Turing Kernelization for Finding Long Paths in Graph Classes Excluding a Topological Minor*†

Bart M. P. Jansen[1], Marcin Pilipczuk[‡2], and Marcin Wrochna[§3]

1 **Eindhoven University of Technology, The Netherlands**
2 **University of Warsaw, Poland**
3 **University of Warsaw, Poland**

Abstract

The notion of Turing kernelization investigates whether a polynomial-time algorithm can solve an NP-hard problem, when it is aided by an oracle that can be queried for the answers to bounded-size subproblems. One of the main open problems in this direction is whether k-Path admits a polynomial Turing kernel: can a polynomial-time algorithm determine whether an undirected graph has a simple path of length k, using an oracle that answers queries of size $k^{\mathcal{O}(1)}$?

We show this can be done when the input graph avoids a fixed graph H as a topological minor, thereby significantly generalizing an earlier result for bounded-degree and $K_{3,t}$-minor-free graphs. Moreover, we show that k-Path even admits a polynomial Turing kernel when the input graph is not H-topological-minor-free itself, but contains a known vertex modulator of size bounded polynomially in the parameter, whose deletion makes it so. To obtain our results, we build on the graph minors decomposition to show that any H-topological-minor-free graph that does not contain a k-path has a separation that can safely be reduced after communication with the oracle.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Turing kernel, long path, k-path, excluded topological minor, modulator

Digital Object Identifier 10.4230/LIPIcs.IPEC.2017.23

1 Introduction

Suppose that Alice is a polynomial-time agent faced with an input to an NP-hard problem that she wishes to solve exactly. To facilitate her in this process, she can ask questions to an all-knowing oracle. These will be answered truthfully and instantly, but the oracle is memory-less and will not take previous questions into account when answering the next one. How large do these questions have to be, to allow Alice to find the answer to her problem? Clearly, the answer can be established by sending the entire input to the oracle, who determines the answer and sends it to Alice. Could there be a more clever strategy? Alice can attempt to isolate a small but meaningful question about the behavior of her input,

* This work was supported by the Netherlands Organization for Scientific Research (NWO) Veni grant 639.021.437 "Frontiers in Parameterized Preprocessing" and Gravitation grant 024.002.003 "Networks".
† A full version of the paper is available at [18], https://arxiv.org/abs/1707.01797.
‡ Marcin Pilipczuk is supported by the "Recent trends in kernelization: theory and experimental evaluation" project, carried out within the Homing programme of the Foundation for Polish Science co-financed by the European Union under the European Regional Development Fund.
§ Marcin Wrochna is supported by the National Science Centre of Poland grant number 2013/11/D/ST6/03073 and by the Foundation for Polish Science (FNP) via the START stipend programme.

such that after learning its answer, she can reduce to a smaller input without changing the outcome. Iterating this process solves her problem: when it has become sufficiently small, it can be posed to the oracle in its entirety.

Such problem-solving strategies can be rigorously analyzed using the notion of *Turing kernelization* that originated in parameterized algorithmics. The parameter makes it possible to express how the size of the questions that Alice asks, depends on properties of the input that she is given. (See Section 3 for a formal definition.)

Understanding the power of Turing kernelization is one of the main open research horizons in parameterized algorithmics. There is a handful of problems for which a nontrivial Turing kernelization is known [1, 2, 3, 5, 12, 15, 17, 19, 22, 24]. On the other hand, there is a hierarchy of parameterized complexity classes which are conjectured not to admit polynomial Turing kernels [14]. Arguably, the main open problem (cf. [4, 3, 14]) in this direction is to determine whether the k-Path problem (determine whether an undirected graph has a simple path of length k) has a polynomial Turing kernel. In earlier work [16], the first author showed that k-Path indeed admits polynomial Turing kernels on several graph classes. In this work, we develop Turing kernels for k-Path in a much more general setting.

Our results. Our algorithmic contributions are twofold. First of all, we extend the Turing kernelization for k-Path to much broader families of sparse graphs. Whereas the earlier work could only deal with $K_{3,t}$-minor-free graphs, claw-free graphs, and bounded-degree graphs, we show that a Turing kernelization exists on H-minor-free graphs for all fixed graphs H. We even lift the kernelization to H-topological-minor-free graphs, thereby capturing a common generalization of the bounded-degree and $K_{3,t}$-minor-free cases.

▶ **Theorem 1.** *For every fixed graph H, the k-Path problem, restricted to graphs excluding H as a topological minor, admits a polynomial Turing kernel. Furthermore, the kernel runs in time $k^{\mathcal{O}_H(1)}n^2m$ and invokes $k^{\mathcal{O}_H(1)} \cdot n$ calls to the oracle.*

Our second contribution is the following theorem. By a novel algorithmic approach, we obtain a Turing kernelization even when the input graph does not belong to the desired restricted graph class itself, but contains a small known vertex modulator whose deletion places the graph in such a graph class.

▶ **Theorem 2.** *For every fixed graph H, the k-Path problem, on instances consisting of a graph G, integer k, and a modulator $M \subseteq V(G)$ such that $G - M$ is H-topological-minor-free, admits a polynomial Turing kernel, when parameterized by k and $|M|$.*

Techniques. To explain our approach, we briefly recall the idea behind the Turing kernelization for k-Path on planar graphs. At the core lies a win/win: there is a polynomial-time algorithm that either (i) establishes that a planar graph G has a k-path (a simple path on k vertices), or (ii) finds a separation (A, B) in G with the following property: the size of A is polynomially bounded in k, but large enough that after marking a witness structure for each reasonable way in which a k-path might intersect A, some vertex remains unmarked. Using bounded-size oracle queries to mark the witness structures, this allows the problem to be simplified by removing an unmarked vertex from A without changing the answer.

Theorem 1 is established by lifting this win/win approach to H-(topological)-minor-free graphs. This requires an adaptation of the decomposition theorems of Robertson and Seymour [21] (for minors) and of Grohe and Marx [13] (for topological minors), to obtain the following. Every H-free graph that does not have a k-path, has a tree decomposition of constant adhesion and width $\text{poly}(k)$. A reducible separation can be found by inspecting

this tree decomposition. To establish this result, we exploit known theorems stating that triconnected n-vertex graphs that exclude $K_{3,t}$ as a minor for some t [7], contain paths of length $\Omega(n^{\varepsilon})$ for some $\varepsilon > 0$. Roughly speaking, this allows us to infer the existence of a k-path if there is a large embedded part in the nearly-embeddable graph corresponding to a bag of the graph minors decomposition, since graphs embeddable in a fixed surface are $K_{3,t}$-minor-free for some t. We use lower bounds on the circumference of graphs of bounded degree [6, 23] to achieve a similar conclusion from the existence of a large bounded-degree bag in the topological-minor-free decomposition. Several technical steps are needed to translate this into the desired win/win, due to the existence of vortices, virtual edges, and the lack of a direct polynomial-time algorithm to compute the decomposition.

To prove Theorem 2, we introduce a new algorithmic tool for finding irrelevant vertices for the k-Path problem in the presence of a modulator M in the input graph G. Since Theorem 1 can be applied to find a k-path in $G - M$ if one exists, the challenge is to detect a k-path in G that jumps between M and $G - M$ several times. The absence of a k-path in $G - M$ implies it has a tree decomposition of width $\mathrm{poly}(k)$ and constant adhesion. Using Theorem 1 as a subroutine, along with a packing argument, we can compute a vertex set X of size polynomial in $k + |M|$ with the following guarantee. If there is a k-path, then there is a *guarded* k-path P in which each successive pair of vertices in $M \cap P$ are connected by a subpath through $G - M$ that intersects X. Using the tree decomposition of $G - M$, the standard ancestor-marking technique allows us to identify a vertex subset C of $G - (M \cup X)$ that is adjacent to constantly many vertices from X. Unless G is already small, we can find such a set C that is sufficiently large to be reducible but small enough that we may invoke the oracle for questions about it. We can then reduce the graph without losing the existence of a guarded k-path, by marking a witness for each sensible way in which a constant-size subset from M can connect to prescribed vertices in X through C. The fact that C only has constantly many neighbors in X implies that there are only polynomially many relevant choices. We may then safely remove the unmarked vertices.

Organization. After preliminaries in Section 2, we give a generic Turing-style reduction rule for k-Path in Section 3. In Section 4 we show that an H-minor-free graph either has a k-path or a separation that is suitable for reduction. In Section 5 we extend this to topological minors. Finally, in Section 6 we present a Turing kernel applicable when the input graph has a small modulator to a suitable graph class. Proofs of statements marked with ♠ can be found in the full version [18].

2 Preliminaries

All graphs we consider are finite, simple, and undirected. A *separation* of a graph G is a pair (A, B), $A, B \subseteq V(G)$ such that $A \cup B = V(G)$ and there are no edges between $A \setminus B$ and $B \setminus A$. The *order* of the separation (A, B) is $|A \cap B|$. A graph is *triconnected* if it is connected and cannot be disconnected by deleting fewer than three vertices. When referring to the *size* of a graph in our statements, we mean the number of vertices.

A *tree decomposition* of a graph G is a pair $(T, \mathcal{X})$ where T is a rooted tree and $\mathcal{X}$ is a function that assigns to every node $t \in V(T)$ a subset $\mathcal{X}(t)$ of $V(G)$ called a *bag* such that:

- $\bigcup_{t \in V(T)} \mathcal{X}(t) = V(G)$;
- for each edge $uv \in E(G)$, there is a node $t \in V(T)$ with $u, v \in \mathcal{X}(t)$;
- for each $v \in V(G)$, the nodes $\{t \mid v \in \mathcal{X}(t)\}$ induce a (connected) subtree of T.

The *width* of $(T, \mathcal{X})$ is $\max_{t \in V(T)} |\mathcal{X}(t)| - 1$. Its *adhesion* is $\max_{tt' \in E(T)} |\mathcal{X}(t) \cap \mathcal{X}(t')|$. We also call the set $\mathcal{X}(t) \cap \mathcal{X}(t')$ the *adhesion of* tt', for every edge tt' of T. For a decomposition $(T, \mathcal{X})$ of G and a node $t \in V(T)$, the *torso*, denoted $\textsc{torso}(G, \mathcal{X}(t))$, is the graph obtained from $G[\mathcal{X}(t)]$ by adding an edge between each pair of vertices in $\mathcal{X}(t) \cap \mathcal{X}(t')$, for every neighbor t' of t in T (so each adhesion induces a clique in the torso). Added edges not present in G are called *virtual edges*. For a subtree $T' \subseteq T$ we write $\mathcal{X}(T')$ for the union $\bigcup_{t \in V(T')} \mathcal{X}(t)$ of bags in T'.

For an edge $t_1t_2 \in E(T)$, let T_i be the connected component of $T - \{t_1t_2\}$ that contains t_i. Let $V_i = \mathcal{X}(T_i)$. Observe that the properties of a tree decomposition imply that (V_1, V_2) is a separation with $V_1 \cap V_2 = \mathcal{X}(t_1) \cap X(t_2)$.

