Fixed-Parameter Tractability of Graph Isomorphism in Graphs with an Excluded Minor

Daniel Lokshtanov[*]
daniello@ucsb.edu
University of California, Santa Barbara
Santa Barbara, USA

Marcin Pilipczuk[†]
marcin.pilipczuk@mimuw.edu.pl
Institute of Informatics, University of Warsaw
Warsaw, Poland

Michał Pilipczuk[†]
michal.pilipczuk@mimuw.edu.pl
Institute of Informatics, University of Warsaw
Warsaw, Poland

Saket Saurabh[†‡]
saket@imsc.res.in
Institute of Mathematical Sciences
Chennai, India
Department of Informatics, University of Bergen
Bergen, Norway

ABSTRACT

We prove that Graph Isomorphism and Canonization in graphs excluding a fixed graph H as a minor can be solved by an algorithm working in time $f(H) \cdot n^{O(1)}$, where f is some function. In other words, we show that these problems are fixed-parameter tractable when parameterized by the size of the excluded minor, with the caveat that the bound on the running time is not necessarily computable. The underlying approach is based on decomposing the graph in a canonical way into *unbreakable* (intuitively, well-connected) parts, which essentially provides a reduction to the case where the given H-minor-free graph is unbreakable itself. This is complemented by an analysis of unbreakable H-minor-free graphs, which reveals that every such graph can be canonically decomposed into a part that admits few automorphisms and a part that has bounded treewidth.

CCS CONCEPTS

• **Theory of computation → Fixed parameter tractability**.

KEYWORDS

graph isomorphism, fixed-parameter tractability, excluded minor

ACM Reference Format:
Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2022. Fixed-Parameter Tractability of Graph Isomorphism in Graphs with an Excluded Minor. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3519935.3520076

1 INTRODUCTION

The Graph Isomorphism problem is arguably the most widely known problem whose membership in P is unknown, but which is not believed to be NP-hard. After decades of research, a quasi-polynomial time algorithm was proposed by Babai in 2015 [1].

While the existence of a polynomial-time algorithm on general graphs is still elusive, the complexity of Graph Isomorphism has been well understood on several classes of graphs, where structural properties of graphs in question have been used to design polynomial-time procedures solving the problem. Classic results in this area include an $n^{O(d)}$-time algorithm on graphs of maximum degree d [2, 29], a polynomial-time algorithm for planar graphs [20–22, 37], an $n^{O(g)}$-time algorithm on graphs of Euler genus g [11, 30], an $O(n^{k+4.5})$-time algorithm for graphs of treewidth k [3], an $n^{O(k)}$-time algorithm for graphs of rankwidth k [15, 18], an $n^{f(|H|)}$-time algorithm for graphs excluding a fixed graph H as a minor [34], and an $n^{f(|H|)}$-time algorithm for graphs excluding a fixed graph H as a topological minor [13] (where f is some computable function).

In all the results mentioned above, the degree of the polynomial bound on the running time depends on the parameter — maximum degree, genus, treewidth, rankwidth, or the size of the excluded (topological) minor — in at least a linear fashion. Since the parameter can be as high as linear in the size of the graph, for large values of the parameter the running time bound of the quasi-polynomial-time algorithm of Babai [1], which works on general graph, is preferable. During the last few years, there has been several successful attempts of bridging this gap by using the group-theoretic approach of Babai in conjunction with structural insight about considered graph classes. This led to algorithms with running time of the form $n^{\mathrm{polylog}(p)}$, where p is any of the following parameters: maximum degree [16], Euler genus [31], treewidth [38], and the size of a fixed graph H excluded as a minor [19]. We refer to a recent survey [14] for an excellent exposition.

A parallel line of research is to turn the aforementioned algorithms into fixed-parameter algorithms for the parameters in question. That is, instead of a running time bound of the form $n^{f(p)}$ for

[*]Supported by BSF award 2018302 and NSF award CCF-2008838.
[†]This research is a part of projects that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme Grant Agreement 714704 (Marcin Pilipczuk), 667651 and 948057 (Michał Pilipczuk), and 819416 (Saket Saurabh).
[‡]Supported by the Swarnajayanti Fellowship (No. DST/SJF/MSA01/2017-18).

a computable function f and a parameter p, we would like to have an algorithm with running time bound $f(p) \cdot n^c$ for a universal constant c. In other words, the degree of the polynomial governing the running time bound should be independent of the parameter; only the leading multiplicative factor may depend on it.

