_{\mathcal{F}}$, because otherwise $X_{T'}$ would be an excessive protrusion. Here, we crucially use the first condition of the neatness of T to argue that $G - X_{T'}$ has at most two connected components. Hence, the total number of edges in graphs $G[X_{T'}]$ for such components T' is bounded by $(|M| - 1) \cdot 2b_{\mathcal{F}}c_{\mathcal{F}}$, which is linear in $|F| = \mathsf{OPT}(G)$.

We are left with considering components T' that are adjacent to exactly one node of M. We again have that $\|G[X_{T'}]\| \leq 2b_F c_{\mathcal{F}}$ for each such component T', because otherwise $X_{T'}$ would be an excessive protrusion. However, a priori we do not have any bound on the number of such components. Suppose for a moment that a large number of such components T' is adjacent to the same node $t \in M$. By Lemma 8, all but a constant number of them are connected to t via edges of the decomposition with thin adhesions. Moreover, for each of these adhesions, all the (at most 2) edges of the adhesion have both endpoints in the bag X_t. Since the size of X_t is bounded by a constant, and each $X_{T'}$ induces a graph of constant size, we can infer that there is a constant number of isomorphism types for graphs $G[X_{T'}]$, together with the choice of the attachment points in X_t. So if the number of the considered components T' was very large, then some of them would form a bouquet, a contradiction.

This shows that, in fact, the number of components T' adjacent to only one vertex of M is also bounded linearly in $|M|$, so also in $\mathsf{OPT}(G)$. The fact that G has no thetas is used to bound the number of edges contained in graphs induced by the bags of M. By combining all these bounds, we conclude the proof of Lemma 19.

Finding a constant-factor approximation piece by piece. To handle bouquets and thetas, we first show that they are disjoint, as otherwise they would constitute a large protrusion.

▶ **Lemma 20** ($\star$). *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.*

The following lemma is the crucial step for our approximation: we find a subset of edges Δ with a guarantee that a constant fraction of Δ is used in some optimum solution.

▶ **Lemma 21** ($\star$). *Given a connected graph G with no excessive protrusion that is not $\mathcal{F}$-free, one can find in in time $\mathcal{O}(|G|^3)$ a set $\Delta \subseteq E(G)$ such that (for some c depending on $\mathcal{F}$ only):* $\mathsf{OPT}(G - \Delta) < \mathsf{OPT}(G)$ *and* $|\Delta| \leq c \cdot (\mathsf{OPT}(G) - \mathsf{OPT}(G - \Delta))$.

The set Δ given by Lemma 21 is constructed as follows. First, we locate all thetas and bouquets in G; Lemma 20 ensures that they do not overlap. Δ is defined as the set of all edges of G, with the exception that in each bouquet and in each theta we exclude from Δ the edges of all but $d_{\mathcal{F}} - 1$ elements of the bouquet/theta. We show that the subgraph given by edges of Δ satisfies the assumptions of Lemma 19, thus bounding $|\Delta|$. The bound is linear in the size of the intersection of Δ with any optimal solution, allowing to conclude the claim.

To get a constant-factor approximation algorithm, we iteratively invoke the algorithm of Lemma 21 (extended to general graphs by considering each connected component separately). More precisely, we perform iteratively the following procedure, starting with $G_1 = G$. At step i, given a graph G_i, we first run the algorithm of Lemma 16 to remove excessive protrusions from G_i and obtain a new graph G_i'. Thus, in a sense, we reduce those parts

of the graph where no more edges need to be deleted. Then, we apply the algorithm of Lemma 21 to G'_i, thus finding a set Δ_i with the following property: the deletion of Δ_i reduces OPT by some number p, while increasing the total number of edges deleted so far by at most $c \cdot p$. We proceed to the next iteration with $G_{i+1} := G'_i - \Delta_i$. Eventually, we arrive at the situation when the current graph G_i is already $\mathcal{F}$-free, in which case we stop. A constant-factor approximate solution can be then obtained by reverting the iterations: Proceeding from the last iteration to the first, we always add the extracted set Δ_i to the constructed solution, and roll-back the protrusion reductions performed by the algorithm of Lemma 16 while lifting the current solution using the solution-lifting algorithm. This concludes the proof of Theorem 2 (the formal description is in the full version of the paper).