A decomposition $(T, \mathcal{X})$ is *connected* if for every $t \in V(T)$ and its child t', if $T_{t'}$ is the subtree of T rooted at t', we have (i) that $G[\mathcal{X}(T_{t'}) \setminus \mathcal{X}(t)]$ is connected, and (ii) that $\mathcal{X}(T_{t'}) \setminus \mathcal{X}(t)$ has edges to every vertex of the adhesion $\mathcal{X}(t) \cap \mathcal{X}(t')$. It is straightforward to turn any decomposition into a connected one without increasing its width nor adhesion.

We will also need the following non-standard complexity measure of a tree decomposition $(T, \mathcal{X})$. For every $t \in V(T)$, the number of distinct adhesions $\mathcal{X}(t) \cap \mathcal{X}(t')$ for $t' \in N_T(t)$ is called the *adhesion degree* of t. The maximum adhesion degree over all nodes t is the *adhesion degree* of the decomposition $(T, \mathcal{X})$. Observe that if a tree decomposition $(T, \mathcal{X})$ has width less than ℓ and adhesions of size at most h, then its adhesion degree is at most

$$\sum_{i=0}^{h} \binom{\ell}{i} \leq (1 + \ell)^h.$$

However, in sparse graph classes we can prove a much better bound on the adhesion degree due to linear bounds on the number of cliques in such graphs; cf. Lemma 17.

3 Turing kernels

In this section we introduce a general toolbox and notation for our Turing kernel bounds.

3.1 Definitions and the auxiliary problem

For a parameterized problem Π and a computable function f, a *Turing kernel of size* f is an algorithm that solves an input instance (x, k) of Π in polynomial time, given access to an oracle that solves instances (x', k') of Π with $|x'|, k' \leq f(k)$. A Turing kernel is a *polynomial* one if f is a polynomial.

If we are only interested in distinguishing between NP-complete problems admitting a polynomial Turing kernel from the ones that do not admit such a kernel, we can assume that the oracle solves an arbitrary problem in NP, not necessarily the k-Path problem. Indeed, note that by the definition of NP-completeness, an oracle to a problem in NP can be implemented with an oracle to k-Path with only polynomial blow-up in the size of the passed instances.

In our work, it will be convenient to reduce to the Auxiliary Linkage problem, defined as follows. The input consists of an undirected graph G', an integer k', a set of terminals $S \subseteq V(G')$, and a number of requests $R_1, R_2, \ldots, R_r$; a request is a set of at most two terminals. A path P_i in G is said to *satisfy a request* R_i if $V(P_i) \cap S = R_i$ and every vertex of $V(P_i) \cap S$ is an endpoint of P_i. With such an input, the Auxiliary Linkage problem asks for a sequence of r paths $P_1, P_2, \ldots, P_r$ such that P_i satisfies R_i for every $1 \leq i \leq r$, $|\bigcup_{i=1}^{r} V(P_i)| = k'$, and every vertex of $V(G) \setminus S$ is contained in at most one path P_i (i.e.,

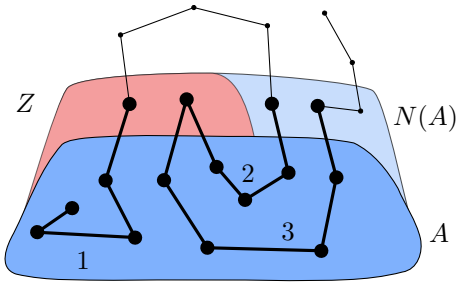

Figure 1 A set A (blue) and a path with three A-traverses (bold). The path is guarded w.r.t. $Z \subseteq N(A)$ (the red set), since each A-traverse has an endpoint in it. (A path fully contained in A (with one A-traverse) or disjoint from A (with no A-traverses) would also be guarded.)

the paths P_i are vertex-disjoint, except that they may share an endpoint, but only if the requests ask them to do so).

We remark that Auxiliary Linkage is a more general problem than k-Path: an instance with $G' = G$, $k' = k$, $S = \emptyset$, $r = 1$, and $R_1 = \emptyset$ asks precisely for a k-path in G.

Clearly, the decision version of the Auxiliary Linkage problem belongs to the class NP. By using its self-reducibility (cf. [16, Lemma 2]), we assume that the oracle returns a sequence of paths $(P_i)_{i=1}^r$ in case of a positive answer. That is, in all subsequent bounds on the number of Auxiliary Linkage oracle calls, the bound adheres to the number of calls to an oracle that returns the actual paths P_i; if one wants to use a decision oracle, one should increase the bound by the blow-up implied by the self-reducibility application (i.e., at most $|E(H)|$ for calls on a graph H).

3.2 Generic reduction rule

We now show a generic reduction rule for the k-Path problem. We start with a few definitions.

▶ **Definition 3.** For a graph G, a subset $A \subseteq V(G)$, and a simple path P in G, an *A-traverse* of P is a maximal subpath of P that contains at least one vertex of A and has all its internal vertices in A.

Note that if Q is an A-traverse of P, then every endpoint of Q is either an endpoint of P or lies in $N_G(A)$. See Figure 1.

▶ **Definition 4.** Let G be a graph, $A \subseteq V(G)$, and let k be an integer. A set $Z \subseteq N(A)$ is called a *k-guard of A* if the following implication holds: if G admits a k-path, then there exists a k-path P in G that is either contained in A or such that every A-traverse of P has at least one endpoint in Z.

Given a graph G, a set $A \subseteq V(G)$, and a k-guard $Z \subseteq N(A)$ of A, a k-path P satisfying properties as in the above definition is called *guarded* (w.r.t. k, A, and Z). If the integer k and the set A are clear from the context, we call such a set Z simply a guard.

Observe that $Z = N(A)$ is always a guard, but sometimes we will be able to find smaller ones. Of particular interest will be guards of constant size, as our kernel sizes will depend exponentially on the guard size. To describe our single reduction rule, we show how solutions to Auxiliary Linkage can be used to preserve the existence of guarded k-paths.

Assume we are given a graph G, a set $A \subseteq V(G)$, an integer k, and a k-guard $Z \subseteq N(A)$ of A. Let $h = |Z|$ and $\ell = |N(A)|$. Furthermore, assume that G admits a k-path, and let P be a guarded one w.r.t. A and Z. Let $(Q_1, Q_2, \ldots, Q_r)$ be the A-traverses of P, let

$R_i = V(Q_i) \setminus A = V(Q_i) \cap N(A)$ for $1 \leq i \leq r$, let $G' = G[N[A]]$, $S = N(A)$, and let $k' = |\bigcup_{i=1}^{r} V(Q_i)|$. Observe that $(Q_1, Q_2, \ldots, Q_r)$ is a feasible solution to the Auxiliary Linkage instance $\mathcal{I}_P := (G[N[A]], k', S, (R_i)_{i=1}^{r})$; the instance $\mathcal{I}_P$ is henceforth called *induced by P and A*. Furthermore, it is easy to see that if $(Q'_1, Q'_2, \ldots, Q'_r)$ is a different feasible solution to $\mathcal{I}_P$, then a path P' obtained from P by replacing every subpath Q_i with Q'_i is also a guarded k-path in G.

The crucial observation is that a small guard limits the number of A-traverses.

▶ **Lemma 5.** *The number r of traverses of the guarded k-path P is bounded by* $\max(1, 2|Z|)$.

Proof. Every vertex of Z can be an endpoint of at most two traverses. If $r > 1$, then none of the traverses Q_i are contained in $G[A]$, and thus every traverse has at least one endpoint in the guard Z. ◀

Lemma 5 in turn limits the number of possible instances $\mathcal{I}$ that can be induced by a guarded k-path, for a fixed set A and guard Z. Note that we have $0 \leq k' \leq k$ and $0 \leq r \leq \max(1, 2|Z|)$. Furthermore, unless $r = 1$ and $R_1 = \emptyset$, we have $R_i \subseteq N(A)$, $|R_i| \in \{1, 2\}$, and every set R_i needs to have at least one element of Z; there are at most $|Z| \cdot (|N(A)| + 1) = h(\ell + 1)$ choices for such a set R_i. Consequently, the number of possibilities for the instance $\mathcal{I}$ is at most

$$(k+1) \cdot \left(1 + \sum_{r=0}^{2h} h^r(\ell+1)^r\right) \leq (k+1) \cdot (h(\ell+1))^{2h+1} =: \mathfrak{p}(k, \ell, h). \tag{1}$$

Reduction rule

If $|A| > k \cdot \mathfrak{p}(k, \ell, h)$, then we can apply the following reduction rule. For each Auxiliary Linkage instance $\mathcal{I}$ out of at most $\mathfrak{p}(k, \ell, h)$ reasonable instances for A-traverses of a guarded k-path in G, we invoke an oracle on the instance $\mathcal{I}$, and mark the vertices of the solution if the oracle finds one. The whole process will mark at most $k \cdot \mathfrak{p}(k, \ell, h) < |A|$ vertices, thus at least one vertex of $|A|$ will remain unmarked. We delete any such vertices.

The observation that on a guarded k-path P one can replace a solution to the instance $\mathcal{I}_P$ induced by P and A by a different solution provides safeness of this reduction. Finally, note that the reduction invokes at most $\mathfrak{p}(k, \ell, h)$ calls to the oracle; each call operates on a subgraph of the graph $G[N[A]]$ with $k' \leq k$ and $r \leq 2|Z|$.

We shall apply the Reduction Rule for a medium-sized set A and a guard set Z of constant size formed from adhesions of a tree decomposition. For most of the paper we will use $Z = N(A)$ with $\ell = h = |Z|$ a constant (depending on the excluded (topological) minor, in the results of Sections 4 and 5). Only in Section 6, when dealing with a modulator M such that $G - M$ has an appropriate structure, it will be important to consider $N(A)$ potentially containing all of M, with a guard set Z of constant size disjoint from M.

3.3 Separation oracles

The natural way of using our reduction rule is to find in a graph a large (but not too large) part of the graph with a small (preferably, constant) boundary. Let us first make an abstract definition of an algorithm finding such a separation.

▶ **Definition 6.** For a graph class $\mathcal{G}$, a constant h, and a computable coordinate-wise nondecreasing function $q : \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$, an algorithm $\mathcal{S}$ is called a $(h, q, T_{\mathcal{S}})$-*separation oracle* if, given a graph $G \in \mathcal{G}$ and integers k and p, in time $T_{\mathcal{S}}(|G|, k, p)$ it finds a separation (A, B) in G of order at most h with $p < |A| \leq q(k, p)$, or correctly concludes that G contains a k-path.

For all considered graph classes, we will be able to provide a separation oracle with q being a polynomial. This, in turn, allows the following generic Turing kernel.

▶ **Lemma 7.** *Let $\mathcal{S}$ be a $(h, q, T_{\mathcal{S}})$-separation oracle for a hereditary graph class $\mathcal{G}$. Take $\hat{h} := (2h)^{4h+3}$. Then, the k-Path problem restricted to graphs from $\mathcal{G}$ can be solved:*

- *in time $\mathcal{O}\big(T_{\mathcal{S}}(|G|, k, k^2\hat{h}) \cdot |V(G)| \ + \ k\hat{h} \cdot |V(G)| \cdot |E(G)|\big)$,*
- *using at most $k\hat{h} \cdot |V(G)|$ calls to* Auxiliary Linkage,
- *each call on an induced subgraph of the input graph of size at most $q(k, k^2\hat{h})$.*

Proof. Let $p = k \cdot \mathfrak{p}(k, h, h) + h \leq k(k+1)(h(h+1))^{2h+1} + h \leq 2k^2(2h)^{4h+2} \leq k^2\hat{h}$.