In this line of research, the current authors developed an FPT algorithm for Graph Isomorphism parameterized by treewidth [26]. This result has been subsequently improved and simplified [17], as well as used to give a slice-wise logspace algorithm [10]. In 2015, Kawarabayashi [23] announced an FPT algorithm for Graph Isomorphism parameterized by the Euler genus of the input graph, with a linear dependency of the running time on the input size. Very recently, Neuen [32] proposed a different and simpler algorithm for this case, which runs in time $2^{O(g^4 \log g)} \cdot n^c$, for some constant c. The recent survey [14] mentions obtaining FPT algorithms with parameterizations by the size of an excluded minor, maximum degree, and the size of an excluded topological minor as important open problems. (Note that the last parameter, the size of an excluded topological minor, generalizes the other two.)

Our Contribution. In this work we essentially solve the first of these open problems by proving the following statement.

Theorem 1.1. *There exists an algorithm that given a graph H and an H-minor-free graph G, works in time $f(H) \cdot n^{O(1)}$ for some function f and outputs a canonical labeling of G.*

Here, a *canonical labeling* of G is a labeling of the vertices of G with labels from $[|V(G)|] = \{1, \ldots, |V(G)|\}$ so that for any two isomorphic graphs G and G', the mapping matching vertices with equal labels in G and G' is an isomorphism between G and G'. Thus, our algorithm solves the more general Canonization problems, while Graph Isomorphism can be solved by computing the canonical labelings for both input graphs and comparing the obtained labeled graphs.

The caveat in Theorem 1.1 is that we are not able to guarantee that f is computable. This makes the algorithm formally fall outside of the usual definitions of fixed-parameter tractability (see e.g. [5]), but we still allow ourselves to call it an FPT algorithm for Graph Isomorphism and Canonization parameterized by the size of an excluded minor. We stress that we obtain a single algorithm that takes H on input, and not a different algorithm for every fixed H.

We now describe the ideas standing behind the proof of Theorem 1.1. First, we need to take a closer look at the FPT algorithm for Graph Isomorphism and Canonization for graphs of bounded treewidth [17, 26]. There, the goal is to obtain an *isomorphism-invariant* tree decomposition of the input graph of width bounded in parameter; then, to test isomorphism of two graphs it suffices to test isomorphism of such decompositions. Unfortunately, it is actually impossible to find one such isomorphism-invariant tree decomposition of the input graph; for instance, in a long cycle one needs to arbitrarily break symmetry at some moment. However, up to technical details, it suffices and is possible to find a small isomorphism-invariant family of tree decompositions. To achieve this goal, the algorithm of [26] heavily relies on the existing understanding of parameterized algorithms for approximating the treewidth of a graph.

The initial idea behind our approach is to borrow more tools from the literature on parameterized graph separation problems, in particular from the work on the technique of *recursive understanding* [4, 6, 8, 24]. This work culminated in the following decomposition theorem for graphs, proved by a superset of authors.

Theorem 1.2 ([6]). *Given a graph G and an integer k, one can in time $2^{O(k \log k)} n^{O(1)}$ compute a tree decomposition of G where every adhesion is of size at most k and for every bag S the following holds: for every separation (A, B) in G of order at most k, either $|A \cap S| \leqslant |A \cap B|$ or $|B \cap S| \leqslant |A \cap B|$.*

The last property of Theorem 1.2 says that a bag S of the tree decomposition cannot be broken by a separation of order at most k into two parts that both contain a large portion of S. This property is often called *unbreakability*. More formally, let us recall the notion of a (q, k)-unbreakable set introduced in [4, 8]. Given a graph G and integers $q \geqslant k \geqslant 0$, we say that a set $S \subseteq V(G)$ is (q, k)-*unbreakable* if for every separation (A, B) in G of order at most k, we have $|A \cap S| \leqslant q$ or $|B \cap S| \leqslant q$. Theorem 1.2 of [6] can be seen as a simpler and cleaner version of an analogous result of [8], where the bags are only guaranteed to be (q, k)-unbreakable for some q bounded exponentially in k, and adhesion sizes are also bounded only exponentially in k.