5 Linear kernel

In the previous section we observed (Lemma 19) 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 16) 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 unable to do this. Instead, bouquets and thetas will be reduced to constant size in the amortized sense.

The next lemma is the crux of our approach: provided we know a "local" solution Δ that isolates a bouquet into an $\mathcal{F}$-free part, this bouquet can be pruned proportionally to the size of Δ without changing $\mathsf{OPT}(G)$. The proof is by a simple replacement argument, and the same reasoning can also be applied to limit the sizes of thetas.

▶ **Lemma 22** (⋆). *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 $\mathsf{OPT}(G) = \mathsf{OPT}(G')$, where G' is obtained from G by removing vertices of all except $d_{\mathcal{F}} + |\Delta|$ elements of the bouquet.*

We are now ready to show the main part of our reasoning: given some solution F, for example the one returned by the approximation algorithm of Theorem 2, we are able to reduce the graph, provided it is not already bounded linearly in $|F|$.

▶ **Lemma 23** (⋆). *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 $\mathsf{OPT}(G) = \mathsf{OPT}(G')$ and $\|G'\| < \|G\|$.*

The proof uses the following strategy based on the idea of *amortization*. Given a solution F, we use parts of F as local solutions to locally bound bouquets and thetas. More precisely, we first perform a similar structural analysis of G with F removed as in the proof of Lemma 19; see the sketch following its statement. There, we considered a neat tree-cut decomposition $\mathcal{T} = (T, \mathcal{X})$ of $G - F$ of width $b_{\mathcal{F}}$. In this decomposition, we highlighted a subset $M \subseteq V(T)$, the lca-closure of those nodes of T whose bags are incident to F. We concluded that the only parts not yet bounded linearly in terms of $|F|$ were large bouquets and thetas with both attachment points contained in a bag X_t of some node $t \in M$. For such a node t, define $\Delta(t)$ as the set containing: (i) all the edges of F incident to X_t, and (ii) all the adhesions of edges of T incident to t that lead to components of $T - M$ containing other vertices of M. Then $\Delta(t)$ easily satisfies the prerequisites of Lemma 22. Hence, $d_{\mathcal{F}} + |\Delta(t)|$ can be used as a bound for reducing these bouquets and thetas. Summing these bounds through all nodes $t \in M$, we achieve an $\mathcal{O}(|F| + |M|)$ bound, thus $\mathcal{O}(|F|)$ due to $|M| \leq 4|F|$.

With Lemma 23, we can now easily conclude the proof of Theorem 3 (the formal description is in the full version of the paper). First, we get rid of all excessive protrusions in the graph (Lemma 16) and compute an approximate solution F_{apx} (Theorem 2). If $|F_{\mathrm{apx}}| > c_{\mathrm{apx}} \cdot k$, the input is a NO instance. Otherwise, we give F_{apx} to the algorithm of Lemma 23 which, provided the graph is still too large to be a kernel, computes a strictly smaller, but equivalent instance. We then recurse on this smaller instance, eventually returning a linear kernel.

Acknowledgements. The authors 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 Rémy Belmonte, Archontia Giannopoulou, Daniel Lokshtanov, and Dimitrios M. Thilikos. The Structure of W_4-Immersion-Free Graphs. *ArXiv e-prints 1602.02002*, February 2016.

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

3 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *J. ACM*, 63(5):44:1–44:69, November 2016. doi:10.1145/2973749.

4 Heather D. Booth, Rajeev Govindan, Michael A. Langston, and Siddharthan Ramachandramurthi. Fast algorithms for K_4 immersion testing. *J. Algorithms*, 30(2):344–378, 1999.

5 Dimitris Chatzidimitriou, Jean-Florent Raymond, Ignasi Sau, and Dimitrios M. Thilikos. An $O(\log \mathsf{OPT})$-approximation for covering/packing minor models of θ_r. *Algorithmica*, 2017. To appear.

6 Rajesh Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016.

7 Bruno Courcelle. The Monadic Second-Order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

8 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

9 Matt Devos, Zdeněk Dvořák, Jacob Fox, Jessica McDonald, Bojan Mohar, and Diego Scheide. A minimum degree condition forcing complete graph immersion. *Combinatorica*, 34(3):279–298, 2014.

10 Zdeněk Dvořák and Paul Wollan. A structure theorem for strong immersions. *J. Graph Theory*, 83(2):152–163, 2016. doi:10.1002/jgt.21990.

11 Zdeněk Dvořák and Liana Yepremyan. Complete graph immersions and minimum degree. *ArXiv e-prints 1512.00513*, December 2015.

12 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.

13 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-Deletion: Approximation and optimal FPT algorithms. *ArXiv e-prints 1204.4230*, October 2012.

14 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar $\mathcal{F}$-deletion: Approximation, kernelization and optimal FPT algorithms. In *Proceedings of FOCS 2012*, pages 470–479. IEEE Computer Society, 2012.

15 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In *Proceedings of SODA 2010*, pages 503–510. SIAM, 2010.

16 Robert Ganian, Eun Jung Kim, and Stefan Szeider. Algorithmic applications of tree-cut width. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Proceedings of MFCS 2015*, volume 9235 of *Lecture Notes in Computer Science*, pages 348–360. Springer, 2015.

17 Archontia C. Giannopoulou, Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. Uniform kernelization complexity of hitting forbidden minors. *ACM Trans. Algorithms*, 13(3):35:1–35:35, March 2017. doi:10.1145/3029051.

18 Archontia C. Giannopoulou, O-joung Kwon, Jean-Florent Raymond, and Dimitrios M. Thilikos. Packing and covering immersion models of planar subcubic graphs. In *Proceedings of WG 2016*, pages 74–84. Springer, 2016. Preprint: *ArXiv e-prints 1602.04042*.

19 Archontia C. Giannopoulou, Michał Pilipczuk, Dimitrios M. Thilikos, Jean-Florent Raymond, and Marcin Wrochna. Linear kernels for edge deletion problems to immersion-closed graph classes. *ArXiv e-prints 1609.07780*, September 2016. URL: https://arxiv.org/abs/1609.07780.

20 Archontia C. Giannopoulou, Iosif Salem, and Dimitris Zoros. Effective computation of immersion obstructions for unions of graph classes. *J. Comput. Syst. Sci.*, 80(1):207–216, 2014.

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

22 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of STOC 2011*, pages 479–488. ACM, 2011.

23 Ananth V. Iyer, H. Donald Ratliff, and Gopalakrishnan Vijayan. On an edge ranking problem of trees and graphs. *Discrete Appl. Math.*, 30(1):43–52, 1991.

24 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath 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.

25 Eun Jung Kim, Sang-il Oum, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. An FPT 2-approximation for tree-cut decomposition. *Algorithmica*, pages 1–20, 2016. doi:10.1007/s00453-016-0245-5.

26 Tak Wah Lam and Fung Ling Yue. Edge ranking of graphs is hard. *Discrete Appl. Math.*, 85(1):71–86, 1998.

27 Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006. doi:10.1016/j.tcs.2005.10.007.

28 Dániel Marx and Paul Wollan. Immersions in highly edge connected graphs. *SIAM J. Discrete Math.*, 28(1):503–520, 2014.

29 Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, Oxford, 2006.

30 N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

31 Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.

32 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.

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

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

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

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