As long as $|V(G)| > p$, we proceed as follows. Invoke algorithm $\mathcal{S}$ on G. If $\mathcal{S}$ claims that G admits a k-Path, we simply output the answer yes. Otherwise, let (A', B') be the separation output by $\mathcal{S}$. Apply the Reduction Rule for k, $A := A' \setminus B'$, and $Z = N(A) \subseteq A' \cap B'$. Note that as $|Z| \leq h$, the Reduction Rule deletes at least one vertex of A. Furthermore, the Reduction Rule invokes at most $\mathfrak{p}(k, h, h) \leq k\hat{h}$ calls to the oracle, each call on an induced subgraph of G of size at most $|A'| \leq q(k, p) \leq q(k, k^2\hat{h})$.

Once we obtain $|V(G)| \leq p$, we solve the instance using a single call to Auxiliary Linkage with $k' = k$, $r = 1$, and $R_1 = \emptyset$. The bounds follow, as there are at most $|V(G)|$ applications of the Reduction Rule, and each call to the oracle takes $\mathcal{O}(|E(G)|)$ time to prepare the instance and parse the output. ◀

Note that for any graph class where separations as in Definition 6 exist, there exists a trivial separation oracle which finds them, running in time $n^{h+\mathcal{O}(1)}$: one iterates over every candidate for $A \cap B$ and, for fixed set $A \cap B$, a straightforward knapsack-type dynamic programming algorithm checks if one can assemble $A \setminus B$ of the desired size from the connected components of $G - (A \cap B)$.

However, this running time bound is unsatisfactory, as it greatly exceeds the number of used oracle calls. For all considered graph classes we prove a much stronger property than just merely the prerequisites of Lemma 7, in particular providing a more efficient separation oracle. We provide necessary definitions in the next section.

3.4 Decomposable graph classes

▶ **Definition 8.** For a constant h and a computable nondecreasing function $w : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$, a graph class $\mathcal{G}$ is called (w, h)-*decomposable* if for every positive integer k and every $G \in \mathcal{G}$ that does not admit a k-path, the graph G admits a tree decomposition of width less than $w(k)$ and adhesions of size at most h.

A standard argument shows that in a decomposable graph class, given the decomposition with appropriate parameters, it is easy to provide a separation oracle.

▶ **Lemma 9.** *Assume we are given a graph G and a tree decomposition $(T, \mathcal{X})$ of G of width less than w, adhesion at most h, and adhesion degree at most $a \geq 2$. Then, given an integer p such that $|V(G)| > p$, one can in time $h^{\mathcal{O}(1)} \cdot (|V(G)| + |E(G)| + |V(T)| + \sum_{t \in V(T)} |\mathcal{X}(t)|)$ find a separation (A, B) of order at most h such that*

$$p < |A| \leq w + p \cdot a.$$

Proof. Root the tree T in an arbitrary node, and for $t \in V(T)$ let T_t be the subtree of T rooted in t. Let t_0 be the lowest node of T such that $|\mathcal{X}(V(T_{t_0}))| > p$; such a node can be computed in linear time in the size of G and $(T, \mathcal{X})$.

Group the children t' of t_0 according to their adhesions $\mathcal{X}(t') \cap \mathcal{X}(t_0)$. Due to the bound on the adhesion degree, there are at most a groups. For every adhesion S, let $X_{t_0,S}$ be the set of the children of t_0 with $S = \mathcal{X}(t') \cap \mathcal{X}(t_0)$. Define $V_S = \bigcup_{t' \in X_{t_0,S}} \mathcal{X}(T_{t'})$.

We consider now two cases. First, assume that $|V_S| \leq p$ for every adhesion S. Then, by the adhesion degree bound, we have $|\mathcal{X}(T_{t_0})| \leq |\mathcal{X}(t_0)| + ap \leq w + ap$. Consequently, we can return the separation (A, B) with $A = \mathcal{X}(T_{t_0})$ and $B = \mathcal{X}(T - V(T_{t_0}))$.

In the other case, there exists an adhesion S with $|V_S| > p$. We greedily take a minimal subset $Y_{t_0,S} \subseteq X_{t_0,S}$ such that $V'_S := \bigcup_{t' \in Y_{t_0,S}} \mathcal{X}(T_{t'})$ is of size greater than p. By the minimality of t_0, for every $t' \in X_{t_0,S}$ we have $|\mathcal{X}(T_{t'})| \leq p$ and, consequently $|V'_S| \leq 2p$. Thus, we can return the separation (A, B) for $A = V'_S$ and $B = N_G[V(G) \setminus V'_S]$, as then $A \cap B \subseteq S$. ◀

A critical insight is that the decomposition used by Cygan et al. [8] to solve the Minimum Bisection problem in fact provides an approximate decomposition in a decomposable graph class. Let us first recall the main technical result of [8].

▶ **Definition 10.** A vertex set $X \subseteq V(G)$ of a graph G is called (q, h)-*unbreakable* if every separation (A, B) of order at most h satisfies $|(A \setminus B) \cap X| \leq q$ or $|(B \setminus A) \cap X| \leq q$.

▶ **Theorem 11** ([8])**.** *There is an algorithm that given a graph G and integer h runs in time $2^{\mathcal{O}(h^2)}|V(G)|^2|E(G)|$ and outputs a connected tree decomposition $(T, \mathcal{Y})$ of G such that: (i) for each $t \in V(T)$, the bag $\mathcal{Y}(t)$ is $(2^{\mathcal{O}(h)}, h)$-unbreakable in G, and (ii) for each $tt' \in E(T)$ the adhesion $\mathcal{Y}(t) \cap \mathcal{Y}(t')$ has at most $2^{\mathcal{O}(h)}$ vertices and is $(2h, h)$-unbreakable in G.*

An unbreakable set can be bounded using only the existence of an appropriate tree decomposition. Thus, the decomposition computed by Theorem 11 approximates the decomposition required in a decomposable graph class:

▶ **Lemma 12** (♠)**.** *Let G be a graph and suppose there exists a decomposition $(T, \mathcal{X})$ of G of width less than w, adhesion h, and adhesion degree a. Let $(T', \mathcal{Y})$ be a tree decomposition of G such that for each $t \in V(T')$, the bag $\mathcal{Y}(t)$ is $(2^{\mathcal{O}(h)}, h)$-unbreakable in G. Then $|\mathcal{Y}(t)| \leq w + a \cdot 2^{\mathcal{O}(h)}$.*

▶ **Corollary 13.** *Let $\mathcal{G}$ be a (w, h)-decomposable graph class. Then, for every $G \in \mathcal{G}$ and $k \in \mathbb{N}$, one can in $2^{\mathcal{O}(h^2)}|V(G)|^2|E(G)|$ time either correctly conclude that G admits a k-path, or find a tree decomposition of G of width at most $(w(k)+1)^{\mathcal{O}(h)}$ and adhesion at most $2^{\mathcal{O}(h)}$.*

Let us now combine all the above. That is, given an integer k and a graph G from a hereditary (w, h)-decomposable graph class $\mathcal{G}$, we start by computing the tree decomposition of Corollary 13 (or conclude there is a k-path). In general this approximated decomposition has adhesion degree $(w(k) + 1)^{2^{\mathcal{O}(h)}}$. We use this decomposition to find separations of any induced subgraphs of G using the algorithm of Lemma 9 in time $2^{\mathcal{O}(h)}$ times linear in the size of G and the computed decomposition. This gives a (h, q, T)-separation oracle with $q(k, p) = p \cdot (w(k) + 1)^{2^{\mathcal{O}(h)}}$ and $T(n, k, p) = 2^{\mathcal{O}(h)} \cdot n \cdot (w(k) + 1)^{\mathcal{O}(h)}$, for any hereditary (w, h)-decomposable graph class. By plugging it into Lemma 7, we obtain the following.

▶ **Corollary 14.** *Let $\mathcal{G}$ be a hereditary (w, h)-decomposable graph class. Then, the k-Path problem, restricted to graphs from $\mathcal{G}$, can be solved in time $2^{2^{\mathcal{O}(h)}}|V(G)|^2|E(G)|$ using $2^{2^{\mathcal{O}(h)}}kn$ calls to* Auxiliary Linkage *on induced subgraphs of the input graph of size $k^2(1+w(k))^{2^{\mathcal{O}(h)}}$.*

We would like to remark that we do not want to claim in this paper the idea that, in the context of H-(topological)-minor-free graphs, the decomposition of Theorem 11 should be related to the decomposition of the Global Structure Theorem via an argument as in the proof of Lemma 12. In particular, this observation appeared previously in a work of the second author with Daniel Lokshtanov, Michał Pilipczuk, and Saket Saurabh [20].

4 Excluding a minor

In this section we tackle proper minor-closed graph classes, that is, we prove Theorem 1 for graph classes excluding a fixed minor, by proving the following.

▶ **Theorem 15.** *For every fixed graph H, the k-Path problem restricted to H-minor-free graphs can be solved in time $\mathcal{O}_H(n^2m)$ using $\mathcal{O}_H(kn)$ calls to* Auxiliary Linkage *on instances being induced subgraphs of the input graph of size $\mathcal{O}_H(k^{24})$.*

Our main technical result is the following:

▶ **Theorem 16 (♠).** *For every H, the class of H-minor-free graphs is (w,h)-decomposable for $w(k) = \mathcal{O}_H(k^{22})$ and $h = \mathcal{O}_H(1)$.*

By plugging the above into Corollary 14, we obtain the desired polynomial Turing kernel, but with worse bounds than promised by Theorem 15. To obtain better bounds, we need to recall the folklore bound on the adhesion degree in sparse graph classes.

▶ **Lemma 17 (♠).** *Let G be a graph not containing H as a topological minor, and let $(T,\mathcal{X})$ be a connected tree decomposition of G of width less than ℓ and adhesion h. Then the adhesion degree of $(T,\mathcal{X})$ is bounded by $f(h,H)\cdot\ell$ for some integer $f(h,H)$ depending only on h and H.*

This way, we conclude that H-minor-free graphs without k-paths have tree decompositions of width $\mathcal{O}_H(k^{22})$, adhesion $\mathcal{O}_H(1)$ and adhesion degree $\mathcal{O}_H(k^{22})$, which we can approximate with Theorem 11. Then Theorem 15 follows from Lemma 7 if we find separations applying Lemma 9 to this decomposition.

Thus, it remains to prove Theorem 16. For the proof, we use the graph minors structure theorem, decomposing an H-minor-free graph G into parts 'nearly embeddable' in surfaces (precise definitions are given in the full version). By carefully analyzing details of the structure, we either find a large triconnected embedded part, which must contain a long path by the following theorem of Chen et al. [7], or we tighten the graph structure to give a tree decomposition where all parts are small (polynomial in k) and adhesions ('boundaries') between them are of constant size.

▶ **Theorem 18** ([7]). *There is a constant $\varepsilon > 0$ such that for every integer t, every triconnected graph on $n \geq 3$ vertices embeddable in a surface of (Euler) genus g contains a cycle of length at least $n^{\varepsilon}/2^{(2g+3)^2}$.*

Two intertwined problems that arise with this approach is that torsos of decompositions are not necessarily triconnected, and long paths in them do not necessarily imply long paths in the original graph, because of virtual edges added in torsos. Torsos can be made triconnected if their near-embeddings include cycles or paths around each vortex, but these may use virtual edges in essential ways. On the other hand, the decomposition can be modified so that virtual edges can be replaced with paths in the original graph, but this requires changes that remove virtual edges, hence potentially removing paths around vortices and destroying triconnectedness.

Because of that, we need to go a little deeper and use a local, strong version of the structure theorem from Graph Minors XVII [21]. For the same reason we cannot use existing algorithms for finding the graph minors decompositions. Instead, we only prove the *existence* of a tree decomposition of bounded adhesion, small width, and with nearly embeddable bags.

The statements of [21] (as exposed in [10]) allow us to assume that: (i) any virtual edges in torsos coming from distinct adhesions (from distinct parts of the decomposition) can be

replaced by paths, (ii) every large vortex has a path of at least the same length, (iii) a torso of the decomposition is triconnected, even if each vortex is replaced by a wheel (a cycle plus a universal vertex). These ingredients allow us to use Theorem 18 on this variant of the torso. Thus, if it contains a long path, we prove the original graph must also contain one, otherwise we conclude the bound required in Theorem 16.

5 Excluding a topological minor

In this section we tackle graph classes excluding a topological minor, that is, we prove Theorem 1 by proving the following.

▶ **Theorem 19.** *For every fixed graph H, the k-Path problem restricted to H-topological-minor-free graphs can be solved in time $\mathcal{O}_H(n^2m)$ using $\mathcal{O}_H(kn)$ calls to* Auxiliary Linkage *on instances being induced subgraphs of the input graph of size $k^{\mathcal{O}_H(1)}$.*

This follows as before from the following decomposability theorem. Note the exponent in the polynomial bound on width (bag size) now depends on H.

▶ **Theorem 20 (♠).** *For every graph H, the class of H-topological-minor-free graphs is (w,h)-decomposable for $w(k) = k^{\mathcal{O}_H(1)}$ and $h = \mathcal{O}_H(1)$.*

To prove the above theorem, we use the structure theorem of Grohe and Marx [13]: when excluding a topological minor, graphs admit a similar structure as for excluding a minor, but apart from nearly embeddable parts, one needs to consider parts that have bounded degree except for a bounded number of vertices. By appropriately contracting everything outside such a part, we obtain a single graph of almost bounded degree. We remove the few vertices of high degree, find the Tutte decomposition into triconnected components, which can be bounded using the following theorem by Shan [23] (see also Chen et al. [6]).

▶ **Theorem 21** ([23])**.** *If G is a triconnected graph with maximum degree at most $\Delta \geq 425$, then G has a cycle of length at least $n^{1/\log_2(\Delta-1)}/4 + 2$.*

Such a small-width Tutte decomposition can then be lifted back to a decomposition of the part we originally considered. Finally, arguments involving tangles allow us to conclude that such a part can be assumed to be small itself. Together with the bound for nearly-embeddable parts, this proves Theorem 20.

6 Adding a modulator

In this section we prove Theorem 2 in a more general setting of Section 3. More precisely, Theorem 2 follows directly from the following theorem via Theorems 16 and 20.

▶ **Theorem 22.** *One can solve in polynomial time a given k-Path instance (G,k), given access to a set $M \subseteq V(G)$ such that $G - M$ admits a tree decomposition of width less than w and adhesion $h = \mathcal{O}(1)$, and an oracle that solves the* Auxiliary Linkage *problem for instances $(G', k', S, (R_i)_{i=1}^r)$ with G' being a subgraph of G, $r, k' \leq k$, $|S| \leq |M| + \mathcal{O}(1)$, and $|V(G')|$ being bounded polynomially in k, w, and $|M|$.*

Here the exponent in the polynomial bound on the size of oracle calls substantially depends on $h = \mathcal{O}(1)$. Therefore, the result of this section is a purely theoretical result classifying the aforementioned parameterization as admitting a polynomial Turing kernel. Let us sketch how, after approximating a tree decomposition of $G - M$, we find a set A with a guard Z of constant

size, to which we can apply the Reduction Rule. For each $u, v \in M$, we mark up to $k+1$ disjoint $u-v$ paths within $G-M$. We mark all their vertices, the bags that contain them, and the lowest-common-ancestor closure of these bags in the decomposition. In total, this still gives polynomially many marked vertices and polynomially many components of the decomposition between marked bags. Using the tree decomposition we can find a medium sized part A of such a component, with a boundary Z contained in at most two adhesions of the decomposition. Now $N(A) \subseteq Z \cup M$, but additionally it is easy to check that any A-traverse of any k-path with endpoints $u, v \in M$ can be replaced with a marked $u-v$ path. Hence Z guards A.

7 Conclusions

We significantly extended the graph classes on which k-Path has a polynomial Turing kernel. In addition, we showed that even an instance that does not belong to such a class, but has a small vertex modulator whose deletion makes it so, can be solved efficiently using small queries to an oracle. A subdivision-based argument (cf. [11]) shows that we cannot generalize much beyond H-topological-minor-free graphs without settling the problem in general. In particular, the existence of a polynomial Turing kernel for graphs of bounded expansion implies its existence in general graphs.

While our narrative focused on k-Path, after small modifications our techniques can also be applied to prove analogues of Theorems 1 and 2 for the k-Cycle problem of detecting a simple cycle of length *at least* k. The main difficulty in adapting our arguments to k-Cycle is the fact that, a priori, the only cycles of length at least k may be arbitrarily much larger than k. However, this issue can easily be resolved in the following way. Since a cycle is contained within a single biconnected component, a Turing kernelization can decompose its input into biconnected components and solve the problem independently in each of them. We then start by testing for the existence of a *path* with k^2 vertices using the algorithms developed in the paper. If there is a path of length k^2 in a biconnected component, then by a classic theorem of Dirac [9] there is a cycle of length at least k, and we are done. If no such path exists, then the longest cycle in G has length less than $2k$, and we can continue under the guarantee that the cycle we are looking for has length at least k and less than $2k$. In this setting, our arguments can be easily adapted. In particular, the absence of a path of length k^2 implies the existence of suitable tree decompositions from which reducible separations can be extracted.

A significant portion of the technical work in this paper was devoted to modifying the graph minors decomposition to obtain the win/win that either answers the problem or finds a reducible separation. In this way, the algorithmic question has driven a challenging graph-theoretic project. It would be interesting to find more problems amenable to such an approach. We conclude with some concrete open problems. Does k-Path have a polynomial Turing kernel on chordal graphs? How about Induced or Directed k-Path, on planar graphs?

References

1 Abhimanyu M. Ambalath, Radheshyam Balasundaram, Chintan Rao H., Venkata Koppula, Neeldhara Misra, Geevarghese Philip, and M. S. Ramanujan. On the kernelization complexity of colorful motifs. In Venkatesh Raman and Saket Saurabh, editors, *Parameterized and Exact Computation - 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, volume 6478 of *Lecture Notes in Computer Science*, pages 14–25. Springer, 2010. doi:10.1007/978-3-642-17493-3_4.

2 Florian Barbero, Christophe Paul, and MichałPilipczuk. Exploring the complexity of layout parameters in tournaments and semi-complete digraphs. In *Proc. 44th ICALP*, 2017. In press.

3 Daniel Binkele-Raible, Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Trans. Algorithms*, 8(4):38:1–38:19, 2012. doi:10.1145/2344422.2344428.

4 Hans L. Bodlaender, Erik D. Demaine, Michael R. Fellows, Jiong Guo, Danny Hermelin, Daniel Lokshtanov, Moritz Müller, Venkatesh Raman, Johan van Rooij, and Frances A. Rosamond. Open problems in parameterized and exact computation - IWPEC 2008. Technical Report UU-CS-2008-017, Utrecht University, 2008.

5 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014.

6 Guantao Chen, Zhicheng Gao, Xingxing Yu, and Wenan Zang. Approximating longest cycles in graphs with bounded degrees. *SIAM J. Comput.*, 36(3):635–656, 2006.

7 Guantao Chen, Xingxing Yu, and Wenan Zang. The circumference of a graph with no $K_{3,t}$-minor, II. *J. Comb. Theory, Ser. B*, 102(6):1211–1240, 2012.

8 Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Minimum bisection is fixed parameter tractable. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 323–332. ACM, 2014. doi:10.1145/2591796.2591852.

9 G. A. Dirac. Some theorems on abstract graphs. *Proceedings of the London Mathematical Society*, s3-2(1):69–81, 1952. doi:10.1112/plms/s3-2.1.69.

10 Jan-Oliver Fröhlich and Theodor Müller. Linear connectivity forces large complete bipartite minors: An alternative approach. *J. Comb. Theory, Ser. B*, 101(6):502–508, 2011. doi:10.1016/j.jctb.2011.02.002.

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

12 Valentin Garnero and Mathias Weller. Parameterized certificate dispersal and its variants. *Theor. Comput. Sci.*, 622:66–78, 2016. doi:10.1016/j.tcs.2016.02.001.

13 Martin Grohe and Dániel Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM J. Comput.*, 44(1):114–159, 2015. doi:10.1137/120892234.

14 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/s00453-014-9910-8.

15 Falk Hüffner, Christian Komusiewicz, and Manuel Sorge. Finding highly connected subgraphs. In *Proc. 41st SOFSEM*, pages 254–265, 2015.

16 Bart M. P. Jansen. Turing kernelization for finding long paths and cycles in restricted graph classes. *J. Comput. Syst. Sci.*, 85:18–37, 2017. doi:10.1016/j.jcss.2016.10.008.

17 Bart M. P. Jansen and Dániel Marx. Characterizing the easy-to-find subgraphs from the viewpoint of polynomial-time algorithms, kernels, and turing kernels. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 616–629. SIAM, 2015. doi:10.1137/1.9781611973730.42.

18 Bart M. P. Jansen, Marcin Pilipczuk, and Marcin Wrochna. Turing kernelization for finding long paths in graph classes excluding a topological minor. *ArXiv e-prints*, 2017. arXiv:1707.01797.

19 Sudeshna Kolay and Fahad Panolan. Parameterized algorithms for deletion to (r, ell)-graphs. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 420–433. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.420.

20 Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Manuscript, 2017.

21 Neil Robertson and Paul D. Seymour. Graph minors: XVII. taming a vortex. *J. Comb. Theory, Ser. B*, 77(1):162–210, 1999. doi:10.1006/jctb.1999.1919.

22 Alexander Schäfer, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, 6(5):883–891, 2012. doi:10.1007/s11590-011-0311-5.

23 Songling Shan. *Homeomorphically Irreducible Spanning Trees, Halin Graphs, and Long Cycles in 3-connected Graphs with Bounded Maximum Degrees*. PhD thesis, Georgia State University, 2015. URL: http://scholarworks.gsu.edu/math_diss/23/.

24 Stéphan Thomassé, Nicolas Trotignon, and Kristina Vuskovic. A polynomial Turing-kernel for weighted independent set in bull-free graphs. In *Proc. 40th WG*, pages 408–419. Springer, 2014.