Theorem 1.2 and its predecessor from [8] have been used to design parameterized algorithms for several graph separation problems [6, 8], most notably for Minimum Bisection. In most cases, when looking for a deletion set of size at most k, one performs dynamic programming on the tree decomposition provided by Theorem 1.2, where every step handling a single bag can be solved using color-coding thanks to the unbreakability of the bag. More recent applications include a parameterized approximation scheme for Min k-Cut [28], as well as algorithms and data structures for problems definable in first-order logic with connectivity predicates [33].

In this paper we propose to use the ideas behind the tree decomposition of Theorem 1.2 in the context of Graph Isomorphism and Canonization in order to provide a reduction to the case when the input graph is suitably unbreakable. The next statement is a wishful-thinking theorem that in some variant we were able to prove, but in the end the result turned out to be too cumbersome to use. In essence, it says the following: in FPT time one can compute a tree decomposition with the same qualitative properties as the one provided by Theorem 1.2, but in an isomorphism-invariant way.

"Wishful-thinking Statement" 1.3. *There is a computable function f and an algorithm that, given a graph G and an integer k, in time $f(k)n^{O(1)}$ computes an isomorphism-invariant tree decomposition of G such that*

- *the sizes of adhesions are bounded by $f(k)$; and*
- *every bag is either of size at most $f(k)$ or is $(f(k), k)$-unbreakable.*

Instead of proving and using (a formal and correct version of) "Theorem" 1.3, we resort to the strategy used in the earlier works, namely recursive understanding. Here, in some sense we compute a variant of the tree decomposition of "Theorem" 1.3 on the fly, shrinking the already processed part of the graph into constant-size representatives. Importantly, the run of this process is isomorphism invariant. An overview of this approach is provided in Section 2.1, while the full statement and its proof will be presented in the full version of the paper.

At first glance, "Theorem" 1.3 may seem unrelated to the setting of graphs excluding a fixed minor. However, let us recall one of the main results of the Graph Minor Theory, namely the Structure Theorem for graphs excluding a fixed minor, proved by Robertson and Seymour in [35]. Informally speaking, this statement says that if a graph G excludes a fixed graph H as a minor, then G admits a tree decomposition (called henceforth the *RS-decomposition*) in which the sizes of adhesions are bounded in terms of H and the torso of every bag is nearly embeddable in a surface of Euler genus bounded in terms of H. Without going into the precise definition of "nearly", let us make the following observation: if we are given an H-minor-free graph G and we apply to G the algorithm of Theorem 1.2 with k equal to the expected bound on adhesion sizes in an RS-decomposition, then the resulting tree decomposition should roughly resemble the RS-decomposition. In particular, one may expect that the "large" bags of the decomposition of Theorem 1.2 should also be nearly embeddable in some sense, as (due to unbreakability) they cannot be partitioned much finer in an RS-decomposition whose adhesions are of size at most k.

For the Graph Isomorphism and Canonization problems, the main issue now is that the output of Theorem 1.2 is not necessarily isomorphism-invariant; this is where the wishful-thinking "Theorem" 1.3 comes into play. In particular, one could expect that the "large" bags of the decomposition of "Theorem" 1.3 would be nearly embeddable in some sense. We are able to carry this intuition through the recursive understanding point of view, and show the following.

Theorem 1.4 (simplified variant). *For every graph H there exists a function $q_H : \mathbb{N} \to \mathbb{N}$ such that the following holds. Suppose there exists an integer k and a canonization algorithm $\mathcal{A}$ that works on all graphs that are H-minor-free and $(q_H(i), i)$-unbreakable for all $i \leqslant k$. Then there is also a canonization algorithm $\mathcal{B}$ for all H-minor-free graphs. Furthermore, $\mathcal{B}$ can be chosen to run in time upper bounded by $g(H) \cdot n^{O(1)}$ plus the total time taken by at most $g(H) \cdot n^{O(1)}$ invocations of $\mathcal{A}$ on H-minor-free graphs with no more than n vertices, where g is some function.*

The proof of Theorem 1.4 is sketched in Section 2.1. The full version will be proved in the full version of the paper. In essence, the full version differs from the one above in that it is uniform in H: if there is a single algorithm $\mathcal{A}$ that works for all H, then there is one resulting algorithm $\mathcal{B}$ that works for all H.

With Theorem 1.4 understood, it suffices to "only" provide a Canonization algorithm working on graphs that are H-minor-free and $(q(i), i)$-unbreakable for all $i \leqslant k$. We comment here on the quantifier order: the algorithm needs to accept any unbreakability guarantee q, and then we have to choose the threshold k to which this guarantee will be used based on H and q. However, both k and the values of q (up to $q(k)$) can be included as parameters in the final running time bound. That said, we prove the following statement.

Theorem 1.5 (simplified variant). *For every graph H and function $q : \mathbb{N} \to \mathbb{N}$ there exists a constant k and an algorithm that given an H-minor-free graph G with a promise that G is $(q(i), i)$-unbreakable for all $i \leqslant k$, outputs a canonical labeling of G in time $f(\sum_{i=1}^{k} q(i)) \cdot n^{O(1)}$, where f is a computable function.*

Note that Theorem 1.5 combined with Theorem 1.4 yield a non-uniform version of Theorem 1.1. Again, the full version of Theorem 1.5 is an appropriately uniform statement that together with Theorem 1.4 proves Theorem 1.1 in full generality.

Let us now discuss the proof of Theorem 1.5. Observe that if we set k higher than the bound on adhesion sizes in an RS-decomposition for H-minor-free graphs, then we expect any RS-decomposition of a $(q(k), k)$-unbreakable H-minor-free graph G to have one central bag (with a nearly embeddable torso) that may be huge, while all other bags have sizes at most $q(k)$. Thus, it seems natural that a procedure for Canonization on nearly embeddable graphs can be lifted to a procedure working on graphs as above.

The main step towards the proof of Theorem 1.5 is provided by the following statement.

Theorem 1.6 (simplified version). *For every graph H and function $q : \mathbb{N} \to \mathbb{N}$ there exists a constant k, computable functions $f_{\mathrm{time}}, f_{\mathrm{tw}}, f_{\mathrm{genus}} : \mathbb{N} \to \mathbb{N}$, and an algorithm that given a graph G that is H-minor-free and $(q(i), i)$-unbreakable for all $i \leqslant k$, works in time bounded by $f_{\mathrm{time}}(\sum_{i=1}^{k} q(i)) \cdot n^{O(1)}$ and computes a partition of $V(G) = V_{\mathrm{tw}} \uplus V_{\mathrm{genus}}$ and a nonempty family $\mathcal{F}_{\mathrm{genus}}$ of bijections $V_{\mathrm{genus}} \mapsto [|V_{\mathrm{genus}}|]$ such that*

(1) the partition $V(G) = V_{\mathrm{tw}} \uplus V_{\mathrm{genus}}$ is isomorphism-invariant;
(2) $G[V_{\mathrm{tw}}]$ has treewidth bounded by $f_{\mathrm{tw}}(\sum_{i=1}^{k} q(i))$;
(3) $|\mathcal{F}_{\mathrm{genus}}| \leqslant f_{\mathrm{genus}}(\sum_{i=1}^{k} q(i)) \cdot n^{O(1)}$; and
(4) $\mathcal{F}_{\mathrm{genus}}$ is isomorphism-invariant.

The statement of Theorem 1.6 is quite technical, so let us provide more intuitive explanation. We are given an H-minor-free graph G with sufficiently strong unbreakability properties. Then the claim is that G can be vertex-partitioned in an isomorphism-invariant way into two parts V_{tw} and V_{genus}. The part V_{genus} is *rigid* in the sense that the subgraph induced by it admits a polynomially-sized isomorphism-invariant family of labelings. The other part V_{tw} may have multiple automorphisms, but the subgraph induced by it has bounded treewidth. Informally speaking, the bounded-treewidth part V_{tw} always contains all the "near-" elements of a near-embedding: vortices, apices, etc., while the rigid part V_{genus} contains the core of the embedded part; its rigidity is witnessed by the embedding.

Theorem 1.5 follows quite easily from Theorem 1.6 combined with the FPT canonization procedure on graphs of bounded treewidth [26]. So the main weight of argumentation is contained in the proof of Theorem 1.6. This proof is sketched in Section 2.2 and will be proven formally in the full version.

Finally, let us remark that the full version of Theorem 1.4 is stated and proved in terms of graph classes defined by forbidding *topological minors*. Thus, it can be readily used also to reduce the Canonization problem on H-topological-minor-free graphs to the same problem on suitably unbreakable H-topological-minor-free graphs. For the latter setting, one could apply the Structure Theorem of Grohe and Marx [13] to reason that unbreakable H-topological-minor-free graphs are either close to being nearly embeddable (and this case is treated in this paper), or they essentially have bounded maximum degree. In spirit, this reduces the problem of finding an FPT algorithm for Canonization on H-topological-minor-free graphs to the problem of finding such an algorithm on graphs of bounded maximum degree. We refrain from expanding this discussion in this paper, as it is tangential to the main direction of our work.

2 OVERVIEW

2.1 Decomposition

In this section we provide an informal overview of the proof of Theorem 1.4. For convenience, we restate Theorem 1.4 here.

Theorem 1.4 (restated, simplified variant of Theorem 1.4) *For every graph H there exists a function $q_H : \mathbb{N} \to \mathbb{N}$ such that the following holds. Suppose there exists an integer k and a canonization algorithm $\mathcal{A}$ that works on graphs that are H-minor-free and $(q_H(i), i)$-unbreakable for all $i \leqslant k$. Then there is also a canonization algorithm $\mathcal{B}$ for H-minor-free graphs. Furthermore, $\mathcal{B}$ runs in time upper bounded by $g(H) \cdot n^{O(1)}$ plus the total time taken by at most $g(H) \cdot n^{O(1)}$ invocations of $\mathcal{A}$ on H-minor-free graphs with no more than n vertices, where g is some function.*

The proof of Theorem 1.4 is based on the *recursive understanding* technique, pioneered in [12, 24], and explicitly defined and further refined in [4].

We first discuss the essence of the recursive understanding technique, as it has been applied in the past, in the context of some generic problem. Much of this discussion does *not* apply to Canonization, but it helps motivate our approach and naturally introduces the crucial notions of representatives and replacement.

Recursive understanding based algorithms proceed as follows. Either the input graph is already (q, k)-unbreakable, then we are already done, because the goal is to reduce the original problem to the problem on unbreakable graphs. Otherwise there exists a separation (A, B) of order at most k such that both A and B have size q, which is much bigger than k. The algorithm calls itself recursively on $G[A]$, and after thoroughly analyzing the graph $G[A]$, produces a *representative* G_A^R for it.

The representative G_A^R is a tiny graph on at most $f(k) < q$ vertices, and all of the vertices of G_A^R are new vertices that are not present in G, except that G_A^R shares the vertices $D = A \cap B$ with G (we will require that $G_A^R[D] = G[D]$.) The algorithm then *replaces* $G[A]$ in G with G_A^R. This means that the algorithm removes all vertices of $A \setminus B$ from G, and attaches G_A^R to D instead. We call the resulting graph $G^\star$. Note that $|V(G^\star)| \leqslant |B| + q$. The algorithm then calls itself recursively on $G^\star$, and finally lifts the solution on $G^\star$ to a solution of G.

We upper bound the running time of the algorithm by a function $T(n, k)$ of the number of vertices and the parameter k - the size of the separators. $T(n, k)$ satisfies the following recurrence.

$$T(n, k) \leqslant S(n, k) + T(|A|, k) + T(|B| + f(k), k) + L(n, k) \quad (1)$$

Here, $S(n, k)$ denotes the time to find the separation (A, B) or decide that such a separation does not exist, $T(|A|, k)$ is the running time of the algorithm on $G[A]$, $T(|B| + f(k), k)$ is the running time of the algorithm on $G^\star$, and $L(n, k)$ is the time it takes to lift a solution to $G^\star$ back to a solution of G. A simple recurrence analysis of Equation 1 shows that $T(n, k) \leqslant (S(n, k) + L(n, k)) \cdot g(k) \cdot n$. Thus, as long as we are able to find a separator in FPT time, lift solutions of $G^\star$ back to G in FPT time, and solve the base case of (q, k) unbreakable graphs in FPT time, we get an FPT algorithm for the original problem.

Being able to execute the above scheme rests on a crucial property of the relationship between the graph $G[A]$ and the representative $G[A]^R$ that we replace it with. In particular a solution to the reduced graph $G^\star$ needs to be useful for the lifting procedure in order to recover the solution to G. This is tricky, because the algorithm needs to compute G_A^R from $G[A]$ without looking at $G[B]$. In particular this means that the reduction from G_A to G_A^R has to work for *every* possible $G[B]$. This is the reason for the name "*recursive understanding*" - the procedure needs to "understand" $G[A]$ to such an extent that it can efficiently lift a solution of $G^\star$ to a solution of G, pretty much independently of what $G^\star$ is (since only k vertices of $G[A]$ come from $G^\star$). This typically means that our algorithm should not just solve the original problem, but a generalized version of the problem that takes as input the graph $G[A]$, together with the set $D \subseteq A$, and the promise that "the rest of the graph" will attach to $G[A]$ (and G_A^R) only via D. The task is to compute sufficient information about $(G[A], D)$ to be able to produce G_A^R.

While this task may appear much more difficult than the original problem, for many computational problems the "recursive understanding" task is no harder than the original problem in the formal sense that an FPT algorithm for the original problem implies the existence of an FPT algorithm for recursive understanding [27]. However no such results were previously known for Graph Isomorphism or Canonization, and it was far from obvious that such a statement would even be true for these problems.

The first main hurdle in applying recursive understanding for Canonization is that we need to be able to find a separation (A, B) of order at most k and $\min\{|A|, |B|\} > q$ in an *isomorphism invariant* way. Here, by isomorphism invariant we mean that if two graphs G and $\hat{G}$ are isomorphic, then if we run the algorithm on G and find a separation (A, B) of G, and on $\hat{G}$, and find a separation $(\hat{A}, \hat{B})$ of $\hat{G}$ then *every* isomorphism from G to $\hat{G}$ maps A to $\hat{A}$ and B to $\hat{B}$. We have no idea whatsoever how to find a single separation (A, B) of G of order at most k with $\min\{|A|, |B|\} \geqslant q$ (or determine that such a separation does not exist) in an isomorphism invariant way in FPT time. Indeed, the task of finding such a separation looks as difficult as canonizing G.

For this reason we turn to an easier problem, finding a single set $B \subseteq V(G)$ in an isomorphism invariant way such that (i) B is either small (that is has size at most $f(H)$), or it has some nice properties - for example B could be (q, k)-unbreakable in G (for some q and k upper bounded by a function of H), and (ii) every connected component C of $G - B$ has at most k neighbors in B.

It is convenient to formalize the decomposition of G into B and the remaining components in terms of tree decompositions. We follow the notation of [13] for tree decompositions. We say that a *star decomposition* is a tree decomposition (T, χ) where T is a star rooted at its center node b (the unique node of degree larger than 1 in T). The *bags* of the decomposition are the sets $\{\chi(v) : v \in V(T)\}$ and the *adhesions* are the sets $\{\sigma(\ell) = \chi(\ell) \cap \chi(b) : \ell \in V(T) \setminus \{b\}\}$.

We will frequently need to find an isomorphism invariant star decomposition (T, χ) of G with center bag b such that B has some nice properties, and all adhesions have size at most k. Here isomorphism invariant means that if G and $\hat{G}$ are isomorphic with star decompositions (T, χ) and $(\hat{T}, \hat{\chi})$ respectively, then for every isomorphism ϕ of G to $\hat{G}$ there exists an isomorphism $\phi_T : V(T) \to V(\hat{T})$ such that for every vertex $u \in V(G)$ and node $v \in V(T)$ it holds that $u \in \chi(v)$ if and only if $\phi(u) \in \hat{\chi}(\phi_T(v))$. Informally, every isomorphism ϕ that maps G to $\hat{G}$ also maps (T, χ) to $(\hat{T}, \hat{\chi})$. We will defer the discussion of how exactly we find such decompositions to the end of this section, because there are many different cases, and how exactly we find (T, χ) depends on the context.

Isomorphism Invariant Recursive Understanding. We are now ready to flesh out the overall scheme used in our algorithm. From now on, we will assume that all parameters that we introduce are upper bounded by some function of H, unless explicitly stated otherwise.

We are trying to find a canonical labeling for a graph G. However, just as in the recursive understanding scheme discussed in the beginning of this section, our algorithm will also take as input a distinguished set D. The interpretation is again that G is not necessarily the entire instance that we are trying to solve, and that the remaining "hidden" part of the instance attaches to G via D.

The first step is to compute an isomorphism invariant star decomposition (T, χ) of G with central bag $b \in V(T)$, such that $B = \chi(b)$, $D \subseteq B$, and B either has size upper bounded by $f(H)$ or has some nice properties (such as being (q, k)-unbreakable in G, or that for every pair of vertices u, v in B there is no u-v separator of size at most k.) Additionally we require all adhesions of (T, χ) to have size at most $f(k)$. Once we have identified (T, χ) we run the algorithm recursively on $G[\chi(\ell)]$ for every leaf ℓ in $V(T)$. Having "*understood*" each of the leaves we need to use this understanding to "*understand*" G with distinguished set D.

At this point we need to unpack precisely what we mean by understanding in the context of canonization. Because we know that every automorphism of G maps (T, χ) to itself we know that the center bag of (T, χ) maps to itself, while every leaf ℓ of (T, χ) maps in its entirety to itself, or to some other leaf ℓ' of (T, χ). We would like to use the "understanding" from the recursive calls in order to determine which leaves ℓ' the leaf ℓ can map to. Additionally, an automorphism of G might map ℓ to itself, but permute the vertices of the adhesion $\sigma(\ell)$. Therefore we also need to be able to use the "understanding" from the recursive calls to determine the set of permutations from $\sigma(\ell)$ to $\sigma(\ell)$ that can be completed to an automorphism of $G[\chi(\ell)]$.

It can be shown that both of these goals can be achieved if we formalize the "understanding" task for the leaves as follows: for every bijection $\pi : \sigma(\ell) \to [|\sigma(\ell)|]$ we need to find a canonical labeling of $G[\chi(\ell)]$ that coincides with π on $\sigma(\ell)$. Because the distinguished set D is actually one such $\sigma(\ell)$ in the recursive call corresponding to the parent of the current call in the recursion tree, the formalization of the "understanding" task for G with distinguished set D becomes to compute for every bijection $\pi : D \to [|D|]$ a canonical labeling of G that coincides with π on D.

The recursive scheme (compute (T, χ), understand all leaves recursively) now leaves us with the following task. We have as input a graph G and vertex set D and an isomorphism invariant star decomposition (T, χ) with central bag b, such that $D \subseteq \chi(b)$, together with a set of canonical labelings

$$\{\lambda_{\pi,\ell} : \ell \in V(T) \setminus \{b\} \text{ and } \pi : \sigma(\ell) \to [|\sigma(\ell)|] \text{ is a bijection}\}.$$

Here $\lambda_{\pi,\ell}$ is a canonical labeling of $G[\chi(\ell)]$ that coincides with π on $\sigma(\ell)$. The goal is to compute for every permutation $\pi : D \to [D]$ a canonical labeling λ_π of G that coincides with π on D.

At this point some case work is in order. The easy case is when the size of the center bag B is upper bounded by a function of H. In this case we can proceed in a manner almost identical to a single step of the dynamic programming step in the algorithm of [26] or even [3]. The idea is to simply brute force over all permutations of B.

The hard case is when B is large. In this case we need to use the fact that B has some "nice" property. For now, let us not worry precisely what the property is, because the next crucial step of the algorithm works independently of it.

Unpumping and Lifting. The case when the central bag B is large is the only step of the algorithm in which we actually employ the "replacing a graph by its representative" part of the recursive understanding technique. The *unpumping* procedure takes as input G with a distinguished set D, (T, χ), as well as the set of labelings $\{\lambda_{\pi,\ell}\}$ and produces the graph $G^\star$ by replacing every leaf $G[\chi(\ell)]$ with distinguished set $\sigma(\ell)$ by representative G_ℓ^R whose size is upper bounded by a function only of k. We will shortly discuss in more detail the properties of the representatives, but before that let us state the properties of the unpumped graph $G^\star$ that we need.

(1) (Lifting) For every permutation $\pi : D \to D$ a canonical labeling $\lambda_\pi^\star$ of $G^\star$ that coincides with π on D can be lifted in polynomial time to a canonical labeling λ_π, which coincides with π on D, of G.
(2) (Feasibility) If G is H-minor-free then $G^\star$ is H-minor-free.
(3) (Maintains Cut Properties Of B) If B is (q, k)-unbreakable in G then B is also (q, k)-unbreakable in $G^\star$. If, for every $u, v \in B$ there is no x-y separator of size at most k in G then there is no such separator in $G^\star$.
(4) (Small Leaves) $G^\star$ has a star decomposition $(T, \chi^\star)$ with the same decomposition star T as G, with $\hat{\chi}(b) = \chi(b)$ and $|\hat{\chi}(\ell)| \leqslant g(k)$ for some function g. *Remark*: This function g is possibly not computable.

Representatives for Canonization. Before proceeding to discussing how the algorithm uses the unpumping/lifting procedure we need to discuss how it is able to guarantee the key properties of $G^\star$. These properties follow by how we define the representative G_ℓ^R for each leaf $(G[\chi(\ell)], \sigma(\ell))$. Specifically, when we compute a representative of ℓ we need to ensure that:

- (Lifting I) The isomorphism class of G_ℓ^R depends only on the isomorphism class of $(G[\chi(\ell)], \sigma(\ell))$.
- (Lifting II) For all permutations $\iota : \sigma(\ell) \to \sigma(\ell)$, the graph $(G[\chi(\ell)], \sigma(\ell))$ has an automorphism that coincides with ι on $\sigma(\ell)$ if and only if G_ℓ^R does.
- (Feasibility) G_ℓ^R contains exactly the same set of minors on at most $|V(H)|$ vertices, and these minors intersect with $\sigma(\ell)$ in exactly the same way in G_ℓ^R and in $G[\chi(\ell)]$.

- (Maintains Cut Properties) For every separation (L, R) of $G[\sigma(\ell)]$ the minimum order of a separation of $G[\chi(\ell)]$ that coincides with (L, R) on $\sigma(\ell)$ and the minimum order of a separation of G^R_ℓ that coincides with (L, R) on $\sigma(\ell)$ are the same.
- (Small Leaves) $|V(G^R_\ell)|$ is upper bounded by a function only of H and k, and therefore only of H.

It is fairly easy to see that the properties (Feasibility), (Maintains Cut Properties) and (Small Leaves) of unpumping follow from the corresponding properties of representatives. On the other hand, an attentive reader should be worried about (Lifting) for the unpumping property to follow from properties (Lifting I) and (Lifting II) of representatives. Indeed, the graph $G[\chi(\ell)]$ is possibly a large graph on up to n vertices, while the size of G^R_ℓ is bounded as a function of H. Thus, by pigeon hole principle non-isomorphic graphs $G[\chi(\ell)]$ may end up having the same representative G^R_ℓ.

How can we now ensure that such leaves do not get mapped to each other by automorphisms of $G^\star$? Well, before unpumping we can use the canonical labelings of $G[\sigma(\ell)]$ for every leaf ℓ to determine which leaves ℓ and ℓ' have isomorphic graphs $G[\sigma(\ell)]$ and $G[\sigma(\ell')]$. Then we encode non-isomorphism of non-isomorphic leaves in G by assigning in $G^\star$ colors to vertices in the representatives so that the representatives of non-isomorphic leaves receive distinct colors.

$G^\star$ Inherits Properties of B. As we already have alluded to before, the properties of B that we will be maintaining are either that B is (q, k)-unbreakable in G and in $G^\star$, or even stronger, that there is no x-y separator of size at most k in G and in $G^\star$ for every pair of vertices x, y in B.

This, together with the property (Small Leaves) can be used to show that $G^\star$ itself is (q', k)-unbreakable for q' that is only a function of q and the size of the leaves (in the first case), or is is (q, k, k)-*improved clique unbreakable* (in the second case). Here (q, k, k)-improved clique unbreakable means that there is no separation (L, R) of G of order at most k such that $\min\{|L|, |R|\} \geqslant q$ and for every pair x, y of vertices in $L \cap R$ the minimum order of an x-y separation is at least $k + 1$. Thus, the good property of the bag B in G is in an approximate sense inherited by the entire unpumped graph $G^\star$.

Overall Scheme, Again. Our reduction from general graphs to unbreakable graphs proceeds in two steps. In the first step we reduce from general graphs to improved clique-unbreakable graphs, while in the second step we reduce from improved clique-unbreakable graphs to unbreakable graphs.

In the first step, the reduction from general graphs to improved clique-unbreakable graphs, all we still have to do is to design an algorithm that given a general graph G and set D, outputs an isomorphism invariant star decomposition (T, χ) of G such that the central bag B of (T, χ) satisfies that there is no x-y separator of size at most k in G for every pair of vertices x, y in B. If we can achieve this, then the argument in the previous section yields that $G^\star$ is improved clique unbreakable, and we have achieved a reduction to improved clique unbreakable graphs. This can be done by essentially observing that it was already done by Elberfeld and Schweitzer [10]. However we need to modify their algorithm to work in FPT time rather than log space and $O(n^k)$ time.

In the second case we have (G, D) and the promise that G is improved clique unbreakable. What we need is to find a star decomposition (T, χ) where center B is unbreakable, because then $G^\star$ will be unbreakable as desired.

Finding Next Bag in Improved Clique Unbreakable Case. