Dynamic Meta-Kernelization*

Christian Bertram
University of Copenhagen
Copenhagen, Denmark
chbe@di.ku.dk

Deborah Haun
Karlsruhe Institute of Technology,
Karlsruhe, Germany
uvfzn@student.kit.edu

Mads Vestergaard Jensen
University of Copenhagen
Copenhagen, Denmark
mvje@di.ku.dk

Tuukka Korhonen
University of Copenhagen
Copenhagen, Denmark
tuko@di.ku.dk

Abstract

Kernelization studies polynomial-time preprocessing algorithms. Over the last 20 years, the most celebrated positive results of the field have been linear kernels for classical NP-hard graph problems on sparse graph classes. In this paper, we lift these results to the dynamic setting.

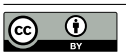
As the canonical example, Alber, Fellows, and Niedermeier [J. ACM 2004] gave a linear kernel for dominating set on planar graphs. We provide the following dynamic version of their kernel: Our data structure is initialized with an n-vertex planar graph G in $O(n \log n)$ amortized time, and, at initialization, outputs a planar graph K with $\mathrm{OPT}(K) = \mathrm{OPT}(G)$ and $|K| = O(\mathrm{OPT}(G))$, where $\mathrm{OPT}(\cdot)$ denotes the size of a minimum dominating set. The graph G can be updated by insertions and deletions of edges and isolated vertices in $O(\log n)$ amortized time per update, under the promise that it remains planar. After each update to G, the data structure outputs $O(1)$ updates to K, maintaining $\mathrm{OPT}(K) = \mathrm{OPT}(G)$, $|K| = O(\mathrm{OPT}(G))$, and planarity of K.

Furthermore, we obtain similar dynamic kernelization algorithms for all problems satisfying certain conditions on (topological-)minor-free graph classes. Besides kernelization, this directly implies new dynamic constant-approximation algorithms and improvements to dynamic FPT algorithms for such problems.

Our main technical contribution is a dynamic data structure for maintaining an approximately optimal *protrusion decomposition* of a dynamic topological-minor-free graph. Protrusion decompositions were introduced by Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, and Thilikos [J. ACM 2016], and have since developed into a part of the core toolbox in kernelization and parameterized algorithms.

*Due to space limits, most technical details are omitted or just sketched. The full version of the paper is available on arXiv [7]. This work was supported by the VILLUM Foundation, Grant Number 54451, Basic Algorithms Research Copenhagen (BARC). T. Korhonen was supported by the European Union under Marie Skłodowska-Curie Actions (MSCA), project no. 101206430.

Funded by the European Union



STOC '26, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-2536-4/2026/06
https://doi.org/10.1145/3798129.3800774

CCS Concepts

• **Theory of computation → Dynamic graph algorithms**; **Parameterized complexity and exact algorithms**.

Keywords

dynamic graph algorithms,parameterized complexity,protrusion decompositions,kernelization

ACM Reference Format:
Christian Bertram, Deborah Haun, Mads Vestergaard Jensen, and Tuukka Korhonen. 2026. Dynamic Meta-Kernelization. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC '26), June 22–26, 2026, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3798129.3800774

1 Introduction

The field of kernelization studies polynomial-time preprocessing algorithms for NP-hard problems. In order to achieve worst-case performance guarantees in this setting, one must consider parameterized problems where the hardness of an instance is captured by a parameter. A *kernelization algorithm*, also called simply a *kernel*, for a parameterized problem takes an input (I, k), runs in time $\mathrm{poly}(|I|, k)$, and outputs an instance (K, k') of the problem so that (1) (K, k') is a yes-instance if and only if (I, k) is a yes-instance, and (2) $|K| + k' \leq f(k)$ for a computable function f [25].

It is a classic observation that a problem has a kernel if and only if it is fixed-parameter tractable [17]. This motivates the notions of a *polynomial kernel* and a *linear kernel*, which restrict the function f in the definition above to be a polynomial (resp. linear) function. Lower bounds for kernels are known under the assumption that coNP $\not\subseteq$ NP/poly, showing that some natural problems, such as Longest Path parameterized by the length, do not (likely) have polynomial kernels despite being fixed-parameter tractable [11, 36]. Similarly, Feedback Vertex Set has a kernel with $O(k^2)$ edges [65], but not with $O(k^{2-\varepsilon})$ edges for any $\varepsilon > 0$, unless coNP $\subseteq$ NP/poly [23]. We refer the reader to [34] for a recent textbook on kernelization.

In this paper, we design *dynamic* linear kernels for graph problems on sparse graph classes. This means designing data structures that support updating the input graph while simultaneously efficiently maintaining a kernel. To be concrete about our contribution before diving deeper into the literature, let us start by stating a representative example of a result that we obtain. It provides a dynamic linear kernel for Dominating Set (parameterized by the solution size) on planar graphs.

Theorem 1.1. *There is a data structure that is initialized with a planar graph G in $O(|G| \log |G|)$[1] amortized time, supports updating G via insertions and deletions of edges and isolated vertices in amortized $O(\log |G|)$ time per update under the promise that G remains planar, and throughout maintains a graph K so that*

(1) K is planar,
(2) $\mathrm{OPT}(K) = \mathrm{OPT}(G)$, where $\mathrm{OPT}(\cdot)$ denotes the size of a minimum dominating set, and
(3) $|K| \leq O(\mathrm{OPT}(G))$.

The data structure outputs K at the initialization, and after each update it outputs at most $O(1)$ updates to K, which consist of insertions and deletions of edges and isolated vertices.

Theorem 1.1 follows from our meta-theorem Theorem 1.3, which provides similar results for any pair of a problem and a graph class that satisfy certain conditions.

The reader may notice that the kernel of Theorem 1.1 does not quite match the definition we provided a couple of paragraphs earlier, but is in fact slightly stronger, working for all values k of the parameter simultaneously. We also highlight the fact that the graph K is updated by (worst-case) $O(1)$ updates per update to G, which is a non-trivial result under any polynomial running time. It enables efficient chaining of Theorem 1.1 with other dynamic data structures. Another observation is that the data structure of Theorem 1.1 also maintains a constant approximation to the minimum dominating set size due to the inequalities $\mathrm{OPT}(G) \leq |K| \leq O(\mathrm{OPT}(G))$.

Let us then discuss the literature of linear kernels on sparse graph classes before stating our results in the full generality.

Linear kernels on sparse graph classes. Among the most influential results in kernelization are linear kernels for NP-hard graph problems parameterized by solution size on sparse graph classes. The first such result was by Alber, Fellows, and Niedermeier [1], who gave a linear kernel for Dominating Set on planar graphs with running time $O(n^3)$. We note that even the existence of any kernel for Dominating Set on planar graphs is non-trivial and perhaps surprising, as on general graphs Dominating Set is W[2]-complete [24] and thus has no kernel under the standard assumption FPT $\neq$ W[2].

The result of Alber et al. led to a flurry of linear kernels on planar graphs, for example, for Feedback Vertex Set [15], Cycle Packing [16], Induced Matching [48, 60], Full-Degree Spanning Tree [44], and Connected Dominating Set [56] (see also [20]). Guo and Niedermeier [43] gave a framework capturing some of the techniques for kernels on planar graphs, obtaining linear kernels also for Connected Vertex Cover, Edge Dominating Set, Triangle Packing, and Efficient Dominating Set. Fomin and Thilikos generalized the planar Dominating Set kernel to graphs of bounded Euler genus [35].

In 2009, all of the aforementioned results were subsumed by a general meta-theorem of Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, and Thilikos [12, 13]. The theorem states that all problems on graphs of bounded Euler genus that (1) are "quasi-coverable", and (2) have "finite integer index" (FII) have a linear kernel. The appendix of [13] mentions 31 such problems.

[1]We denote by $|G| = |V(G)| + |E(G)|$ the total number of vertices and edges of a graph G.

The condition (1) is a technical statement that uses distances on a surface. Its modern equivalent [33] formulation would be that all instances G of the problem admit a treewidth-η-modulator $X \subseteq V(G)$ of size $|X| \leq O(\mathrm{OPT}(G))$ for a constant η, i.e., a set X such that $\mathrm{tw}(G \setminus X) \leq \eta$.[2] The condition (2), i.e. FII, means roughly speaking that the problem has a dynamic programming algorithm on graphs of bounded treewidth, such that the integer values on the dynamic programming table are well-behaved.

The main technique introduced by [13] is *protrusion replacement*. A *c-protrusion* in a graph G is a set $P \subseteq V(G)$ so that $\mathrm{tw}(G[P]) \leq c$ and $|\partial P| \leq c$, where ∂P denotes the vertices in P that have a neighbor outside of P. The parameter c should be seen as a constant that depends on the problem and the graph class. Protrusion replacement means the operation of replacing a protrusion P by a small gadget, while maintaining that the optimum value of the problem stays the same, up to a shifting constant that can be computed while replacing the protrusion. Having FII implies that protrusions can be replaced by constant-size gadgets [13].

The algorithm of [13] works by replacing protrusions by constant-size gadgets until the graph has no more large enough protrusions to make progress. To argue that this results in a kernel, they use *protrusion decompositions*. A (k, c)-protrusion decomposition of a graph G is a pair (T, bag), where T is a tree rooted at a node r and $\mathrm{bag}\colon V(T) \to 2^{V(G)}$ is a function that satisfies

(1) for all $uv \in E(G)$ there is $t \in V(T)$ with $u, v \in \mathrm{bag}(t)$,
(2) for all $v \in V(G)$, the set $\{t \in V(T) \mid v \in \mathrm{bag}(t)\}$ is non-empty and connected in T,
(3) $|\mathrm{bag}(r)| \leq k$ and the degree of r is $\leq k$, and
(4) $|\mathrm{bag}(t)| \leq c$ for all $t \in V(T) \setminus \{r\}$.

In other words, a (k, c)-protrusion decomposition is a tree decomposition (T, bag), where the root-bag has size and degree $\leq k$ and other bags size $\leq c$. The subtrees rooted at the children of the root form c-protrusions, and thus, a graph that has a (k, c)-protrusion decomposition, but whose c-protrusions have size $O(1)$, has size $O(k)$. Bodlaender et al. showed that the condition (1) on graphs of bounded genus implies the existence of an $(O(\mathrm{OPT}(G)), O(1))$-protrusion decomposition, implying that replacing protrusions results in a linear kernel.

The machinery of protrusion replacement and protrusion decompositions was later used for even more general kernelization meta-theorems [33, 49], for other applications in kernelization [22, 28–32, 37, 38, 41, 46, 51, 57], and for applications in parameterized and approximation algorithms outside of kernelization [6, 18, 30, 42, 47, 49, 50].

The meta-theorem of Fomin, Lokshtanov, Saurabh, and Thilikos [33] states that (1) all problems that are "CMSO_2-definable", "linear-separable", and "minor-bidimensional" admit linear kernels on minor-free graphs, and (2) all problems that are "CMSO_2-definable", "linear-separable", and "contraction-bidimensional" admit linear kernels on apex-minor-free graphs. The problems for which (1) applies include for example Cycle Packing and Feedback Vertex Set. The problems for which (2) applies include additionally (r-)Dominating Set, Connected Vertex Cover, and r-Scattered

[2]We denote the treewidth of a graph G by $\mathrm{tw}(G)$. For the definition of treewidth we refer the reader to Section 3.2 of the full version [7], and for a comprehensive introduction, see [34, Chapter 14].

Set. The meta-theorem of Kim, Langer, Paul, Reidl, Rossmanith, Sau, and Sikdar [49] states that all problems that are "linearly treewidth-bounding" and have FII have linear kernels on topological-minor-free graphs. These include for example Feedback Vertex Set, Chordal Vertex Deletion, and Edge Dominating Set.

The algorithms of both [33] and [49] follow the same idea as the algorithm of [13]: They use FII and employ protrusion replacement, and then argue via treewidth modulators that each G has an $(O(\mathrm{OPT}(G)), O(1))$-protrusion decomposition, certifying that protrusion replacement results in a linear kernel. These algorithms, as well as other purely protrusion-replacement based kernels, can be implemented in linear time by the linear-time protrusion-replacement routine of Fomin, Lokshtanov, Misra, Ramanujan, and Saurabh [29] (see also [30, 34]).

Our contribution. The main technical contribution of this paper is to provide a dynamic version of the protrusion replacement technique, resulting in dynamic versions of protrusion-based kernels. While there are some previous works on dynamic kernelization [2, 3, 5, 21, 45], there are no previous dynamic algorithms based on protrusions nor dynamic versions of the above discussed linear kernels for sparse graph classes.

We provide a dynamic algorithm that maintains a protrusion decomposition of a topological-minor-free graph, so that the first parameter of the decomposition is linear in the minimum size of a treewidth modulator. We use $O_{\bar{p}}(\cdot)$ to denote the O-notation that ignores factors that depend on a tuple of parameters $\bar{p}$ and are computable given $\bar{p}$. We denote by $\text{tw-mod}_\eta(G)$ the size of a smallest set $X \subseteq V(G)$ so that $\mathrm{tw}(G \setminus X) \leq \eta$. We state here a version of our main theorem omitting certain technical details. The full statement appears as Theorem 7.5 in the full version [7].

Theorem 1.2. *There is a data structure that is initialized with a graph H, integer η, and an H-topological-minor-free graph G. It supports updating G via insertions and deletions of edges and isolated vertices, under the promise that G remains H-topological-minor-free. It maintains an $(O_{H,\eta}(\text{tw-mod}_\eta(G)), O_{H,\eta}(1))$-protrusion decomposition $\mathcal{T}$ of G. The amortized running time of the initialization is $O_{H,\eta}(|G| \log |G|)$, and the amortized update time is $O_{H,\eta}(\log |G|)$.*

Furthermore, the data structure provides infrastructure for maintaining dynamic programming procedures on the subtrees of $\mathcal{T}$ rooted at the children of the root. Each update to G causes updates to only $O_{H,\eta}(1)$ such subtrees, and changes the root-bag only by at most $O_{H,\eta}(1)$.

The second paragraph of the statement of Theorem 1.2 is an informal explanation of the technical details that are needed for the application to dynamic kernelization.

The parameters of the protrusion decomposition of Theorem 1.2 are optimal by $O_{H,\eta}(1)$-factors, since having a (k, η)-protrusion decomposition implies that $\text{tw-mod}_\eta(G) \leq k$. Moreover, topological-minor-free graph classes are the most general subgraph-closed graph classes where a linear relation between treewidth modulators and protrusion decompositions holds (see Proposition 9.1 in the full version [7]), so the restriction to topological-minor-free graphs is justified.

To the best of our knowledge, the initialization algorithm[3] of Theorem 1.2 is the first algorithm to explicitly compute a protrusion decomposition with approximately optimal parameters in near-linear time, without a given treewidth-modulator. Previously, an $O_{H,\eta}(|G|^2)$ time algorithm was given by Kim, Serna, and Thilikos [51] (in the same setting of H-topological-minor-free graphs and $O_{H,\eta}(1)$-approximation). However, most of the previous kernels based on protrusion replacement do not use the decomposition explicitly, but instead employ iterative protrusion replacement, which can be implemented in linear time [29].

The data structure of Theorem 1.2 can be seen as a generalization of the recent dynamic treewidth data structure of Korhonen [52]. Applying it to graphs of treewidth $\leq \eta$ (which exclude $K_{\eta+2}$ as a topological-minor and have $\text{tw-mod}_\eta(G) = 0$) recovers the result of [52], but with a significantly higher dependency on η in both the running time and the width. This implies that the factor $\log |G|$ in the amortized update time is optimal, as it is required (unconditionally) even for maintaining dynamic forests [62].

Dynamic meta-kernelization. We then present our kernelization meta-theorem, which provides dynamic kernelization algorithms similar to Theorem 1.1 for a large class of problems. For that, we need a couple of definitions.

By CMSO_2 we mean the Counting Monadic Second-Order logic on graphs (see e.g. [34, Section 14.5]). We say that a graph class $\mathcal{G}$ is CMSO_2-definable if there is a CMSO_2-sentence Φ so that $G \models \Phi$ if and only if $G \in \mathcal{G}$. Most of the natural graph classes are CMSO_2-definable. A graph class $\mathcal{G}$ excludes a topological minor if there exists a graph H so that no graph in $\mathcal{G}$ contains H as a topological minor. Classes excluding a topological minor include classes excluding a minor, such as the planar graphs, but also the graphs of bounded degree.

