Decremental Sensitivity Oracles for Covering and Packing Minors

Lawqueen Kanesh
Indian Institute of Technology Jodhpur, India

Fahad Panolan
School of Computing, University of Leeds, UK

M. S. Ramanujan
University of Warwick, UK

Peter Strulo
University of Warwick, UK

Abstract

In this paper, we present the first decremental fixed-parameter sensitivity oracles for a number of basic covering and packing problems on graphs. In particular, we obtain the first decremental sensitivity oracles for Vertex Planarization (delete k vertices to make the graph planar) and Cycle Packing (pack k vertex-disjoint cycles in the given graph). That is, we give a sensitivity oracle that preprocesses the given graph in time $f(k,\ell)n^{\mathcal{O}(1)}$ such that, when given a set of ℓ edge deletions, the data structure decides in time $f(k,\ell)$ whether the updated graph is a positive instance of the problem. These results are obtained as a corollary of our central result, which is the first decremental sensitivity oracle for Topological Minor Deletion (cover all topological minors in the input graph that belong to a specified set, using k vertices).

Though our methodology closely follows the literature, we are able to produce the first explicit bounds on the preprocessing and query times for several problems. We also initiate the study of fixed-parameter sensitivity oracles with so-called structural parameterizations and give sufficient conditions for the existence of fixed-parameter sensitivity oracles where the parameter is just the treewidth of the graph. In contrast, all existing literature on this topic and the aforementioned results in this paper assume a bound on the solution size (a weaker parameter than treewidth for many problems). As corollaries, we obtain decremental sensitivity oracles for well-studied problems such as Vertex Cover and Dominating Set when only the treewidth of the input graph is bounded. A feature of our methodology behind these results is that we are able to obtain query times independent of treewidth.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Dynamic graph algorithms

Keywords and phrases Sensitivity oracles, Data Structures, FPT algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2024.28

Funding *M. S. Ramanujan*: Supported by the Engineering and Physical Sciences Research Council (grant numbers EP/V007793/1 and EP/V044621/1).

1 Introduction

The study of basic graph problems on dynamic inputs has been a central aspect of algorithmics for several decades. A well-studied model in this line of research is the "fault tolerance model". In this model, one assumes that the network at hand is susceptible to a bounded number of faulty network components (i.e., failing nodes or links) at any given time. The goal is to efficiently preprocess the network and produce a sufficiently small data structure so that once the set of faulty nodes or links is given (or equivalently, the corresponding vertices or edges in the graph are deleted), one can recover various properties of the network from the stored data structure without recomputing these from scratch. The fault tolerance model has been a hugely successful setting for various advances on fundamental data structures such as spanners [39] and distance sensitivity oracles [21]. The "dimensions" of interest in such data structures are: the time needed by the preprocessing algorithm, the space complexity of the data structure, the time required to query the data structure in order to recover various properties of the input graph minus the set of failed elements and in some cases, the time required to update the data structure to reflect the failures.

The primary focus of research in the fault tolerance model has been on polynomial-time solvable problems. However, in a recent paper, Bilò et al. [6] extended the fault tolerance model to NP-complete graph problems by introducing a notion of decremental *fixed-parameter sensitivity oracles* (FSO). For an edge (respectively, vertex) decremental sensitivity oracle for a fixed-parameter tractable (FPT) problem Π, the input is an instance (G, k) of Π, where G is an n-vertex input graph and k is the parameter and a number ℓ, and the goal is to develop a preprocessing algorithm $\mathcal{A}$ that builds a data-structure (i.e., the oracle) that, when queried on a set F of at most ℓ edges (respectively, vertices), decides whether $(G - F, k)$ is a positive instance of the problem, using a query algorithm $\mathcal{Q}$. The goal here is to ensure that the preprocessing time is $f(k, \ell)n^{\mathcal{O}(1)}$ and the query time is $g(k, \ell)n^{o(1)}$ for some functions f and g. Unless otherwise specified, one allows both edge and vertex failures. Using this framework, Bilò et al. [6] gave the first edge decremental FSO for several problems including Long Path and Vertex Cover. Subsequently, Alman and Hirsch [3] extended the work of Bilò et al. [6] to also account for edge additions, by introducing a *fully dynamic* notion of sensitivity oracles. Moreover, Alman and Hirsch [3] define a notion of *efficient* sensitivity oracles, where the preprocessing time is $f(k)n^{\mathcal{O}(1)}$, and the query time is $\ell^{\mathcal{O}(1)}g(k)n^{o(1)}$. That is, the dependence on ℓ in both the preprocessing and query time is polynomial. By developing a dynamic variant of the extensor coding method [12], they show that Long Path has a fully dynamic efficient sensitivity oracle even on directed graphs.

These advances made by Bilò et al. [6] and Alman and Hirsch [3] for individual problems pose some natural questions: Could we prove general statements that provide a unified explanation of the existence of fixed-parameter sensitivity oracles (FSOs) for families of problems? Could we obtain *efficient* FSOs for these problems and give explicit bounds on the preprocessing and query times? These questions at the intersection of data structures and parameterized algorithms are our main motivation.

In this paper, we make significant progress towards answering these questions by presenting meta-theorems from which decremental FSOs for a number of basic covering and packing problems on graphs can be derived.

1.1 Our contributions I: FSOs for vertex deletion problems

Many important NP-hard graph optimization problems can be phrased as a vertex deletion problem to a graph class satisfying some property P. Here, the input is a graph G on

n vertices and the task is to find a minimum size vertex subset S such that the graph $G - S$ obtained from G by removing S and its incident edges has the property P. By the well-known result of Lewis and Yannakakis [40] such problems are NP-complete for hereditary properties. For this reason the study of these problems is an integral part of the areas of approximation algorithms, exact-exponential algorithms and parameterized complexity, and has been responsible for the development of many classic algorithmic techniques.

Hence, the family of vertex deletion problems provide a natural candidate for us to develop meta-theorems. This brings us to the Topological Minor Deletion (TM-Deletion) problem which is a vertex deletion problem that directly generalizes numerous well-known problems vertex deletion problems including Vertex Cover, Feedback Vertex Set (delete at most k vertices to obtain a forest) and Vertex Planarization, to name a few. In TM-Deletion, the input is an undirected graph G, a family $\mathcal{F}$ of undirected graphs such that every graph in $\mathcal{F}$ has at most $h(\mathcal{F})$ vertices, and an integer k. The parameter is $k + h(\mathcal{F})$ and the goal is to decide whether there exists $S \subseteq V(G)$ of size at most k such that $G - S$ contains no graph from $\mathcal{F}$ as a topological minor. A graph H is a *topological minor* of G if H can be obtained from G by deleting vertices or edges, and then contracting edges as long as each such edge is incident to at least one vertex of degree precisely 2. The expressive power of TM-Deletion naturally implies that an FSO for this problem would enable us to obtain as a consequence, FSOs for a host of other problems.

▶ **Theorem 1.** Topological Minor Deletion *has a decremental fixed-parameter sensitivity oracle.*

As a consequence of Theorem 1, we obtain decremental FSOs[1] for many well-studied parameterized problems, thus extending the scope of sensitivity oracles for NP-complete graph problems significantly beyond the state of the art. We refer the reader to the appendix for the definitions of problems not defined here and to Section 2 for the formal definition of treewidth and (topological) minors.

▶ **Corollary 2.** *The following problems have FSOs as a consequence of Theorem 1.*

1. Feedback Vertex Set (FVS). *Or more generally,* η-Treewidth Modulator, *i.e., decide whether we can delete at most k vertices from the input graph to obtain a graph with treewidth at most η.*
2. Vertex Planarization. *Or more generally,* Minor Deletion, *i.e., for a set $\mathcal{F}$ of graphs, decide whether we can delete at most k vertices from the input graph to obtain a graph that excludes every graph in $\mathcal{F}$ as a minor.*
3. Cycle Packing. *Or more generally,* Topological Minor Packing, *i.e., for a set $\mathcal{F}$ of graphs, decide whether the input graph contains k vertex disjoint topological-minor models of graphs in $\mathcal{F}$.*
4. Long Path *and* Long Cycle. *That is, decide whether there is a path (or a cycle, respectively) of length at least k in the input graph.*

A useful feature of our proof techniques is that it allows for easy (albeit rough) estimations of the preprocessing and query times of most of the FSOs in the above statement. As a result, without much additional effort, one can prove the following bounds on specific FSOs in Corollary 2.

[1] Since we only deal with the decremental setting in this paper, we drop the explicit reference to this term in the rest of the paper and simply say, FSO.

▶ **Theorem 3.** *The following bounds can be obtained.*[2]

1. Feedback Vertex Set *has an FSO with preprocessing time* $\mathsf{tow}(3, \mathcal{O}((k+\ell)^{11}))n^4$ *and query time* $\mathsf{tow}(2, \mathcal{O}((k+\ell)^{11}))$.
2. Cycle Packing *has an FSO with preprocessing time* $\mathsf{tow}(3, \mathcal{O}((k+\ell)^{20}))n^4$ *and query time* $\mathsf{tow}(2, \mathcal{O}((k+\ell)^{20}))$.
3. Long Path *and* Long Cycle *have FSOs with preprocessing time* $\mathsf{tow}(2, \mathcal{O}(k\log(\ell)))n^4$ *and query time* $\mathsf{tow}(2, \mathcal{O}(k\log(\ell)))$.

Note that these are the first concrete bounds for Cycle Packing. However, the bounds for Long Path implied by our meta-theorem are significantly worse than that of Bilò et al. [6] and Alman and Hirsch [3], which is not surprising since we obtain these bounds by instantiating a general-purpose theorem. For instance, the former get query time upper bounded by $\mathcal{O}(\ell(\ell+k))$ and the latter, $\ell^2 2^k k^{\mathcal{O}(1)}$. However, we prove the bounds in the above theorem in order to illustrate how to use our methodology to obtain explicit bounds for specific problems.

1.2 Our contributions II: A meta-theorem for efficient FSOs

Algorithmic meta-theorems are general algorithmic results applicable to a whole range of problems. Many prominent algorithmic meta-theorems are about model checking; such theorems state that for certain kinds of logic L, and all classes of structures $\mathcal{C}$ that have a certain property, there is an algorithm that takes as input a formula $\phi \in L$ and a structure $S \in \mathcal{C}$ and efficiently determines whether $S \models \phi$. One of the most famous results in this direction is the seminal theorem of Courcelle [16, 14, 15] for model checking of Monadic Second Order Logic (MSO) on graphs of bounded treewidth (see also [1, 5, 11, 17, 22]). Courcelle's theorem (which also extends to a fragment called Counting Monadic Second Order Logic or CMSO) is a crucial component of the parameterized complexity toolbox because numerous well-studied graph problems can be expressed in this particular fragment of logic. Classic examples of CMSO-definable graph properties are Hamiltonicity and 3-Colorability. We refer the reader to Section 2.2 for a formal description of CMSO-definability. Consequently a natural question arises – "Does an analogue of Courcelle's theorem hold in the fault-tolerant setting?" An affirmative answer is implied by existing results in the literature on query testing MSO formulas on bounded-treewidth graphs (see, for instance, Theorem 6.1.3 in [37]). These results build upon Courcelle's approach of reducing model checking MSO sentences on bounded-treewidth graphs to model checking MSO sentences on labelled trees. However, in the quest for *efficient* FSO (recall that we want preprocessing and query time *polynomial* in the number of failures), this approach does not yield a positive outcome since it involves a reduction to MSO model checking on graphs, where the formula size now depends on the number of failures and so, the query algorithm may take time exponential in the number of failures.

In this paper, we prove the following meta-theorem giving a sufficient condition for the existence of *efficient* FSOs with the additional property that the query times are actually independent of input size n (although the definition allows for sublinear dependence on n).

▶ **Theorem 4.** *Every CMSO-definable graph problem has an efficient non-uniform FSO with query time independent of input size, when parameterized by the treewidth of the input graph and size of the CMSO-sentence defining it.*

[2] The notation $\mathsf{tow}(p, q)$ indicates a runtime that is exponential in q, where q is on top of a tower of iterated exponentials of height p.

A *non-uniform* FSO is simply an FSO where one is allowed to have, for every value of the parameter k and number ℓ of permitted failures, a distinct preprocessing algorithm $\mathcal{A}_{k,\ell}$ and query algorithm $\mathcal{Q}_{k,\ell}$.

Theorem 4 forms a crucial component of our proof of Theorem 1. Essentially, we use Theorem 4 to handle "low-treewidth" instances of TM-Deletion. However, notice that Theorem 4 only guarantees a non-uniform FSO whereas Theorem 1 has no such caveat. Hence, a few remarks are in order here. Often, in the literature on non-uniform FPT algorithms, it has been demonstrated that the non-uniformity can be omitted either through self-reducibility arguments or by a case-by-case understanding of the combinatorics behind each problem. We are able to provide such arguments regarding Theorem 4 that essentially suggest that as long as one could solve the CMSO-definable problem under consideration using an explicit dynamic programming algorithm on bounded-treewidth graphs, i.e., the vast majority of natural CMSO-definable problems in the literature, then one can actually infer an FSO for the problem on bounded-treewidth graphs *without* the caveat of non-uniformity. Moreover, this approach can lead to obtaining explicit running time bounds. This is also why we avoid resorting to the aforementioned black-box results on query testing in the literature to handle the low treewidth case. In this paper, we formally exemplify our strategy of eliminating the non-uniformity resulting from the invocation of Theorem 4 for the special case of TM-Deletion, which enables us to prove Theorem 1. Though we only deal with TM-Deletion, our arguments can be easily seen to extend to other problems, which we use to obtain explicit bounds for some of them. We believe that Theorem 4 will be a crucial component of designing FSOs for more problems, especially in conjunction with techniques such as irrelevant vertex removal [25].

In this context, it is important to mention the work of Courcelle and Vanicat [18]. They prove a meta-theorem that implies an efficient FSO for all CMSO-definable problems when parameterized by the treewidth of the input graph and size of the CMSO-sentence defining it. We note that their query times have a logarithmic dependence on n. It is known in the community (although not explicitly published to the best of our knowledge) that the $\log(n)$ dependence can be removed with appropriate preprocessing. However, we believe that the methodology behind Theorem 4 is useful as it enables us to easily obtain concrete bounds in our applications, which does not appear to be straightforward from the result of [18].

1.3 Our contributions III: Edge FSOs parameterized only by treewidth

We demonstrate the further applicability of the proof technique behind Theorem 4 to obtain a meta-theorem that gives sufficient conditions on a problem to have an edge FSO parameterized by the treewidth alone. Notice that all our results and those in the literature up to this point have the solution size as the parameter either explicitly or implicitly. In particular, in Theorem 4 the parameter also includes the size of the MSO formula, which in turn often depends on the solution size in the case of specific problems. Moreover, we highlight the fact that the oracles in this section have query time with a polynomial dependence on ℓ (in fact, only $\mathcal{O}(\ell^2)$). Moreover, the query times are independent of the treewidth. However, they are not *efficient* oracles as the preprocessing algorithm has exponential dependence on ℓ.

We give the following (non-exhaustive) exemplifications of our meta-thereom.

▶ **Theorem 5.** *The following hold.*

1. Vertex Cover *admits an edge FSO parameterized by the treewidth k with preprocessing time $\ell^{\mathcal{O}(2^k)} \cdot n^{\mathcal{O}(1)}$, and query time $\mathcal{O}(\ell^2)$.*

2. Dominating Set *admits an edge FSO parameterized by the treewidth k with preprocessing time $\ell^{\mathcal{O}(3^k)} \cdot n^{\mathcal{O}(1)}$, and query time $\mathcal{O}(\ell^2)$.*

Note that since the treewidth of a graph is at most the size of the minimum vertex cover, the first statement directly implies an edge FSO for Vertex Cover parameterized by solution size.

Here also, the work of Courcelle and Vanicat [18] is relevant as they prove an optimization version of their meta-theorem. However, their query time depends on the treewidth whereas we are able to obtain FSOs with query times independent of the treewidth.

1.4 Related work

Alman and Hirsch [3] note that the work of van den Brand and Saranurak [46] (see full version [47]) on distance sensitivity oracles in combination with standard color-coding techniques also imply a fully dynamic sensitivity oracle for Long Path on directed graphs, but with a worse dependence on k and ℓ. We note that though a $n^{o(1)}$ multiplicative factor in the query time is permitted in the definition of FSO, this is not exploited in their results and similar to our results, the queries of both Alman and Hirsch [3] as well the alternate oracle implied by Brand and Saranurak [46] run in time independent of the input size.

In other recent work, Pilipczuk et al. [44] gave a sensitivity oracle that answers s-t connectivity in constant time if a constant number of vertex failures occur. Interestingly, Pilipczuk et al. [44] show that the techniques they use to obtain their result can be used to design a model checking algorithm for the recently introduced *separator logic* [10] which is more expressive than First Order Logic but less expressive than MSO. This is a promising sign that advances on sensitivity oracles can have a much broader impact beyond the specific problem for which they are developed. We also note that the Arxiv version of [44] contains the tools (MSO query testing on trees) required to prove Kazana's result [37] on MSO query testing on bounded-treewidth graphs.