We consider parameterized graph problems Π that are either minimization or maximization problems, and denote by $\mathrm{OPT}_\Pi(G)$ the smallest (resp. largest) value k so that (G, k) is a yes-instance.[4] A problem Π is *linearly treewidth-bounding* on a graph class $\mathcal{G}$ if there is a constant η so that for all $G \in \mathcal{G}$ we have $\text{tw-mod}_\eta(G) \leq O(\mathrm{OPT}_\Pi(G))$. For example, for Dominating Set on planar graphs we can take $\eta = 2$ [34]. The statement of Theorem 1.3 also uses the definition of FII, which we have already introduced informally and which is defined in Section 3.3 of the full version [7].

Theorem 1.3. *Let $\mathcal{G}$ be a CMSO_2-definable graph class that excludes a topological minor, and Π a parameterized graph problem that is linearly treewidth-bounding on $\mathcal{G}$ and has FII.*

There is a data structure that is initialized with a graph $G \in \mathcal{G}$ in $O(|G| \log |G|)$ time, supports updating G via insertions and deletions of edges and isolated vertices in amortized $O(\log |G|)$ time per update under the promise that G remains in $\mathcal{G}$, and throughout maintains a graph K and a non-negative integer Δ so that

(1) $K \in \mathcal{G}$,
(2) $\mathrm{OPT}_\Pi(K) + \Delta = \mathrm{OPT}_\Pi(G)$, and
(3) $|K| \leq O(\mathrm{OPT}_\Pi(G))$.

[3] The initialization algorithm simply inserts vertices and edges one-by-one into the data structure.

[4] For the purpose of stating Theorem 1.3, let us assume that such k exists. A more precise form of Theorem 1.3 is given as the pair of Lemmas 8.16 and 8.17 in the full version [7].

The data structure outputs (K, Δ) at the initialization, and after each update it outputs at most $O(1)$ updates to (K, Δ), which consist of insertions and deletions of edges and isolated vertices to/from K, and (unbounded) changes of Δ.

Now, for example Theorem 1.1 follows by taking $\mathcal{G}$ to be the class of planar graphs, Π the Dominating Set problem, and using the known fact that Dominating Set is linearly treewidth-bounding on planar graphs [33]. More generally, we could replace $\mathcal{G}$ by any apex-minor-free CMSO_2-definable graph class [33], for example, the graphs of Euler genus at most 10.

For most of the natural problems such as Dominating Set, we can get rid of the shifting-constant Δ by encoding it in K via gadgets. In particular, from the fact that K changes by at most $O(1)$ updates per update to G, it follows that the value of Δ also changes by at most $O(1)$, so we can within the same bounds maintain a set of Δ additional isolated vertices in K.

The algorithm of Theorem 1.3 applies to all problems to which the meta-theorems of Fomin et al. [33, Theorem 1.1] and Kim et al. [49, Theorem 1] apply. This also encompasses the results of Bodlaender et al. about linear kernels [13, Theorem 1.3]. We give a list of concrete problems to which Theorem 1.3 applies in Section 9 of the full version [7].

The amortized update time $O(\log |G|)$ in Theorem 1.3 is optimal for some problems captured by it. In particular, the result of Pătraşcu and Demaine [62] implies that the problem of maintaining whether a planar graph is a forest or contains a cycle requires $O(\log |G|)$ (amortized and randomized) update time, unconditionally. This implies that $O(\log |G|)$ update time is required for any problem Π on planar graphs whose optimum value is a constant c if and only if G is a forest, for example, Cycle Packing or Feedback Vertex Set.

The statement of Theorem 1.3 is non-constructive, in particular, it is not clear how the problem Π would even be described. The proof is also inherently non-constructive because of the non-constructive nature of the definition of FII and the related protrusion-replacement machinery. However, it can be made constructive for large classes of concrete problems by using the techniques introduced by Garnero, Paul, Sau, and Thilikos [39, 40].

Applications outside of kernelization. Theorem 1.3 has some direct applications outside of kernelization. As the first example, we observe that it improves the update times of many dynamic parameterized algorithms from $2^{O(\sqrt{k})} \log n$ to $2^{O(\sqrt{k})} + O(\log n)$. In particular, Korhonen [52] observed that his dynamic treewidth data structure gives a dynamic algorithm for maintaining whether an n-vertex planar graph contains a dominating set of size $\leq k$ in $2^{O(\sqrt{k})} \log n$ amortized update time. Plugging in Theorem 1.3 (in fact, Theorem 1.1), directly improves this to an amortized update time of $2^{O(\sqrt{k})} + O(\log n)$. Similar effect happens to a large class of parameterized problems on (topological-)minor-free graph classes.

Another direct application is to dynamic approximation algorithms. We observe that if a problem Π satisfies that $\mathrm{OPT}_\Pi(G) \leq O(|G|)$ for all G, then the data structure of Theorem 1.3 directly maintains a constant-factor approximation of $\mathrm{OPT}_\Pi(G)$, because of the inequalities $\mathrm{OPT}_\Pi(G) \leq O(|K|) + \Delta \leq O(\mathrm{OPT}_\Pi(G))$. To the best of our knowledge, this is the first dynamic constant-factor approximation algorithm for many problems captured by Theorem 1.3, for example, for Feedback Vertex Set on planar graphs.

Previously, Korhonen, Nadara, Pilipczuk, and Sokołowski [55] gave dynamic $(1+\varepsilon)$-approximation algorithms for Weighted Independent Set on apex-minor-free graphs and Weighted Dominating Set on bounded-degree minor-free graphs. They achieve $f(\varepsilon) \cdot n^{o(1)}$ amortized update time by implementing a dynamic version of the Baker's scheme [4]. Dynamic approximation algorithms are well-studied for Matching and Vertex Cover, see e.g. [8, 9, 64].

Our techniques. Our main technical contribution is Theorem 1.2, which implies Theorem 1.3 via a dynamic implementation of the known protrusion replacement machinery. The basic blueprint behind the data structure of Theorem 1.2 is inspired by the dynamic treewidth data structure of Korhonen [52]. In particular, the subtrees of bounded treewidth are maintained by the same routine as in [52], and the whole protrusion decomposition satisfies the same key invariant of "downwards well-linkedness" as the tree decomposition maintained by [52]. The difference is that now the root-bag of the decomposition can have large size, and in particular, its size should be maintained to be approximately the same as the minimum treewidth-η-modulator.

It is easy to incorporate edge insertions/deletions to this structure so that they increase the size of the root-bag by a constant in each operation. Therefore, the main challenges are to:

(1) show that if the root-bag grows too large compared to the optimum treewidth-η-modulator, then it can be reduced by chopping off a small part to the bounded-treewidth subtrees, and
(2) implement this chopping within $O(\log n)$ amortized update time and $O(1)$ changes to the root per update.

Both of the parts (1) and (2) are non-trivial. The part (1) yields an essentially new type of an algorithm for constructing a protrusion decomposition, which constructs a protrusion decomposition by iteratively making the root-bag smaller, unlike the previous algorithms, which proceed by starting with a treewidth-η-modulator and then making the root-bag a superset of it. The proof of (1) uses graph-theoretical techniques, requiring topological-minor-freeness and heavily relying on the "downwards well-linkedness" of the decomposition. The part (2) requires a careful dynamic implementation of a local search procedure for finding large sets of vertices with small neighborhoods.

Related work on dynamic kernelization. So far we omitted the discussion on previous dynamic kernelization algorithms, so let us review them here. Let us focus only on dynamic polynomial kernels, and omit the larger body of work on dynamic FPT algorithms (see e.g. [2, 10, 19, 26, 27, 52, 54, 58, 59, 61]).

Iwata and Oka [45] gave dynamic kernels for Vertex Cover and Cluster Vertex Deletion. The kernel for Vertex Cover has $O(k^2)$ update time and size $O(k^2)$, while the kernel for Cluster Vertex Deletion has $O(k^8 \log n)$ update time and size $O(k^5)$.

Alman, Mnich, and Vassilevska Williams [2] improved the update time of the dynamic vertex cover kernelization to $O(k)$ worst-case and $O(1)$ amortized. They also gave dynamic polynomial kernels

for d-Hitting Set, Edge Dominating Set, and Point Line Cover. The kernel for Edge Dominating Set has update time $O(1)$ and size $O(k^2)$, while the kernels for d-Hitting Set and Point Line Cover have update times and sizes $\text{poly}(k)$.

An improved dynamic kernel for d-Hitting Set, along with a dynamic kernel for Set Packing, was given by Bannach, Heinrich, Reischuk, and Tantau [5]. An, Cho, Jang, Jung, Lee, Oh, Shin, Shin, and Song [3] designed dynamic kernels on unit disk graphs for Vertex Cover, Triangle Hitting Set, Feedback Vertex Set, and Cycle Packing.

Organization of the paper. Due to space limitations, we provide only an overview of the proofs. The full version is available on arXiv [7].

2 Overview

We first sketch a proof of Theorem 1.2, explaining how to maintain an approximately optimal protrusion decomposition of a topological-minor-free graph. Afterwards, we describe how this data structure can be used for dynamic kernelization, sketching the proof of Theorem 1.3.

2.1 Dynamic Protrusion Decomposition

We consider a dynamic H-topological-minor-free graph G and a parameter η. Our goal is to maintain, under insertions and deletions of edges and isolated vertices, a $(O_{H,\eta}(\text{tw-mod}_\eta(G)), O_{H,\eta}(1))$-protrusion decomposition of G. This is a rooted tree decomposition (T, bag), where the root-bag $r \in V(T)$ has size and degree $O_{H,\eta}(\text{tw-mod}_\eta(G))$, and other bags have size $O_{H,\eta}(1)$.

We start by describing in Section 2.1.1 the definitions and ideas that are based on those of [52] (many of which in turn originate from [53], and further from [63]). The more novel parts of our algorithm are described in Section 2.1.2.

2.1.1 The Framework and Basic Operations.

Downwards well-linked superbranch decompositions. The key idea of the dynamic treewidth algorithm of [52] is to maintain a structure called *downwards well-linked superbranch decomposition*. We do the same in this paper, although naturally with different constraints to reflect that the superbranch decomposition should correspond to a protrusion decomposition instead of a tree decomposition of bounded width.

A *superbranch decomposition* of a graph G is a pair $\mathcal{T} = (T, \mathcal{L})$, where T is a rooted tree, in which every non-leaf node has at least two children, and $\mathcal{L}$ is a bijection that maps every leaf of T to an edge of G. This is similar to the classic definition of a branch decomposition [63], but allowing nodes of degree higher than three. For a node $t \in V(T)$, we denote by $\mathcal{L}[t] \subseteq E(G)$ the set of edges of G that are associated by $\mathcal{L}$ with leaves in the subtree of T below t.

The *boundary* of an edge set $A \subseteq E(G)$, denoted by $\text{bd}(A) \subseteq V(G)$, is the set of vertices that are incident to edges in both A and $E(G)\setminus A$. We denote the size of the boundary by $\lambda(A) = |\text{bd}(A)|$. The function $\lambda\colon 2^{E(G)} \to \mathbb{Z}_{\geq 0}$ is *symmetric and submodular*, meaning that (1) $\lambda(A) = \lambda(E(G)\setminus A)$ and (2) $\lambda(A\cup B)+\lambda(A\cap B) \leq \lambda(A)+\lambda(B)$ for all $A, B \subseteq E(G)$ [63].

A set $A \subseteq E(G)$ of edges is *well-linked* if for every bipartition (A_1, A_2) of A it holds that $\lambda(A_1) \geq \lambda(A)$ or $\lambda(A_2) \geq \lambda(A)$ (this is called *robust* in [63]). The *well-linked number* $\text{wl}(A)$ of a set $A \subseteq E(G)$ is the maximum of $\lambda(A')$ over well-linked subsets $A' \subseteq A$. A superbranch decomposition is *downwards well-linked* if for every node $t \in V(T)$, the set $\mathcal{L}[t]$ is well-linked.

The intuitive reason why well-linkedness is a powerful notion in our context is the following three properties [52, 53]:

(1) If $A \subseteq E(G)$ is a well-linked set and $B \subseteq E(G)$, then either $\lambda(B\cup A) \leq \lambda(B)$ or $\lambda(B\setminus A) \leq \lambda(B)$. In particular, well-linked sets are perfectly "uncrossable".
(2) Every set $A \subseteq E(G)$ can be partitioned into at most $2^{\lambda(A)}$ well-linked subsets.
(3) For all $A \subseteq E(G)$, it holds that $\text{wl}(A) = \Theta(\text{tw}(G[A]))$.

There is also a fourth powerful property, called the "transitivity" of well-linkedness in [52, 53], but which requires more complex definitions to state, so we omit it for now.

From superbranch to protrusion decompositions. There is a natural way to relate superbranch decompositions to tree decompositions, for which we use the following definition of an *adhesion*. For an edge $tp \in E(T)$ of a superbranch decomposition between a node t and its parent p, the adhesion at tp is the set $\text{adh}(tp) = \text{bd}(\mathcal{L}[t])$. Now, if $(T, \mathcal{L})$ is a superbranch decomposition of a graph G, we construct a function $\text{bag}\colon V(T) \to 2^{V(G)}$ so that (T, bag) is a tree decomposition of G as follows. For a leaf-node ℓ with $\mathcal{L}(\ell) = uv$, we set $\text{bag}(\ell) = \{u, v\}$, and for a non-leaf-node t, we set $\text{bag}(t) = \bigcup_{s\in N(t)} \text{adh}(st)$, where $N(t)$ denotes the neighbors of t. (This does not quite work if G contains isolated vertices, but let us ignore that for now.) We observe that (T, bag) is an $(O_{H,\eta}(\text{tw-mod}_\eta(G)), O_{H,\eta}(1))$-protrusion decomposition of G if

a. every adhesion has size $O_{H,\eta}(1)$,
b. the root-node has degree $O_{H,\eta}(\text{tw-mod}_\eta(G))$, and
c. every non-root-node has degree $O_{H,\eta}(1)$.

This follows from the fact that the size of $\text{bag}(t)$ is bounded by the product of the degree of t and the maximum adhesion size. Instead of maintaining the conditions of Items a to c directly, our goal is to maintain a superbranch decomposition $\mathcal{T} = (T, \mathcal{L})$ satisfying that

(1) $\mathcal{T}$ is downwards well-linked,
(2) for all non-root $t \in V(T)$, $\text{wl}(\mathcal{L}[t]) \leq O_{H,\eta}(1)$,
(3) the root-node has degree $O_{H,\eta}(\text{tw-mod}_\eta(G))$,
(4) every non-root-node has degree $O_{H,\eta}(1)$, and
(5) T has depth $O_{H,\eta}(\log |G|)$.

By the definition of well-linked number and downwards well-linkedness, Items 1 and 2 imply the condition of Item a Thus, any protrusion decomposition satisfying Items 1 to 4 also satisfies Items a to c, and thus corresponds to a $(O_{H,\eta}(\text{tw-mod}_\eta(G)), O_{H,\eta}(1))$-protrusion decomposition. The logarithmic-depth requirement of Item 5 is for efficient dynamic maintenance of the superbranch decomposition.

Basic maintenance of the superbranch decomposition. The idea of our data structure is that we first implement the procedures for inserting and deleting edges in an "easy" way that increases the degree of the root by $O_{H,\eta}(1)$ per operation but maintains the other invariants. Then, the hard part of our data structure is a procedure that decreases the degree of the root whenever it is too

large compared to tw-mod$_\eta(G)$. This procedure will be run after every update to keep the degree of the root controlled.

We postpone the hard part to Section 2.1.2, and start here with the "easy" procedure for inserting and deleting edges. At this point, we need to reveal the technical detail that the superbranch decomposition is not actually a superbranch decomposition of G, but of the hypergraph $\mathcal{H}(G)$ that has vertex set $V(\mathcal{H}(G)) = V(G)$, and in addition to the normal edges $\{u, v\}$ for all $uv \in E(G)$, contains singleton edges $\{v\}$ for all $v \in V(G)$. Our definitions extend naturally to hypergraphs, and later we will also use hypergraphs with edges of size more than 2.

We use the tree-rotation techniques from [52] for maintaining the subtrees below the root. A key subroutine that we implement with those techniques is a procedure that "rotates up" a specified set of leaves of the decomposition. In particular, given a set A of hyperedges with $|A| = O(1)$, the procedure uses tree rotations to transform $(T, \mathcal{L})$ so that

(1) the leaves corresponding to A become children of the root,
(2) the degree of the root increases by $O_{H,\eta}(1)$, and
(3) all other invariants are maintained.

This subroutine runs in $O_{H,\eta}(\log |G|)$ amortized time.

Now, to insert an edge between vertices u and v, we use the subroutine with $A = \{\{u\}, \{v\}\}$, and observe that after $\{u\}$ and $\{v\}$ are children of the root, inserting the hyperedge $\{u, v\}$ while maintaining the invariants (but increasing the root-degree by 1) is trivial, by simply adding a leaf corresponding to it as another child of the root. Similarly, to delete an edge uv, we use the procedure with $A = \{\{u, v\}, \{u\}, \{v\}\}$, after which the deletion becomes similarly straightforward. In both cases, the degree of the root increases by $O_{H,\eta}(1)$ because of the rotating-up subroutine.

The reason why it is essential that also $\{u\}$ and $\{v\}$ are children of the root while inserting/deleting $\{u, v\}$ is that this ensures that the boundaries $\mathrm{bd}(\mathcal{L}[t])$ stay unchanged in the decomposition. Otherwise, the boundaries $\mathrm{bd}(\mathcal{L}[t])$ could abruptly change globally throughout the decomposition, potentially ruining downwards well-linkedness.

2.1.2 Controlling the root degree. Let us then turn to the hard part of our data structure, namely, the procedure for controlling the degree of the root. We will decrease the degree of the root by finding a set of at least 2 but at most $O_{H,\eta}(1)$ subtrees rooted at children of the root, and combining them into one subtree rooted at a new child of the root. This combination procedure will be done in a straightforward manner, in particular, by simply adding a new child and moving the subtrees to be rooted under that child instead of the root.

Denote the children of the root corresponding to such subtrees by $C = \{c_1, c_2, \ldots, c_h\}$, and denote by $\mathcal{L}[C] = \bigcup_{c_i \in C} \mathcal{L}[c_i]$ the set of hyperedges corresponding to leaves below C. We need that

(1) $|C| \geq 2$ so that the degree of the root actually decreases,
(2) $|C| \leq O_{H,\eta}(1)$ so that the degree of the new node is bounded,
(3) $\mathcal{L}[C]$ is well-linked so that the decomposition stays downwards well-linked, and
(4) $\mathsf{wl}(\mathcal{L}[C]) \leq O_{H,\eta}(1)$ so that the well-linked number of the subtrees stays bounded.

Furthermore, the bounds hidden by $O_{H,\eta}(\cdot)$ above should not depend on the current parameters of the decomposition, but only on the original parameters H and η, in order to not make the parameters of the decomposition gradually worse throughout updates. Decreasing the degree by combining such a set of children C maintains all other invariants of the decomposition except may increase the depth by one because of the new subtree. However, the new subtree can be balanced in amortized $O_{H,\eta}(\log |G|)$ time with the techniques from [52].

It follows that now, the two main challenges are to show that

(1) whenever the degree of the root is too large compared to tw-mod$_\eta(G)$, such a set of children C exists, and
(2) in that case, we can also find C efficiently.

For both of these two challenges, the key definition we need is that of the *torso hypergraph*. Let t be a node of a superbranch decomposition. The torso of t, denoted by $\mathrm{torso}(t)$, is the hypergraph with the vertex set $V(\mathrm{torso}(t)) = \bigcup_{s \in N(t)} \mathrm{adh}(st)$, and having the hyperedge $\mathrm{adh}(st)$ for every $s \in N(t)$. Note that the same hyperedge can occur multiple times if $\mathrm{adh}(s_1 t) = \mathrm{adh}(s_2 t)$ for $s_1 \neq s_2$, and we indeed treat $\mathrm{torso}(t)$ as a "multi"-hypergraph, thinking about the hyperedge set of $\mathrm{torso}(t)$ as consisting of labels e_s for all $s \in N(t)$, together with a mapping that maps each label e_s to the set $\mathrm{adh}(st)$.

Let r be the root node and $A \subseteq E(\mathrm{torso}(r))$ a subset of hyperedges of its torso, which naturally corresponds to a set of children C_A of the root. We denote by $A \triangleright \mathcal{T} = \mathcal{L}[C_A] \subseteq E(\mathcal{H}(G))$ the set of hyperedges corresponding to the leaves under C_A. The "transitivity of well-linkedness" from [52, 53] tells that $A \triangleright \mathcal{T}$ is well-linked in $\mathcal{H}(G)$ if and only if A is well-linked in $\mathrm{torso}(r)$. Now, the task of finding the required set of children C translates into finding a set $A \subseteq E(\mathrm{torso}(r))$ so that

(1) $2 \leq |A| \leq O_{H,\eta}(1)$,
(2) A is well-linked in $\mathrm{torso}(r)$, and
(3) $\mathsf{wl}(A \triangleright \mathcal{T}) \leq O_{H,\eta}(1)$.

We can in fact simplify these three conditions even further. The combination of Items 2 and 3 implies that $\lambda(A)$ should be bounded by $O_{H,\eta}(1)$. By using the fact that any set $A \subseteq E(\mathrm{torso}(r))$ can be partitioned into at most $2^{\lambda(A)}$ well-linked sets, we can relax the condition that A is well-linked into a condition that A is large enough compared to $2^{\lambda(A)}$ – then a postprocessing routine can find a well-linked subset of A. We end up with the conditions

(1) $2^{\lambda(A)} < |A| \leq O_{H,\eta}(1)$ and
(2) $\mathsf{wl}(A \triangleright \mathcal{T}) \leq O_{H,\eta}(1)$.

Existence of a mergeable subset. Let us then get to the proof that such a set $A \subseteq E(\mathrm{torso}(r))$ exists if the root degree is too large. Note that the root degree equals $|E(\mathrm{torso}(r))|$.

The first ingredient is a proof showing that $|E(\mathrm{torso}(r))| \leq O_{H,\eta}(|V(\mathrm{torso}(r))|)$. This uses the H-topological-minor-freeness of G and the downwards well-linkedness, in particular, if $\mathrm{torso}(r)$ would be too dense, we could realize this density as a topological minor via downwards well-linkedness. Another case is that $\mathrm{torso}(r)$ contains a lot of hyperedges with exactly the same vertex set, but in that case the desired conclusion is easy to achieve by just selecting a subset of them. Therefore, we can assume that $V(\mathrm{torso}(r))$ is large compared to tw-mod$_\eta(G)$.

Now, we consider an imaginary "optimal" protrusion decomposition of G. By [49], G indeed has an $(O_{H,\eta}(\text{tw-mod}_\eta(G)), O_{H,\eta}(1))$-protrusion decomposition $\mathcal{T}^* = (T^*, \text{bag}^*)$, rooted at a node r^*. If $V(\text{torso}(r))$ is of size larger than $\Omega_{H,\eta}(|\text{bag}^*(r^*)| + \Delta(r^*)) = \Omega_{H,\eta}(\text{tw-mod}_\eta(G))$[5], then there exists a child c^* of r^*, so that the subtree of $\mathcal{T}^*$ rooted at c^* contains at least δ vertices from $V(\text{torso}(r))$, and has treewidth ω, where δ and ω are parameters in $O_{H,\eta}(1)$ with $\omega << \delta$. By choosing an appropriate subtree under c^*, we indeed obtain a set $B \subseteq E(\mathcal{H}(G))$ so that

(1) $\delta \leq |V(B) \cap V(\text{torso}(r))| \leq 3 \cdot \delta$,[6]
(2) $\lambda(B) \leq \omega$, and
(3) $\text{wl}(B) \leq \omega$,

where δ is set to be roughly $2^{\Theta(\omega)}$.

The set B would be perfect for us if it would be "uncrossed" with the sets $\mathcal{L}[c]$ for the children c of the root r of our superbranch decomposition, that is, if either $\mathcal{L}[c] \subseteq B$ or $\mathcal{L}[c] \cap B = \emptyset$ would hold for all c. This would allow us to construct the desired set $A \subseteq E(\text{torso}(r))$ by taking $e_c \in A$ whenever $\mathcal{L}[c] \subseteq B$.

Now the natural idea is to use the downwards well-linkedness to uncross the set B with the sets $\mathcal{L}[c]$. It implies that either $\lambda(B \setminus \mathcal{L}[c]) \leq \lambda(B)$ or $\lambda(B \cup \mathcal{L}[c]) \leq \lambda(B)$, so one can always either include or exclude $\mathcal{L}[c]$ without increasing $\lambda(B)$. Furthermore, we show that doing this for all children c does not drastically affect the quantity $|V(B) \cap V(\text{torso}(r))|$, as long as it is large enough compared to $\lambda(B)$.

However, this uncrossing idea has one issue that makes the proof much more complicated: The quantity $\text{wl}(B)$ can increase if we set $B := B \cup \mathcal{L}[c]$. On the surface this seems to not be a big issue since $\text{wl}(\mathcal{L}[c])$ is anyway bounded by $O_{H,\eta}(1)$, but here we run into the issue that the "new parameters" of the decomposition should not depend on the "old parameters" of the decomposition, or otherwise they would stack up throughout many operations. In particular, this uncrossing would work fine if we could assume that $\text{wl}(\mathcal{L}[c])$ is small enough compared to the other parameters of the decomposition.

We fix this issue by ensuring that this type of uncrossing can indeed only happen if $\text{wl}(\mathcal{L}[c])$ is really small. In particular, we observe that $\text{wl}(\mathcal{L}[c])$ can be larger than $O(\eta)$ only for at most $\text{tw-mod}_\eta(G)$ many subtrees of our decomposition, as the modulator must hit all such subtrees. By tuning the existence proof of protrusion decompositions from [49], we can ensure that the set $\text{bd}(\mathcal{L}[c])$ is contained in $\text{bag}^*(r^*)$ for all such subtrees, and therefore by choosing B according to this tuned decomposition (T^*, bag^*), $V(B)$ can intersect such $\text{bd}(\mathcal{L}[c])$ only in $\text{bd}(B)$. With this, we can ensure that the uncrossing of type $B := B \cup \mathcal{L}[c]$ never happens for such children c.

Thus, skipping many tedious technical details here, the uncrossing idea works out in the end to prove the existence of the desired set $A \subseteq E(\text{torso}(r))$.

Finding a mergeable subset. The next challenge is to find a set $A \subseteq E(\text{torso}(r))$ with (1) $2^{\lambda(A)} < |A| \leq O_{H,\eta}(1)$ and (2) $\text{wl}(A \triangleright \mathcal{T}) \leq O_{H,\eta}(1)$ whenever such a set exists, in $O_{H,\eta}(\log |G|)$ time. We start by observing that the proof above gives us some slack for approximation, in particular, it is enough to be able to either return such a set, or to conclude that there is no such set with (1) replaced by (1') $2^{2\cdot\lambda(A)} < |A| \leq O_{H,\eta}(1)$ and (2) replaced by a smaller bound on the well-linked number. In particular, instead of bound the well-linked number of $A \triangleright \mathcal{T}$, we bound $\lambda(A)$ and the *internal treewidth* of $A \triangleright \mathcal{T}$, that is $\text{tw}(G[V(A \triangleright \mathcal{T}) \setminus \text{bd}(A)])$. This is approximately equivalent to bounding the well-linked number.

Let us start by discussing how to even check whether a given set A satisfies these properties. In our data structure, we maintain $\text{torso}(r)$ explicitly, so (1) is easy enough to check in $O_{H,\eta}(1)$ time. However, checking (2) is not easy, as the internal treewidth involves information not captured by $\text{torso}(r)$. At this point, we note that thanks to the techniques from [52], we are able to maintain dynamic programming procedures on the subtrees of $\mathcal{T}$ below the root. In particular, since such subtrees correspond to tree decompositions of bounded width, we can use powerful bounded-treewidth machinery to capture information about them. This is used for the application of our data structure for kernelization, but we also use it here to figure out the internal treewidth of $A \triangleright \mathcal{T}$. In particular, we maintain a modified version of the Bodlaender-Kloks dynamic programming procedure for computing treewidth [14] on the subtrees, and when given A, combine the information from the subtrees corresponding to A to compute the internal treewidth of $A \triangleright \mathcal{T}$.

Now we know how to efficiently check if a given set A is suitable, but still, there are too many candidates to be brute-forced. Let us sketch an $O_{H,\eta}(|E(\text{torso}(r))|)$ time algorithm. A set $B \subseteq E(\text{torso}(r))$ is *internally connected* if all vertices in $V(B)$ can reach each other by paths through hyperedges in B but not containing vertices from $\text{bd}(B)$ as internal vertices. Internally connected sets are useful because for parameters k and s and a hyperedge e, there are at most s^k internally connected sets B with $|B| \leq s$, $\lambda(B) \leq k$, and $e \in B$, and they can be listed via a local-search procedure in $s^{O(k)}$ time.

Now, a suitable set A can be uniquely partitioned into internally connected sets $A_1, \ldots, A_h$, so that $\text{bd}(A_i) \subseteq \text{bd}(A)$. By possibly shrinking $|A|$ by a factor of $2^{\lambda(A)}$, which is fine by the slack for approximation, we can assume that $\text{bd}(A_i) = \text{bd}(A)$ for all A_i. We also observe that the internal treewidth of $A \triangleright \mathcal{T}$ is equal to the maximum internal treewidth of $A_i \triangleright \mathcal{T}$. Now, this type of A is easy enough to find in $O_{H,\eta}(|E(\text{torso}(r))|)$ time by first enumerating all internally connected sets with specified size and boundary size, checking their internal treewidth, and then grouping them by their boundary. Furthermore, by using the fact that each hyperedge participates in only $O_{H,\eta}(1)$ such internally connected sets, we can design a dynamic implementation of this algorithm, that works in $O_{H,\eta}(\log |E(\text{torso}(r))|)$ time per update to $\text{torso}(r)$.

2.2 Dynamic Kernelization

The application of our data structure to kernelization follows the high-level idea of protrusion replacement, which was pioneered by Bodlaender et al. [13], and afterwards used in dozens of kernelization algorithms. However, our implementation of protrusion replacement is non-standard in that we explicitly compute a protrusion decomposition and replace each of its protrusions in one shot. The typical protrusion replacement implementations work by repeatedly replacing constant-size protrusions, without actually

[5] Where $\Delta(r^*)$ denotes the degree of r^*.
[6] Where $V(B) \subseteq V(G)$ denotes all vertices incident to B.

computing a protrusion decomposition. Nevertheless, this approach of explicitly computing a protrusion decomposition has been used before by Kim, Serna, and Thilikos [51], who needed it in the context of kernelization for counting problems.

Let $\mathcal{G}$ be a CMSO_2-definable graph class that excludes a topological minor H, and Π a problem that has FII and is linearly treewidth-bounding on $\mathcal{G}$. For example, we can let $\mathcal{G}$ be the planar graphs and Π the Dominating Set problem. Because Π is linearly treewidth-bounding in $\mathcal{G}$, there exists η so that $\text{tw-mod}_\eta(G) \leq O(\text{OPT}_\Pi(G))$ for all $G \in \mathcal{G}$. Now, it suffices to give a kernelization algorithm parameterized by the parameter $\text{tw-mod}_\eta(G)$ instead of $\text{OPT}_\Pi(G)$. For this, we apply the data structure of Theorem 1.2, initialized with the parameters H and η.

Let $(T, \mathcal{L})$ be the superbranch decomposition maintained by the data structure, corresponding to a $(O_{H,\eta}(\text{tw-mod}_\eta(G)), O_{H,\eta}(1))$-protrusion decomposition (T, bag). For kernelization, it is in fact more instructive to think about $(T, \mathcal{L})$ than about (T, bag). For each child c of the root r of T, we consider the *boundaried graph* G_c, having vertex set $V(G_c) = V(\mathcal{L}[c])$, i.e., the union of the hyperedges in $\mathcal{L}[c]$, and edge set $E(G_c)$ being the edges of G in $\mathcal{L}[c]$, and boundary $\text{bd}(\mathcal{L}[c])$.

Now, the idea is to replace the boundaried graph G_c by a *representative* of bounded size. In particular, we replace G_c by a boundaried graph R_c with the same boundary as G_c, but which has bounded size (bounded by a function of the size of the boundary, the problem Π, and the class $\mathcal{G}$). If we denote by G' the graph obtained after this replacement, we want that $\text{OPT}_\Pi(G') = \text{OPT}_\Pi(G) - \Delta_c$, for a non-negative *shifting constant* Δ_c that can be computed from G_c. We also want that $G' \in \mathcal{G}$ if $G \in \mathcal{G}$.

The known properties of FII and CMSO_2 imply that such a representative R_c indeed exists (see e.g. [13]), and can be computed by dynamic programming on the tree decomposition of G_c, together with the shifting constant Δ_c. This dynamic programming can indeed be maintained on the bounded-treewidth subtrees of $(T, \mathcal{L})$ by the dynamic treewidth data structure, so with our data structure we can maintain the representatives R_c and shifting constants Δ_c for all children c of the root.

Now, the kernel is obtained simply as the union of the boundaried graphs R_c (which may overlap at the boundary vertices). As these graphs have constant size, and there are at most $O_{H,\eta}(\text{tw-mod}_\eta(G))$ of them, the size of the kernel is $O_{H,\eta}(\text{tw-mod}_\eta(G))$. The final shifting constant Δ is obtained as the sum of the individual shifting constants Δ_c.

This kernel can be maintained with logarithmic update time and with a constant number of changes to it thanks to the properties guaranteed by the data structure of Theorem 1.2. In particular, it guarantees that one update to the graph causes updates only in a constant number of the subtrees rooted at the children c of the root, so the representatives R_c and shifting constants Δ_c need to be updated only for them. Both of them are re-computed in amortized logarithmic update time thanks to the dynamic treewidth data structure we maintain on the subtrees.

References

[1] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. 2004. Polynomial-time data reduction for dominating set. *J. ACM* 51, 3 (2004), 363–384. doi:10.1145/990308.990309

[2] Josh Alman, Matthias Mnich, and Virginia Vassilevska Williams. 2020. Dynamic Parameterized Problems and Algorithms. *ACM Trans. Algorithms* 16, 4 (2020), 45:1–45:46. doi:10.1145/3395037

[3] Shinwoo An, Kyungjin Cho, Leo Jang, Byeonghyeon Jung, Yudam Lee, Eunjin Oh, Donghun Shin, Hyeonjun Shin, and Chanho Song. 2024. Dynamic Parameterized Problems on Unit Disk Graphs. In *Proceedings of the 35th International Symposium on Algorithms and Computation, ISAAC 2024 (LIPIcs, Vol. 322)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:15. doi:10.4230/LIPICS.ISAAC.2024.6

[4] Brenda S. Baker. 1994. Approximation Algorithms for NP-Complete Problems on Planar Graphs. *J. ACM* 41, 1 (1994), 153–180. doi:10.1145/174644.174650

[5] Max Bannach, Zacharias Heinrich, Rüdiger Reischuk, and Till Tantau. 2022. Dynamic Kernels for Hitting Sets and Set Packing. *Algorithmica* 84, 11 (2022), 3459–3488. doi:10.1007/S00453-022-00986-0

[6] Julien Baste, Ignasi Sau, and Dimitrios M. Thilikos. 2023. Hitting Minors on Bounded Treewidth Graphs. IV. An Optimal Algorithm. *SIAM J. Comput.* 52, 4 (2023), 865–912. doi:10.1137/21M140482X

[7] Christian Bertram, Deborah Haun, Mads Vestergaard Jensen, and Tuukka Korhonen. 2025. Dynamic Meta-Kernelization. *CoRR* abs/2511.03461 (2025). arXiv:2511.03461 doi:10.48550/ARXIV.2511.03461

[8] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F. Italiano. 2018. Deterministic Fully Dynamic Data Structures for Vertex Cover and Matching. *SIAM J. Comput.* 47, 3 (2018), 859–887. doi:10.1137/140998925

[9] Sayan Bhattacharya, Peter Kiss, Thatchaphol Saranurak, and David Wajc. 2024. Dynamic Matching with Better-than-2 Approximation in Polylogarithmic Update Time. *J. ACM* 71, 5 (2024), 33:1–33:32. doi:10.1145/3679009

[10] Hans L. Bodlaender. 1993. Dynamic Algorithms for Graphs with Treewidth 2. In *Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science, WG '93 (LNCS, Vol. 790)*. Springer, 112–124. doi:10.1007/3-540-57899-4_45

[11] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. 2009. On problems without polynomial kernels. *J. Comput. Syst. Sci.* 75, 8 (2009), 423–434. doi:10.1016/J.JCSS.2009.04.001

[12] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. 2009. (Meta) Kernelization. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*. IEEE, 629–638. doi:10.1109/FOCS.2009.46

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

[14] Hans L. Bodlaender and Ton Kloks. 1996. Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs. *J. Algorithms* 21, 2 (1996), 358–402. doi:10.1006/JAGM.1996.0049

[15] Hans L. Bodlaender and Eelko Penninkx. 2008. A Linear Kernel for Planar Feedback Vertex Set. In *Proceedings of the Third International Workshop on Parameterized and Exact Computation, IWPEC 2008 (LNCS, Vol. 5018)*. Springer, 160–171. doi:10.1007/978-3-540-79723-4_16

[16] Hans L. Bodlaender, Eelko Penninkx, and Richard B. Tan. 2008. A Linear Kernel for the k-Disjoint Cycle Problem on Planar Graphs. In *Proceedings of the 19th International Symposium on Algorithms and Computation, ISAAC 2008 (LNCS, Vol. 5369)*. Springer, 306–317. doi:10.1007/978-3-540-92182-0_29

[17] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. 1997. Advice Classes of Parameterized Tractability. *Ann. Pure Appl. Log.* 84, 1 (1997), 119–138. doi:10.1016/S0168-0072(95)00020-8

[18] Dimitris Chatzidimitriou, Jean-Florent Raymond, Ignasi Sau, and Dimitrios M. Thilikos. 2018. An $O(\log OPT)$-Approximation for Covering and Packing Minor Models of Θ_r. *Algorithmica* 80, 4 (2018), 1330–1356. doi:10.1007/S00453-017-0313-5

[19] Jiehua Chen, Wojciech Czerwinski, Yann Disser, Andreas Emil Feldmann, Danny Hermelin, Wojciech Nadara, Marcin Pilipczuk, Michal Pilipczuk, Manuel Sorge, Bartlomiej Wróblewski, and Anna Zych-Pawlewicz. 2021. Efficient fully dynamic elimination forests with applications to detecting long paths and cycles. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*. SIAM, 796–809. doi:10.1137/1.9781611976465.50

[20] Jianer Chen, Henning Fernau, Iyad A. Kanj, and Ge Xia. 2007. Parametric Duality and Kernelization: Lower Bounds and Upper Bounds on Kernel Size. *SIAM J. Comput.* 37, 4 (2007), 1077–1106. doi:10.1137/050646354

[21] Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. 2015. Parameterized Streaming: Maximal Matching and Vertex Cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*. SIAM, 1234–1251. doi:10.1137/1.9781611973730.82

[22] Konrad Kazimierz Dabrowski, Petr A. Golovach, Pim van 't Hof, Daniël Paulusma, and Dimitrios M. Thilikos. 2017. Editing to a planar graph of given degrees. *J. Comput. Syst. Sci.* 85 (2017), 168–182. doi:10.1016/J.JCSS.2016.11.009

[23] Holger Dell and Dieter van Melkebeek. 2014. Satisfiability Allows No Nontrivial Sparsification unless the Polynomial-Time Hierarchy Collapses. *J. ACM* 61, 4 (2014), 23:1–23:27. doi:10.1145/2629620

[24] Rodney G. Downey and Michael R. Fellows. 1995. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM J. Comput.* 24, 4 (1995), 873–921. doi:10.1137/S0097539792228228

[25] R. G. Downey and M. R. Fellows. 1999. *Parameterized Complexity*. Springer, New York, NY. doi:10.1007/978-1-4612-0515-9

[26] Zdenek Dvorák, Martin Kupec, and Vojtech Tuma. 2014. A Dynamic Data Structure for MSO Properties in Graphs with Bounded Tree-Depth. In *Proceedings of the 22nd Annual European Symposium on Algorithms, ESA 2014 (LNCS, Vol. 8737)*. Springer, 334–345. doi:10.1007/978-3-662-44777-2_28

[27] Zdenek Dvorák and Vojtech Tuma. 2013. A Dynamic Data Structure for Counting Subgraphs in Sparse Graphs. In *Proceedings of the 13th International Symposium on Algorithms and Data Structures, WADS 2013 (LNCS, Vol. 8037)*. Springer, 304–315. doi:10.1007/978-3-642-40104-6_27

[28] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, Geevarghese Philip, and Saket Saurabh. 2016. Hitting Forbidden Minors: Approximation and Kernelization. *SIAM J. Discret. Math.* 30, 1 (2016), 383–410. doi:10.1137/140997889

[29] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. 2015. Solving d-SAT via Backdoors to Small Treewidth. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*. SIAM, 630–641. doi:10.1137/1.9781611973730.43

[30] Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. 2012. Planar F-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*. IEEE, 470–479. doi:10.1109/FOCS.2012.62

[31] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. 2012. Linear kernels for (connected) dominating set on H-minor-free graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*. SIAM, 82–93. doi:10.1137/1.9781611973099.7

[32] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. 2018. Kernels for (Connected) Dominating Set on Graphs with Excluded Topological Minors. *ACM Trans. Algorithms* 14, 1 (2018), 6:1–6:31. doi:10.1145/3155298

[33] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. 2020. Bidimensionality and Kernels. *SIAM J. Comput.* 49, 6 (2020), 1397–1422. doi:10.1137/16M1080264

[34] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. 2019. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press.

[35] Fedor V. Fomin and Dimitrios M. Thilikos. 2004. Fast Parameterized Algorithms for Graphs on Surfaces: Linear Kernel and Exponential Speed-Up. In *Proceedings of 31st International Colloquium on Automata, Languages and Programming, ICALP 2004 (LNCS, Vol. 3142)*. Springer, 581–592. doi:10.1007/978-3-540-27836-8_50

[36] Lance Fortnow and Rahul Santhanam. 2011. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.* 77, 1 (2011), 91–106. doi:10.1016/J.JCSS.2010.06.007

[37] Harmender Gahlawat, Abhishek Rathod, and Meirav Zehavi. 2025. (Almost-)optimal FPT Algorithm and Kernel for T-cycle on Planar Graphs. In *Proceedings of the 52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025 (LIPIcs, Vol. 334)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 82:1–82:18. doi:10.4230/LIPICS.ICALP.2025.82

[38] Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. 2017. Kernelization using structural parameters on sparse graph classes. *J. Comput. Syst. Sci.* 84 (2017), 219–242. doi:10.1016/J.JCSS.2016.09.002

[39] Valentin Garnero, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. 2015. Explicit Linear Kernels via Dynamic Programming. *SIAM J. Discret. Math.* 29, 4 (2015), 1864–1894. doi:10.1137/140968975

[40] Valentin Garnero, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. 2019. Explicit Linear Kernels for Packing Problems. *Algorithmica* 81, 4 (2019), 1615–1656. doi:10.1007/S00453-018-0495-5

[41] Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. 2021. Linear Kernels for Edge Deletion Problems to Immersion-Closed Graph Classes. *SIAM J. Discret. Math.* 35, 1 (2021), 105–151. doi:10.1137/18M1228839

[42] Petr A. Golovach, Giannos Stamoulis, and Dimitrios M. Thilikos. 2023. Hitting Topological Minor Models in Planar Graphs is Fixed Parameter Tractable. *ACM Trans. Algorithms* 19, 3 (2023), 23:1–23:29. doi:10.1145/3583688

[43] Jiong Guo and Rolf Niedermeier. 2007. Linear Problem Kernels for NP-Hard Problems on Planar Graphs. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming, ICALP 2007 (LNCS, Vol. 4596)*. Springer, 375–386. doi:10.1007/978-3-540-73420-8_34

[44] Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. 2006. Fixed-Parameter Tractability Results for Full-Degree Spanning Tree and Its Dual. In *Proceedings of the Second International Workshop on Parameterized and Exact Computation, IWPEC 2006 (LNCS, Vol. 4169)*. Springer, 203–214. doi:10.1007/11847250_19

[45] Yoichi Iwata and Keigo Oka. 2014. Fast Dynamic Graph Algorithms for Parameterized Problems. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2014 (LNCS, Vol. 8503)*. Springer, 241–252. doi:10.1007/978-3-319-08404-6_21

[46] Bart M. P. Jansen and Michal Wlodarczyk. 2025. Lossy Planarization: A Constant-Factor Approximate Kernelization for Planar Vertex Deletion. *SIAM J. Comput.* 54, 1 (2025), 1–91. doi:10.1137/22M152058X

[47] Gwénaël Joret, Christophe Paul, Ignasi Sau, Saket Saurabh, and Stéphan Thomassé. 2014. Hitting and harvesting pumpkins. *SIAM Journal on Discrete Mathematics* 28, 3 (2014), 1363–1390. doi:10.1137/120883736

[48] Iyad A. Kanj, Michael J. Pelsmajer, Marcus Schaefer, and Ge Xia. 2011. On the induced matching problem. *J. Comput. Syst. Sci.* 77, 6 (2011), 1058–1070. doi:10.1016/J.JCSS.2010.09.001

[49] Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. 2016. Linear Kernels and Single-Exponential Algorithms Via Protrusion Decompositions. *ACM Trans. Algorithms* 12, 2 (2016), 21:1–21:41. doi:10.1145/2797140

[50] Eun Jung Kim, Christophe Paul, and Geevarghese Philip. 2015. A single-exponential FPT algorithm for the K_4-minor cover problem. *J. Comput. Syst. Sci.* 81, 1 (2015), 186–207. doi:10.1016/J.JCSS.2014.05.001

[51] Eun Jung Kim, Maria J. Serna, and Dimitrios M. Thilikos. 2018. Data-Compression for Parametrized Counting Problems on Sparse Graphs. In *Proceedings of the 29th International Symposium on Algorithms and Computation, ISAAC 2018 (LIPIcs, Vol. 123)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 20:1–20:13. doi:10.4230/LIPICS.ISAAC.2018.20

[52] Tuukka Korhonen. 2025. Dynamic Treewidth in Logarithmic Time. *CoRR* abs/2504.02790 (2025). arXiv:2504.02790 doi:10.48550/ARXIV.2504.02790 To appear in FOCS 2025..

[53] Tuukka Korhonen. 2025. Linear-Time Algorithms for k-Edge-Connected Components, k-Lean Tree Decompositions, and More. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025*. ACM, 111–119. doi:10.1145/3717823.3718123

[54] Tuukka Korhonen, Konrad Majewski, Wojciech Nadara, Michal Pilipczuk, and Marek Sokolowski. 2023. Dynamic treewidth. In *Proceedings of the 64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*. IEEE, 1734–1744. doi:10.1109/FOCS57990.2023.00105

[55] Tuukka Korhonen, Wojciech Nadara, Michal Pilipczuk, and Marek Sokolowski. 2024. Fully dynamic approximation schemes on planar and apex-minor-free graphs. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*. SIAM, 296–313. doi:10.1137/1.9781611977912.12

[56] Daniel Lokshtanov, Matthias Mnich, and Saket Saurabh. 2011. A linear kernel for planar connected dominating set. *Theor. Comput. Sci.* 412, 23 (2011), 2536–2543. doi:10.1016/J.TCS.2010.10.045

[57] Daniel Lokshtanov, Ramanujan Maadapuzhi Sridharan, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. 2025. Wannabe Bounded Treewidth Graphs Admit a Polynomial Kernel for Directed Feedback Vertex Set. *ACM Trans. Comput. Theory* 17, 1 (2025), 2:1–2:28. doi:10.1145/3711669

[58] Konrad Majewski, Michal Pilipczuk, and Marek Sokolowski. 2025. Maintaining $CMSO_2$ properties on dynamic structures with bounded feedback vertex number. *ACM Trans. Comput. Theory* 17, 2 (2025), 8:1–8:72. doi:10.1145/3715884

[59] Konrad Majewski, Michal Pilipczuk, and Anna Zych-Pawlewicz. 2024. Parameterized Dynamic Data Structure for Split Completion. In *Proceedings of the 32nd Annual European Symposium on Algorithms, ESA 2024 (LIPIcs, Vol. 308)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 87:1–87:17. doi:10.4230/LIPICS.ESA.2024.87

[60] Hannes Moser and Somnath Sikdar. 2007. The Parameterized Complexity of the Induced Matching Problem in Planar Graphs. In *Proceedings of the First Annual International Workshop on Frontiers in Algorithmics, FAW 2007 (LNCS, Vol. 4613)*. Springer, 325–336. doi:10.1007/978-3-540-73814-5_32

[61] Jedrzej Olkowski, Michal Pilipczuk, Mateusz Rychlicki, Karol Wegrzycki, and Anna Zych-Pawlewicz. 2023. Dynamic Data Structures for Parameterized String Problems. In *Proceedings of the 40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023 (LIPIcs, Vol. 254)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 50:1–50:22. doi:10.4230/LIPICS.STACS.2023.50

[62] Mihai Pătraşcu and Erik D. Demaine. 2006. Logarithmic Lower Bounds in the Cell-Probe Model. *SIAM J. Comput.* 35, 4 (2006), 932–963. doi:10.1137/S0097539705447256

[63] Neil Robertson and Paul D. Seymour. 1991. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory B* 52, 2 (1991), 153–190. doi:10.1016/0095-8956(91)90061-N

[64] Shay Solomon. 2016. Fully Dynamic Maximal Matching in Constant Update Time. In *Proceedings of the 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*. IEEE, 325–334. doi:10.1109/FOCS.2016.43

[65] Stéphan Thomassé. 2010. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms* 6, 2 (2010), 32:1–32:8. doi:10.1145/1721837.1721848

Received 2025-11-04; accepted 2026-02-01