In recent years, spurred by the first systematic exploration of the intersection of parameterized and dynamic graph algorithms by Alman et al. [4], there has been a significant amount of work combining techniques from these two areas. Of special interest in the context of our paper is the work of Dvorak et al. [23] (improved upon by Chen et al. [13]) and Majewski et al. [42], who gave fully dynamic data structures that are able to maintain CMSO properties. That is, they obtain a data structure that is stronger than just a sensitivity oracle, but at the cost of weaker parameters than the one we use (i.e., treewidth).

Finally, on the topic of intersecting parameterized complexity and fault-tolerant data structures, Lochet et al. [41], in a work preceding the work of Bilò et al. [6], studied fault-tolerant spanners in directed graphs by choosing parameters expressing certain types of structure. Recently, Misra [43] initiated the study of computing fault-tolerant *solutions* (e.g., a solution that remains a feedback vertex set of the graph even if one vertex is removed from the solution) for NP-hard problems, with follow up work by Blazej et al. [7].

2 Preliminaries

2.1 Graphs

Given a graph G, let $V(G)$ and $E(G)$ denote the vertex and edge set of G, respectively. We only deal with simple graphs in this paper. When G is clear from the context, let n and m denote $|V(G)|$ and $|E(G)|$, respectively. For a graph G, $\mathsf{paths}(G)$ denotes the set of all simple paths in G. For a set $A \subseteq V(G)$, we denote by $E(A)$ the set of those edges in G with both endpoints in A.

Formally, the treewidth of a graph is defined as follows.

▶ **Definition 6** **(Tree decomposition).** *A* tree decomposition *of a graph G is a pair (T, β) of a tree T and $\beta : V(T) \to 2^{V(G)}$, such that: (i) $\bigcup_{t \in V(T)} \beta(t) = V(G)$, (ii) for any edge $e \in E(G)$, there exists a node $t \in V(T)$ such that both endpoints of e belong to $\beta(t)$, and (iii) for any vertex $v \in V(G)$, the subgraph of T induced by the set $T_v = \{t \in V(T) : v \in \beta(t)\}$ is a tree. We say a tree decomposition is* nice *if it additionally satisfies the conditions on page 161 of [19]. The* width *of (T, β) is $\max_{v \in V(T)}\{|\beta(v)|\} - 1$. The* treewidth *of G is the minimum width of a tree decomposition of G.*

Let (T, β) be a tree decomposition of a graph G. We refer to the vertices of the tree T as *nodes*. We always assume that T is a rooted tree and so, we have a natural parent-child and ancestor-descendant relationship among nodes in T. The set $\beta(t)$ is called the bag at t. For two nodes $u, t \in T$, we say that u is a *descendant* of t, denoted $u \preceq t$, if t lies on the unique path connecting u to the root. Note that every node is its own descendant. If $u \preceq t$ and $u \neq t$, then we write $u \prec t$. For a tree decomposition (T, β) we also have a mapping $\gamma : V(T) \to 2^{V(G)}$ defined as $\gamma(t) = \bigcup_{u \preceq t} \beta(u)$. For every $t \in V(T)$, we also define $\widehat{\beta(t)} = \beta(t) \cup E(\beta(t))$ and $\widehat{\gamma(t)} = \gamma(t) \cup E(\gamma(t))$. Recall that for a vertex set S, $E(S)$ denotes the set of all edges with both endpoints in S. We call a tree decomposition *nice* if it satisfies the conditions in section 7.2 of [20].

There is an algorithm that, given a graph G on n vertices and an integer w, runs in time $\mathcal{O}(f(w)n^3)$ and either correctly answers that G has treewidth more than w or outputs a tree decomposition of G of optimal width [8].

We next recall the classic notions of minors and topological minors.

▶ **Definition 7** **(Minors).** *A graph H is a* minor *of G if there exists a function $\phi : V(H) \to 2^{V(G)}$ with the following properties: (i) for every $h \in V(H)$, $G[\phi(h)]$ is a connected graph, (ii) for all distinct $h, h' \in V(H)$, $\phi(h) \cap \phi(h') = \emptyset$, and (iii) for all $\{h, h'\} \in E(H)$, there exist $u \in \phi(h)$ and $v \in \phi(h')$ such that $\{u, v\} \in E(G)$. The function ϕ is called a* minor model *of H in G.*

▶ **Definition 8** **(Topological minors).** *Let G and H be two graphs. We say that H is a* topological minor *of G if there exist injective functions $\phi : V(H) \to V(G)$ and $\varphi : E(H) \to$* $\mathsf{paths}(G)$ *such that*

- *for every $e = \{h, h'\} \in E(H)$, the endpoints of $\varphi(e)$ are $\phi(h)$ and $\phi(h')$,*
- *for every distinct $e, e' \in E(H)$, the paths $\varphi(e)$ and $\varphi(e')$ are internally vertex-disjoint,*
- *there does not exist a vertex v in the image of ϕ and an edge $e \in E(H)$ such that v is an internal vertex on $\varphi(e)$.*

We say that (ϕ, φ) is a topological-minor model *of H in G.*

Note that if H is a topological minor of G, then it is also a minor of G. However, the converse does not hold.

Boundaried graphs. Roughly speaking, a boundaried graph is a graph where some vertices are labeled. A formal definition is as follows.

▶ **Definition 9** **(Boundaried graph).** *A* boundaried graph *is a graph G with a set $\partial(G) \subseteq V(G)$ of distinguished vertices called* boundary vertices*, and an injective labeling $\lambda_G : \partial(G) \to \mathbb{N}$. The set $\partial(G)$ is the* boundary *of G, and the* label set *of G is $\Lambda(G) = \{\lambda_G(v) \mid v \in \partial(G)\}$.*

Given a finite set $I \subseteq \mathbb{N}$, $\mathcal{G}_I$ denotes the class of all boundaried graphs whose label set is I, and $\mathcal{G}_{\subseteq I} = \bigcup_{I' \subseteq I} \mathcal{G}_{I'}$. A boundaried graph in $\mathcal{G}_{\subseteq [t]}$ is called a *t-boundaried* graph. Note that if G is a boundaried graph and $x \in V(G)$ is a vertex in the boundary, then $G - x$ is a boundaried graph that inherits its boundary and labeling from G in the natural way. That is, we simply remove x and preserve the labeling of the remaining vertices.

2.2 Counting Monadic Second Order Logic

The syntax of Monadic Second Order Logic (MSO) of graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, variables for vertices, edges, sets of vertices and sets of edges, the quantifiers $\forall$ and $\exists$, which can be applied to these variables, and the binary relations: (i) $u \in U$, where u is a vertex variable and U is a vertex set variable; (ii) $d \in D$, where d is an edge variable and D is an edge set variable; (iii) $\mathbf{inc}(d, u)$, where d is an edge variable, u is a vertex variable, and the interpretation is that the edge d is incident to u; (iv) equality of variables representing vertices, edges, vertex sets and edge sets.

An MSO sentence is an MSO formula without free variables. *Counting Monadic Second Order Logic* (CMSO) extends MSO by including atomic sentences testing whether the cardinality of a set is equal to q modulo r, where q and r are integers such that $0 \leq q < r$ and $r \geq 2$. That is, CMSO is MSO with the following atomic sentence: $\mathbf{card}_{q,r}(S) = \mathbf{true}$ if and only if $|S| \equiv q \pmod{r}$, where S is a set. We refer to [5, 16, 15] for a detailed introduction to CMSO. We note that what we refer to as CMSO in this paper is sometimes called CMSO$_2$ in the literature to indicate that quantifying over edge sets is permitted.

▶ **Definition 10 (Property).** *A* property *is a function σ from the set of all graphs to* {true, false}*. For a CMSO sentence ψ, the property σ_ψ is defined as follows. Given a graph G, $\sigma_\psi(G)$ equals* true *if and only if $G \models \psi$.*

▶ **Definition 11 (CMSO-definable property).** *A property σ is* CMSO-definable *if there exists a CMSO sentence ψ such that $\sigma = \sigma_\psi$. In this case, we say that ψ* defines *σ.*

We next recall an implication of the classic Courcelle's Theorem [16, 14, 15] proof (see also [17]). This fact, which is a central component in the proof of Theorem 4, says that a certain canonical equivalence relation over boundaried graphs has finite index. We first need to identify precisely those pairs of graphs that could potentially be related by the canonical equivalence and so, we define the *compatibility* equivalence relation $\equiv_c$ on boundaried graphs as follows. We write $G_\alpha \equiv_c G_\beta$ and say that G_α is *compatible* with G_β if $\Lambda(G_\alpha) = \Lambda(G_\beta)$. Now, we define the *canonical equivalence relation* $\equiv_\sigma$ on boundaried graphs.

▶ **Definition 12 (Canonical equivalence).** *Given a property σ of graphs, the* canonical equivalence relation $\equiv_\sigma$ *on boundaried graphs is defined as follows. For two boundaried graphs G_α and G_β, we say that $G_\alpha \equiv_\sigma G_\beta$ if* **(i)** *$G_\alpha \equiv_c G_\beta$, and* **(ii)** *for every boundaried graphs G_γ compatible with G_α (and thus also with G_β), we have: $\sigma(G_\alpha \oplus G_\gamma) = \mathsf{true} \Leftrightarrow \sigma(G_\beta \oplus G_\gamma) = \mathsf{true}$.*

Here, the gluing operator $\oplus$ identifies equally-labeled vertices of the two boundaried graphs.

A property σ of graphs has *finite state* if $\forall I \subseteq \mathbb{N}$, the set of equivalence classes of $\equiv_\sigma$ when restricted to $\mathcal{G}_{\subseteq I}$ is finite. Given a CMSO sentence ψ, the canonical equivalence relation associated with ψ is $\equiv_{\sigma_\psi}$, and for simplicity, we denote this relation by $\equiv_\psi$.

We are now ready to state the required consequence of Courcelle's Theorem (see, for example, [16, 14, 15, 9]).

▶ **Proposition 13.** *Every CMSO-definable property on graphs has finite state.*

We use the RAM model (see Harel and Tarjan [29]) with addition and uniform cost measure. Each word holds $\mathcal{O}(\log n)$ bits and each basic operation on a word is assumed to take constant time.

3 Technical overview

In this section, we give an overview of our techniques, omitting details where necessary due to space constraints.

Since we effectively use Theorem 4 in our proof of Theorem 1, we first describe our proof of this result and how it implies a (uniform) FSO for TM-Deletion parameterized by the treewidth of the input graph in addition to the standard parameterization comprising the deletion set size k and the size of the largest graph in the family $\mathcal{F}$ that we want to exclude as topological minors.

3.1 The FSO for CMSO-definable problems on bounded-treewidth graphs

We first give an overview of our proof of Theorem 4. For every CMSO formula ψ, one can define a canonical equivalence relation (see Definition 12) over the set of all boundaried graphs. A boundaried graph is simply a graph where some vertices are called boundary vertices and are assigned labels from a finite label set. A graph is t-boundaried if the label set has size at most t. The Myhill-Nerode equivalence for ψ on boundaried graphs says that if we take two t-boundaried graphs G_1 and G_2 then they are equivalent if and only if their boundary labels are equal and moreover, for any t-boundaried graph whose boundary labels are the same as those of G_1 (and hence also of G_2), if we glue the graphs G_1 and H along the boundaries by identifying equally labeled vertices and similarly, if we glue the graphs G_2 and H along the boundaries in this way, either both resulting graphs model ψ or neither one does. It is known (e.g., from the proof of Courcelle's theorem itself) that for every CMSO formula ψ and every t, the number of equivalence classes induced by this relation over t-boundaried graphs is a function of ψ and t alone (a property called finite state). This implies that there is an $r \in \mathbb{N}$ that depends only on ψ and t (in our context, t will be 1 plus the treewidth of the graph), such that both the number of equivalence classes and the length of the encoding of a smallest boundaried graph in each equivalence class is upper bounded by r. If ψ and t are fixed, then r is constant. Now, suppose that $\mathcal{R}$ is the set comprising a smallest boundaried graph from each of the equivalence classes (having fixed ψ and t) and suppose we know $\mathcal{R}$.

Our key insight is the following. Suppose we take a nice tree decomposition of the input graph and pick a bag. Now, consider the boundaried graph obtained by taking the graph induced by those vertices that appear in this bag or below it and making the vertices in the bag the boundary. Then, we observe that (a) this boundaried graph is equivalent to one of the graphs in $\mathcal{R}$ and (b) regardless of the element failures in this boundaried graph, the resulting graph will still be equivalent to one of the graphs in $\mathcal{R}$. The only catch here is that before the query is given, one cannot know the representative of the "future" equivalence class. Hence, our preprocessing strategy aims to keep track, for each boundaried subgraph of the input graph obtained in the way we described above, *all* possible canonical equivalence classes that this graph can fall into, upon the removal of the failed vertices or edges given by the query in future. Our querying strategy, on the other hand, is a dynamic-programming algorithm. Depending on the at-most-ℓ queried edges and vertices, we identify a set of $\mathcal{O}(\ell)$

boundaried graphs that we have preprocessed and by examining the possible different ways in which the equivalence classes of only these specific boundaried graphs can be impacted by the failures, we are able to produce a correct answer to the query. This gives us Theorem 4. The non-uniformity comes from the assumption of knowing $\mathcal{R}$.

To overcome the non-uniformity aspect while applying Theorem 4, one must avoid the requirement of knowing $\mathcal{R}$. Instead, it is sufficient if, for every bounded-treewidth boundaried graph, we could efficiently compute an equivalent (under the Myhill-Nerode equivalence) boundaried graph whose size is bounded by some computable function of ψ and treewidth.

This approach, which was first introduced in order to obtain constructive versions of meta-algorithmic results on kernelization [26], has proved useful in several other instances in the literature [24, 25]. We show that this approach is indeed applicable to TM-Deletion and in fact our arguments suggest that it is generally applicable as long as one has an explicit dynamic programming algorithm on bounded-treewidth graphs. Hence, we are able to obtain an FSO for TM-Deletion parameterized by the deletion set size k, the treewidth of the input graph, and the size of the largest graph in the family $\mathcal{F}$ that we want to exclude as topological minors.

The same idea is also used to obtain the bounds in Theorem 3. That is, we essentially reuse the non-uniform oracles given by the proof of Theorem 4 (while avoiding the only source of non-uniformity) to handle low-treewidth instances of these specific problems. We then use a win-win argument to extend to FSOs with explicit bounds. In the win-win argument, we use the fact that if the treewidth of the input graph is already high, we get a trivial oracle that always answers either yes or no depending on the problem.

As an illustration, we describe the proof of the consequence for Cycle Packing in Corollary 2. That is, Cycle Packing has an FSO parameterized by the solution size k. Towards this, we first show that there is an FSO for this problem parameterized by k and treewidth. Let us assume that the treewidth bound is the max of treewidth and k, and let it be w. Then, we show that there is an FSO for this problem parameterized by w with preprocessing time $\mathsf{tow}(3, \mathcal{O}(w^2)) \cdot n^4$ and query time $\mathsf{tow}(2, \mathcal{O}(w^2)) + \ell^{\mathcal{O}(1)}$. This is done by showing that the size of the set $\mathcal{R}$ and maximum size of a graph in $\mathcal{R}$ are computable and then using Theorem 4. Intuitively, the bound on the size of the set $\mathcal{R}$ comes from the size of the table at a bag when performing the standard dynamic programming over tree decompositions. The same fact can be used to also obtain a bound on the size of the graphs in $\mathcal{R}$. That is, if the subgraph "rooted" at a bag is larger than some computable function of w, then one can find a pair of ancestor-descendant bags that have the same dynamic programming tables, implying that these two graphs are equivalent. Now, a standard replacement operation of "cutting" the subgraph below the ancestor and "pasting" the subgraph below the descendant gives a strictly smaller equivalent graph and this process can be repeated.

Now, consider a general graph G. If G has treewidth greater than $s(k, \ell)$ for some function s (from grid-minor theorem), we can conclude that G has a sufficiently large grid, implying sufficiently many vertex-disjoint cycles. One can argue that after removing at most ℓ vertices or edges we will still have at least k vertex-disjoint cycles and hence we will have a positive instance of Cycle Packing. So we simply output an oracle that always answers yes to any query. If G has a smaller treewidth, then we can apply the above argument for low treewidth graphs, giving us a preprocessing time of $\mathsf{tow}(3, \mathcal{O}((k+\ell)^{\mathcal{O}(1)}))$ and a query time of $\mathsf{tow}(2, \mathcal{O}((k+\ell)^{\mathcal{O}(1)}))$.

3.2 The FSO for Topological Minor Deletion in general graphs

We now give an overview of the proof of Theorem 1 assuming that we have an FSO for TM-Deletion parameterized by k, treewidth of the input graph and the size of the largest graph in the family $\mathcal{F}$ that we want to exclude as topological minors. Recall that, here the input is $(G, k, \mathcal{F})$ and an integer ℓ, and we want our preprocessing algorithm to essentially encode the answers to the input instances $(G - F, k, \mathcal{F})$ of TM-Deletion for any $F \subseteq V(G) \cup E(G)$ of size at most ℓ such that these answers can be retrieved efficiently by the query algorithm. Towards the preprocessing of $(G, k, \mathcal{F})$ we use the results about "irrelevant vertices" for the TM-Deletion problem by Fomin et al. [25]. As these irrelevant vertex results are for the vertex deletion problem, as a first step we construct an "equivalent" instance $(G', k, \mathcal{F}')$ for the FSO, where all the edge failures in the original instance can be replaced with appropriate vertex failures in the new instance. This allows to restrict ourselves to handling vertex failures alone. More formally, for each $H \in \mathcal{F}$, we define H' to be the graph obtained from H by adding three pendant (i.e., degree-1) vertices adjacent to each vertex v of H. Then, we set $\mathcal{F}' = \{H' : H \in \mathcal{F}\}$. The graph G' is constructed from G as follows. First we subdivide each edge in G. For each $e \in E(G)$, let u_e be the subdivision vertex in G' corresponding to e. Then for each vertex $v \in V(G)$ we add three pendant vertices adjacent to v. Then, each edge e failing in G can be thought of as a vertex u_e failing in G'. We prove that it is enough to give a vertex FSO for the instance $(G', k, \mathcal{F}')$. Then, the preprocessing algorithm $\mathcal{A}$ uses the template of Fomin et al. [25] which in turn was built on the approach introduced by Robertson and Seymour for Disjoint Paths [45]. Their approach has found applications in many significant results in the area [28, 30, 34, 31, 33, 36, 35, 32, 27]. In the template of Fomin et al. [25], we have three exhaustive cases.

Case 1. The treewidth of the input graph G is upper bounded by a function of k, ℓ, and $\mathcal{F}'$. In this case we use our fault-tolerance oracle for TM-Deletion parameterized by k, $\mathcal{F}'$ and the treewidth which we have outlined in the previous subsection.

Case 2. The input graph G has a clique minor whose size is lower bounded by a computable function of k, ℓ, and $\mathcal{F}'$. In this case Fomin et al. [25] gave an algorithm to find an irrelevant vertex v with respect to "any" vertex deletion set of size $k + \ell$. That is, for any vertex subset $S \subseteq V(G')$, the topological minors of size at most δ in $G - S$ and $G - S - v$ are same. Here, we set δ to be the maximum size of a graph in $\mathcal{F}'$.

Case 3. Case 1 and 2 are not applicable. In this case the "weak structure theorem" [45] implies that the graph G contains a "large flat wall". Here, large means that its size is lower bounded by a function of k, ℓ, and $\mathcal{F}'$. In this case as well Fomin et al. [25] gave an algorithm to find an irrelevant vertex v with respect to "any" vertex deletion set of size $k + \ell$.

In our preprocessing algorithm, as long as Case 2 or Case 3 is applicable, we delete the irrelevant vertices computed and finally we end up with a graph with bounded treewidth, which places us in Case 1, which we have described how to handle.

Corollaries of Theorem 1.

We now argue that the assertions made in Corollary 2 hold.

▶ **Proposition 14** ([2, 38]). *For every $\eta \in \mathbb{N}$, there is a set $\mathcal{F}_\eta$ of graphs such that a graph has treewidth at most η if and only if it excludes the graphs in $\mathcal{F}_\eta$ as a minor.*

We also require the following simple fact.

▶ **Proposition 15.** *For every family $\mathcal{F}$ of graphs, there is a set $\mathcal{F}'$ of graphs such that any graph contains a minor model of a graph from $\mathcal{F}$ if and only if it contains a topological-minor model of a graph from $\mathcal{F}'$.*

We note that the families $\mathcal{F}_\eta$ in Proposition 14 and $\mathcal{F}'$ in Proposition 15 are constructive. In combination with these two propositions, Theorem 1 implies the first two statements of Corollary 2. That is, η-TREEWIDTH MODULATOR is precisely TM-DELETION where the family $\mathcal{F}'$ of forbidden topological minors is obtained by first using η as "input" to Proposition 14 to obtain $\mathcal{F}_\eta$, and then plugging in $\mathcal{F}_\eta$ as input to Proposition 15 to obtain the required family $\mathcal{F}'$. Similarly, MINOR DELETION where $\mathcal{F}$ is the family of graphs to exclude as minors can be written as TM-DELETION, where the forbidden family of topological minors is obtained by plugging $\mathcal{F}$ in to Proposition 15. We now proceed to the remaining statements of Corollary 2.

Let us now consider the dual problem to TM-DELETION, i.e., TOPOLOGICAL MINOR PACKING (TM-PACKING), which is formally defined as follows.

TOPOLOGICAL MINOR PACKING (TM-PACKING)

Input:	An undirected graph G, a family of undirected graphs $\mathcal{F}$ such that every graph in $\mathcal{F}$ has at most h^* vertices, and an integer k.
Parameter:	$k + h^*$.
Problem:	Does there exist k vertex-disjoint topological-minor models of graphs in $\mathcal{F}$?

▶ **Theorem 16.** TM-PACKING *has an FSO.*

Proof. Let $(G, \mathcal{F}, k, \ell)$ be the input of TM-PACKING. Now we define a family $\mathcal{F}'$ as follows.

$$\mathcal{F}' = \{H \mid \exists H_1, \ldots, H_k \in \mathcal{F} : H \text{ is the disjoint union of } H_1, \ldots, H_k\}$$

Then, notice that any subgraph of G contains k vertex-disjoint topological-minor models from $\mathcal{F}$ if and only if the same subgraph of G contains a graph from $\mathcal{F}'$ as topological minor. Moreover, $|\mathcal{F}'| \le |\mathcal{F}|^k$ and the largest graph in $\mathcal{F}'$ has size at most $k \cdot h(\mathcal{F})$.

Now, we are ready to give an FSO $(\mathcal{A}, \mathcal{Q})$ for TM-PACKING by using an FSO $(\mathcal{A}', \mathcal{Q}')$ for TM-DELETION (Theorem 1) as a subroutine.

The preprocessing algorithm $\mathcal{A}$: Let the input to $\mathcal{A}$ be an instance $(G, \mathcal{F}, k)$ of TM-PACKING and $\ell \in \mathbb{N}_0$.

Step 1: Construct the family $\mathcal{F}'$ defined above.

Step 2: Run the algorithm $\mathcal{A}'$ on input $(G, \mathcal{F}', 0)$ and ℓ and return its output.

This completes the description of the preprocessing algorithm.

The query algorithm $\mathcal{Q}$: Let the input be $F \subseteq V(G) \cup E(G)$.

Step 1: Use the query algorithm $\mathcal{Q}'$ to decide whether $(G - F, \mathcal{F}', 0)$ is a yes-instance of TM-DELETION.

Step 2: If $\mathcal{Q}'$ answers YES, then $\mathcal{Q}$ answers NO. Else, $\mathcal{Q}$ answers YES.

This completes the description of the query algorithm.

Notice that the family $\mathcal{F}'$ can be computed in time bounded by a function of $h(\mathcal{F}) + k$. Moreover, since $(\mathcal{A}', \mathcal{Q}')$ is an FSO for TM-DELETION, it follows that $\mathcal{A}'$ is an FPT algorithm parameterized by $h(\mathcal{F}')+k \leq k\cdot(h(\mathcal{F})+1)$. Since these are the only two steps in $\mathcal{A}$, it follows that $\mathcal{A}$ is also an FPT algorithm parameterized by $h(\mathcal{F})+k$. Similarly, since $\mathcal{Q}'$ runs in time $f'(h(\mathcal{F})+k)$ for some computable function f', it follows that $\mathcal{Q}$ runs in time $f(h(\mathcal{F}')+k)$ for some computable function f. Finally, the correctness of the pair $(\mathcal{A},\mathcal{Q})$ follows from the fact that for every $F \subseteq V(G)\cup E(G)$, the instance $(G-F,\mathcal{F},k)$ of TM-Packing is a yes-instance if and only if the instance $(G-F,\mathcal{F}',0)$ of TM-Deletion is a no-instance. ◀

Finally, notice that Long Path and Long Cycle are both special cases of TM-packing. Hence, the final statement of Corollary 2 can also be obtained as a consequence of Theorem 1.

3.3 Edge FSOs parameterized by treewidth alone

We employ the same high-level approach as that used for Theorem 4 (see Section 3.1). However, instead of considering equivalence among boundaried graphs, we aim to identify equivalent sets of failure edges for the boundaried graph obtained at each bag. This idea can be summarized as follows. First, assume that the problem satisfies certain properties that are typically satisfied by problems for which there is an explicit dynamic programming algorithm over a given tree decomposition. This includes well-known problems such as Vertex Cover and Dominating Set. Further, suppose that when any set of ℓ edges are deleted below a bag B and we were to re-run the dynamic programming algorithm on the graph induced by the vertices in this bag and its descendants, then the dynamic programming table computed at the bag B is only a "small" perturbation of the dynamic programming table that was initially computed at this bag. In our context, "small" simply means that the entries in the cells change (either increase or decrease) by at most ℓ. For instance, in the case of Vertex Cover, deleting ℓ edges will not change any of the partial solutions by more than ℓ. Now, since the size of the bag is bounded by the treewidth, this implies a bounded number of equivalence classes for the set of all edge failures of size at most ℓ. This fact is then used to keep a representative of each equivalence class and also a solution corresponding to them. Finally, we show how the query algorithm can identify the equivalence class of the given query efficiently.

Formally, we prove the result below. The terms in the theorem statement build upon the notation of Garnero et al. [26] and are explained in the subsequent paragraphs.

▶ **Theorem 17.** *Consider a subset problem Π. If Π has an encoder ξ that admits a gap function g, a gap signature computation algorithm, an FPT exact algorithm, and Π is DP effective, then Π admits an edge FSO parameterized by the treewidth.*

Here, a parameterized graph problem Π is called a *subset problem* if there exists a language L_Π associated with Π that comprises of pairs (G,S) where, for every graph G and $S \subseteq V(G)$, $(G,S)\in L_\Pi$ if and only if S is a solution to Π on G.

The following definitions encapsulate the idea of the standard dynamic programming algorithms on treewidth. The boundary of G will be a bag of the tree decomposition so the vertices of the boundary will be the only way that some solution S can "interact" with the rest of the graph. $\mathcal{C}(|\Lambda(G)|)$ represents the space of possible interactions and S is compatible with some encoding R from this space if it does in fact interact as specified by R.

▶ **Definition 18** (Encoder)**.** *Consider a subset problem Π. An encoder ξ of Π is a pair $(\mathcal{C},L_{\mathcal{C}})$, where:*

1. *$\mathcal{C}:\mathbb{N}_0\to 2^{\Sigma^*}$ is a computable function, , with $\mathcal{C}(0)=\varepsilon$. Here, Σ is some finite alphabet depending on Π.*

2. *$L_{\mathcal{C}}$ is a language that comprises of triples (G, S, R), where G is a boundaried graph, $S \subseteq V(G)$, and $R \in \mathcal{C}(|\Lambda(G)|)$. If $(G, S, R) \in L_{\mathcal{C}}$, then we say that S is* compatible *with R under ξ.*
3. *For every 0-boundaried graph G and $S \subseteq V(G)$, the triple $(G, S, \varepsilon) \in L_{\mathcal{C}}$ if and only if $(G, S) \in L_{\Pi}$.*

For example, an encoder of Vertex Cover sets $\mathcal{C}(k)$ to the power set of $[k]$ representing subsets of the boundary and $(G, S, R) \in L_{\mathcal{C}}$ iff S is a vertex cover of G and $S \cap \partial(G) \supseteq R$. That is S is compatible with R when S "agrees" with R on the boundary. Thus, the encoder describes the space that the dynamic programming is over.

We now define the function that such a dynamic programming algorithm would calculate.

▶ **Definition 19** (Family of nice functions associated to an encoder). *Consider a subset problem Π with an encoder $\xi = (\mathcal{C}, L_{\mathcal{C}})$. For a boundaried graph G, we define a* nice function *$\eta^{\xi}_{G} : \mathcal{C}(|\Lambda(G)|) \to \mathbb{N}_0$ as follows: For every $R \in \mathcal{C}(|\Lambda(G)|)$,*

$$\eta^{\xi}_{G}(R) = \begin{cases} |V(G)| + 1 & \textit{if } \{S : (G, S, R) \in L_{\mathcal{C}}\} = \emptyset \\ \min\{|S| : (G, S, R) \in L_{\mathcal{C}}\} & \textit{otherwise.} \end{cases}$$

Note that since $\min\{|S| : (G, S, \varepsilon) \in L_{\mathcal{C}}\} = \min\{|S| : (G, S) \in L_{\Pi}\}$, this means that for all $p \in \mathbb{N}$, $\eta^{\xi}_{G}(\varepsilon) \leq p$ iff $(G, p) \in \Pi$.

Continuing our Vertex Cover example, if ξ is the encoder described above, then $\eta^{\xi}_{G}(R)$ is the size of the minimum vertex cover of G that contains every vertex from R. In the standard dynamic programming algorithm for Vertex Cover parameterized by treewidth, the table entry indexed by some $x \in V(T)$ and $R \subset \beta(x)$ is exactly $\eta^{\xi}_{G^{\downarrow}_{x}}(R)$.

We now wish to bound how much the answer can change due to edge failures. For example, in Vertex Cover the size of the solution will never decrease by more than the number of edge failures.

▶ **Definition 20** (Gap function for encoder). *Consider a subset problem Π with an encoder $\xi = (\mathcal{C}, L_{\mathcal{C}})$. We say that ξ admits a* gap function *$g : \mathbb{N} \to \mathbb{N}$, if for every boundaried graph G, for every $F \subseteq E(G)$, and for every $R \in \mathcal{C}(|\Lambda(G)|)$, we have*

$$|\eta^{\xi}_{G}(R) - \eta^{\xi}_{G-F}(R)| \leq g(|F|).$$

From now onwards we will consider a subset problem Π and assume it has an encoder ξ that admits a gap function g.

▶ **Definition 21** (Gap signature). *For all boundaried graphs G, and sets $F \subseteq E(G)$, the* gap signature *of F in G is the function $\sigma^{F}_{G} : \mathcal{C}(|\Lambda(G)|) \to \{-g(|F|), \dots g(|F|)\}$, defined as follows: for every $R \in \mathcal{C}(|\Lambda(G)|)$,*

$$\sigma^{F}_{G}(R) = \eta^{\xi}_{G}(R) - \eta^{\xi}_{G-F}(R).$$

Bounding the amount the answer changes allows us to keep track of exactly how much the answer changes for each failure set with the gap signature. However, notice that the number of possible gap signatures for any set $F \subseteq E(G)$ is $(2g(|F|) + 1)^{|\mathcal{C}(|\Lambda(G)|)|}$, that is it only depends on the size of boundary of G and the size of F. Being able to calculate the gap signature will be a key part of our algorithm.

▶ **Definition 22** (Gap signature computation algorithm). *A* gap signature computation algorithm *takes as an input (i) a graph G, (ii) a nice tree decomposition of G, (T, β) with width k, (iii) a node $x \in V(T)$, and (iv) a set of edges, $F \subseteq E(G^{\downarrow}_{x,T})$ of size ℓ, runs in time $f_{gs}(k, \ell) n^{\mathcal{O}(1)}$ and outputs $\sigma^{F}_{G^{\downarrow}_{x,T}}$, where f_{gs} is a computable function.*

Since we are mostly interested in the gap signature of a given failure set we introduce the following equivalence relation.

▶ **Definition 23** (Equivalence relation $\equiv_G$). *For all boundaried graphs G, and sets $F_1, F_2 \subseteq E(G)$, we say that $F_1 \equiv_G F_2$ if and only if all of the following statements hold.*
1. $\sigma_G^{F_1} = \sigma_G^{F_2}$*. That is, for every $R \in \mathcal{C}(|\Lambda(G)|)$, $\sigma_G^{F_1}(R) = \sigma_G^{F_2}(R)$.*
2. $F_1 \cap \binom{\partial(G)}{2} = F_2 \cap \binom{\partial(G)}{2}$*, that is, F_1 and F_2 coincide on the set of edges with both endpoints in $\partial(G)$.*
3. $|F_1| = |F_2|$.

Since there are only a small number of possible gap signatures, there are also not many equivalence classes. More precisely, if the sets F_1 and F_2 are of size at most ℓ then there are at most $\mathcal{O}(\ell|\partial(G)|^2)$ equivalence classes for each gap signature. We would like to exploit this by pre-calculating solutions on one element from each equivalence class. To this end we define a set consisting of exactly one element from each equivalence class.

In the following, for a set X and $\ell \in \mathbb{N}$, by $\mathcal{P}_\ell(X)$, we denote the set of all the subsets of X of size at most ℓ. Given a nice tree decomposition (T, β), for all $(u, v) \in E(G)$ let $\mathsf{Highest}((u, v))$ be the unique highest node t in T such that $\beta(t)$ contains both u and v. For each $x \in V(T)$, we define $\chi(x) = \{e \in E(G) : x = \mathsf{Highest}(e)\}$ and $\chi^{\downarrow}(x) = \cup_{y \prec x} \chi(y)$. Note that $\chi(x) \subseteq E(G[\beta(x)])$ and $\chi^{\downarrow}(x) \subseteq E(G_x^{\downarrow})$. Also χ is a partition of $E(G)$, that is $\chi(x) \cap \chi(y) = \emptyset$ for all $x \neq y$ and $\chi^{\downarrow}(\mathsf{root}(T)) = \cup_{x \in V(T)} \chi(x) = E(G)$.

▶ **Definition 24** (Type representative family). *Consider a graph G, a nice tree decomposition (T, β) of G, and $\ell \in \mathbb{N}$. For every node $x \in V(T)$, we define a* type representative family *$\Re(x) \subseteq \mathcal{P}_\ell(\chi^{\downarrow}(x))$ such that:*
1. *For every $F_1, F_2 \in \Re(x)$, if $F_1 \equiv_{G_x^{\downarrow}} F_2$ then $F_1 = F_2$, and*
2. *For every $F \in \mathcal{P}_\ell(\chi^{\downarrow}(x))$, there exists $F' \in \Re(x)$ such that $F' \equiv_{G_x^{\downarrow}} F$.*

Note that since $\chi^{\downarrow}(\mathsf{root}(T)) = E(G^{\downarrow}_{\mathsf{root}(T)}) = E(G)$, $\Re(\mathsf{root}(T))$ contains a representative from each equivalence class of failure sets. Also note that if $F_1 \subseteq \chi(x)$ then, for all $F_2 \in \mathcal{P}_\ell(\chi^{\downarrow}(x))$, we have $F_1 \equiv_{G_x^{\downarrow}} F_2$ if and only if $F_1 = F_2$ so $\mathcal{P}_\ell(\chi(x)) \subseteq \Re(x)$. Let

$$f_R(k, \ell) = \ell \cdot k^2 \cdot (2g(\ell) + 1)^{|\mathcal{C}(k)|}$$

then the size of $\Re(x)$ is at most $\mathcal{O}(f_R(k, \ell))$ where k is the width of (T, β). However there are $\mathcal{O}(n^\ell)$ possible failure sets that could be in $\Re(x)$ which is too many to calculate a representative from each equivalence class by brute force. So we will split the failure set into smaller subproblems and use the following property to calculate the representatives by dynamic programming. Intuitively this says that, when doing dynamic programming on the treewidth, the equivalence 'carries through', that is, if we have some representatives of the equivalence classes of the failure set below y_1 and y_2 rather than the actual failure set, we can use these to construct a representative of the whole failure set below x.

▶ **Definition 25** (DP effective). *We say that our subset problem Π is* DP effective *if, for every graph G, and nice tree decomposition (T, β) of G, the following holds: For every $x, y_1, y_2 \in V(T)$ such that x is a common ancestor of y_1 and y_2, and for every $F_x \subseteq \chi(x)$, $F_1, F_1^* \subseteq \chi^{\downarrow}(y_1)$, and $F_2, F_2^* \subseteq \chi^{\downarrow}(y_2)$, it holds that if $F_1^* \equiv_{G_{y_1}^{\downarrow}} F_1$ and $F_2^* \equiv_{G_{y_2}^{\downarrow}} F_2$, then $(F_x \cup F_1 \cup F_2) \equiv_{G_x^{\downarrow}} (F_x \cup F_1^* \cup F_2^*)$.*

From now onwards we will work on a given graph G, and failure set size ℓ, and assume we know a nice tree decomposition (T, β) of G of width at most k and size $\mathcal{O}(kn) = \mathcal{O}(n^2)$. The following lemma shows that we can calculate a type representative family for each node quickly. Recall that $f_R(k,\ell)$ is a bound on the size of any such family and that $f_{gs}(k,\ell)$ is the superpolynomial component of the runtime of the gap signature computation algorithm. The proof can be found in the appended full version.

▶ **Lemma 26.** *If Π admits a gap signature computation algorithm and is DP effective, then we can compute a type representative family $\Re(x)$ for all $x \in V(T)$ in time $2^{k^2} \cdot f_R(k,\ell)^3 \cdot f_{gs}(k,\ell) \cdot n^{\mathcal{O}(1)}$.*

We now have, at each node $x \in V(T)$, a small set of possible failure sets that together cover every equivalence class. The only remaining obstacle is that we cannot calculate which of these edge sets is equivalent to our actual failure set at query time since the gap signature computation algorithm is too slow. To this end we define the following tables. Together with DP effectiveness they will allow us to calculate a representative of the true failure set from the bottom up.

▶ **Definition 27.** *Suppose $\Re(x)$ is a type representative family for all $x \in V(T)$. Let x, y_1, $y_2 \in V(T)$ such that x is an ancestor of y_1 and y_2 and $x \neq y_1$. Then, for every set $F_x \subseteq \chi(x)$, $F_1 \in \Re(y_1)$, and $F_2 \in \Re(y_2)$ we define the following.*

1. *$H_1[x, y_1, F_x, F_1] = Q_x$, where $F_x \cup F_1 \equiv_{G_x^\downarrow} Q_x$ and $Q_x \in \Re(x)$.*
2. *If $y_1 \neq y_2$ and x is the lowest common ancestor of y_1 and y_2, then define $H_2[x, y_1, y_2, F_x, F_1, F_2] = Q_x$, where $F_x \cup F_1 \cup F_2 \equiv_{G_x^\downarrow} Q_x$ and $Q_x \in \Re(x)$.*

▶ **Lemma 28.** *If Π admits a gap signature computation function and is DP effective, then the tables H_1 and H_2 described in Definition 27 can be filled in time $2^{k^2} \cdot f_R(k,\ell)^3 \cdot f_{gs}(k,\ell) \cdot n^{\mathcal{O}(1)}$. Moreover, H_1 and H_2 both have size at most $2^{k^2} \cdot f_R(k,\ell)^2 \cdot n^{\mathcal{O}(1)}$.*

Finally we will need to use an exact algorithm for the problem on the original graph as a black box. This is a very weak assumption since an FSO is itself an exact algorithm (simply call it with $F = \emptyset$) so we expect an exact algorithm to be easier to obtain than an FSO.

▶ **Definition 29.** *An* FPT exact algorithm *takes as input a graph G and a nice tree decomposition of G, with width k, runs in time $f_{opt}(k)n^{\mathcal{O}(1)}$ and outputs $\eta_G^\xi(\varepsilon)$.*

4 Concluding remarks

Parameterized sensitivity oracles provide a fertile middle ground of study between static FPT algorithms (where many problems are well-understood) and dynamic FPT algorithms (where many problems turn out to be hard) and deserve a thorough exploration. Along with the work of Bilò et al. [6], Alman and Hirsch [3] and Pilipczuk et al. [44], this paper furthers our understanding of the capabilities of state-of-the-art algorithm design techniques used in parameterized complexity. Indeed, Alman and Hirsch [3] in their paper ask whether there examples of techniques other than extensor coding, that are used to solve static versions of parameterized problems and which can be used to design faster dynamic algorithms or sensitivity oracles. Our three main results (Theorem 4, Theorem 1 and Theorem 17) provide useful classification tools to study other problems in this framework and importantly, gives a road map for obtaining explicit bounds. The possibility of obtaining similar classification results in the *fully dynamic* setting is a natural direction for future work.

Acknowledgements.

We thank anonymous reviewers for the pointers to [18, 37].

References

1 Karl Abrahamson and Michael Fellows. Finite automata, bounded treewidth and well-quasiordering. In *Graph structure theory (Seattle, WA, 1991)*, volume 147 of *Contemp. Math.*, pages 539–563, Providence, RI, 1993. Amer. Math. Soc. URL: http://dx.doi.org/10.1090/conm/147/01199, doi:10.1090/conm/147/01199.

2 Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms (SODA 2008)*, pages 641–650. SIAM, 2008. URL: http://portal.acm.org/citation.cfm?id=1347082.1347153.

3 Josh Alman and Dean Hirsch. Parameterized sensitivity oracles and dynamic algorithms using exterior algebras. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 9:1–9:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://doi.org/10.4230/LIPIcs.ICALP.2022.9, doi:10.4230/LIPIcs.ICALP.2022.9.

4 Josh Alman, Matthias Mnich, and Virginia Vassilevska Williams. Dynamic parameterized problems and algorithms. *ACM Trans. Algorithms*, 16(4):45:1–45:46, 2020. URL: https://doi.org/10.1145/3395037, doi:10.1145/3395037.

5 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

6 Davide Bilò, Katrin Casel, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, J. A. Gregor Lagodzinski, Martin Schirneck, and Simon Wietheger. Fixed-parameter sensitivity oracles. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 23:1–23:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: https://doi.org/10.4230/LIPIcs.ITCS.2022.23, doi:10.4230/LIPIcs.ITCS.2022.23.

7 Václav Blazej, Pratibha Choudhary, Dusan Knop, Jan Matyás Kristan, Ondrej Suchý, and Tomás Valla. Constant factor approximation for tracking paths and fault tolerant feedback vertex set. In Jochen Könemann and Britta Peis, editors, *Approximation and Online Algorithms - 19th International Workshop, WAOA 2021, Lisbon, Portugal, September 6-10, 2021, Revised Selected Papers*, volume 12982 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2021. URL: https://doi.org/10.1007/978-3-030-92702-8_2, doi:10.1007/978-3-030-92702-8_2.

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

9 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (meta) kernelization. *J. ACM*, 63(5):44:1–44:69, 2016. URL: http://dl.acm.org/citation.cfm?id=2973749.

10 Mikolaj Bojanczyk. Separator logic and star-free expressions for graphs. *CoRR*, abs/2107.13953, 2021. URL: https://arxiv.org/abs/2107.13953, arXiv:2107.13953.

11 Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7:555–581, 1992.

12 Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164. ACM, 2018. URL: https://doi.org/10.1145/3188745.3188902, doi:10.1145/3188745.3188902.

13 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. Efficient fully dynamic elimination forests with applications to detecting long paths and cycles. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13,*

2021, pages 796–809. SIAM, 2021. URL: `https://doi.org/10.1137/1.9781611976465.50`, `doi:10.1137/1.9781611976465.50`.

14 B. Courcelle. The monadic second-order logic of graphs. III. Tree-decompositions, minors and complexity issues. *RAIRO Inform. Théor. Appl.*, 26(3):257–286, 1992.

15 B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of graph grammars and computing by graph transformation, Vol. 1*, pages 313–400. World Sci. Publ, River Edge, NJ, 1997. URL: `http://dx.doi.org/10.1142/9789812384720_0005`, `doi:10.1142/9789812384720_0005`.

16 Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Inform. and Comput.*, 85:12–75, 1990.

17 Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach.* Cambridge University Press, 2012.

18 Bruno Courcelle and R. Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discret. Appl. Math.*, 131(1):129–150, 2003. URL: `https://doi.org/10.1016/S0166-218X(02)00421-3`, `doi:10.1016/S0166-218X(02)00421-3`.

19 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015. URL: `https://doi.org/10.1007/978-3-319-21275-3`, `doi:10.1007/978-3-319-21275-3`.

20 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

21 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.

22 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013.

23 Zdenek Dvorák, Martin Kupec, and Vojtech Tuma. A dynamic data structure for MSO properties in graphs with bounded tree-depth. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 2014. URL: `https://doi.org/10.1007/978-3-662-44777-2_28`, `doi:10.1007/978-3-662-44777-2\_28`.

24 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, M. S. Ramanujan, and Saket Saurabh. Solving d-sat via backdoors to small treewidth. 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 630–641. SIAM, 2015. URL: `https://doi.org/10.1137/1.9781611973730.43`, `doi:10.1137/1.9781611973730.43`.

25 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Hitting topological minors is FPT. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1317–1326. ACM, 2020. URL: `https://doi.org/10.1145/3357713.3384318`, `doi:10.1145/3357713.3384318`.

26 Valentin Garnero, Christophe Paul, Ignasi Sau, and Dimitrios M. Thilikos. Explicit linear kernels via dynamic programming. *SIAM J. Discret. Math.*, 29(4):1864–1894, 2015. URL: `https://doi.org/10.1137/140968975`, `doi:10.1137/140968975`.

27 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 479–488, 2011. URL: `http://doi.acm.org/10.1145/1993636.1993700`, `doi:10.1145/1993636.1993700`.

28 Martin Grohe, Ken-ichi Kawarabayashi, and Bruce A. Reed. A simple algorithm for the graph minor decomposition - logic meets structural graph theory. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New*

Orleans, Louisiana, USA, January 6-8, 2013, pages 414–431. SIAM, 2013. URL: https://doi.org/10.1137/1.9781611973105.30, doi:10.1137/1.9781611973105.30.

29 Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.*, 13(2):338–355, 1984. URL: https://doi.org/10.1137/0213024, doi:10.1137/0213024.

30 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1802–1811, 2014.

31 Naonori Kakimura and Ken-ichi Kawarabayashi. Fixed-parameter tractability for subset feedback set problems with parity constraints. *Theor. Comput. Sci.*, 576:61–76, 2015. URL: https://doi.org/10.1016/j.tcs.2015.02.004, doi:10.1016/j.tcs.2015.02.004.

32 Ken-ichi Kawarabayashi. Planarity allowing few error vertices in linear time. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 639–648. IEEE Computer Society, 2009. URL: https://doi.org/10.1109/FOCS.2009.45, doi:10.1109/FOCS.2009.45.

33 Ken-ichi Kawarabayashi and Yusuke Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. *J. Comb. Theory, Ser. B*, 102(4):1020–1034, 2012. URL: https://doi.org/10.1016/j.jctb.2011.12.001, doi:10.1016/j.jctb.2011.12.001.

34 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012. URL: https://doi.org/10.1016/j.jctb.2011.07.004, doi:10.1016/j.jctb.2011.07.004.

35 Ken-ichi Kawarabayashi and Bruce A. Reed. An (almost) linear time algorithm for odd cyles transversal. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 365–378. SIAM, 2010. URL: https://doi.org/10.1137/1.9781611973075.31, doi:10.1137/1.9781611973075.31.

36 Ken-ichi Kawarabayashi, Bruce A. Reed, and Paul Wollan. The graph minor algorithm with parity conditions. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 27–36. IEEE Computer Society, 2011. URL: https://doi.org/10.1109/FOCS.2011.52, doi:10.1109/FOCS.2011.52.

37 Wojciech Kazana. *Query evaluation with constant delay. (L'évaluation de requêtes avec un délai constant)*. PhD thesis, École normale supérieure de Cachan, Paris, France, 2013. URL: https://tel.archives-ouvertes.fr/tel-00919786.

38 Jens Lagergren. Upper bounds on the size of obstructions and intertwines. *J. Comb. Theory, Ser. B*, 73(1):7–40, 1998. URL: https://doi.org/10.1006/jctb.1997.1788, doi:10.1006/jctb.1997.1788.

39 Christos Levcopoulos, Giri Narasimhan, and Michiel Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 186–195, New York, NY, USA, 1998. Association for Computing Machinery. URL: https://doi.org/10.1145/276698.276734, doi:10.1145/276698.276734.

40 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. URL: https://doi.org/10.1016/0022-0000(80)90060-4, doi:10.1016/0022-0000(80)90060-4.

41 William Lochet, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Fault tolerant subgraphs with applications in kernelization. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 47:1–47:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. URL: https://doi.org/10.4230/LIPIcs.ITCS.2020.47, doi:10.4230/LIPIcs.ITCS.2020.47.

42 Konrad Majewski, Michal Pilipczuk, and Marek Sokolowski. Maintaining cmso properties on dynamic structures with bounded feedback vertex number. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 46:1–46:13. Schloss Dagstuhl - Leibniz-Zentrum f"ur Informatik, 2023. URL: `https://doi.org/10.4230/LIPIcs.STACS.2023.46`, `doi:10.4230/LIPIcs.STACS.2023.46`.

43 Pranabendu Misra. On fault tolerant feedback vertex set. *CoRR*, abs/2009.06063, 2020. URL: `https://arxiv.org/abs/2009.06063`, `arXiv:2009.06063`.

44 Michal Pilipczuk, Nicole Schirrmacher, Sebastian Siebertz, Szymon Torunczyk, and Alexandre Vigny. Algorithms and data structures for first-order logic with connectivity under vertex failures. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 102:1–102:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: `https://doi.org/10.4230/LIPIcs.ICALP.2022.102`, `doi:10.4230/LIPIcs.ICALP.2022.102`.

45 Neil Robertson and Paul D. Seymour. Graph minors .XIII. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. URL: `https://doi.org/10.1006/jctb.1995.1006`, `doi:10.1006/jctb.1995.1006`.

46 Jan van den Brand and Thatchaphol Saranurak. Sensitive distance and reachability oracles for large batch updates. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 424–435. IEEE Computer Society, 2019. URL: `https://doi.org/10.1109/FOCS.2019.00034`, `doi:10.1109/FOCS.2019.00034`.

47 Jan van den Brand and Thatchaphol Saranurak. Sensitive distance and reachability oracles for large batch updates. *CoRR*, abs/1907.07982, 2019. URL: `http://arxiv.org/abs/1907.07982`, `arXiv:1907.07982`.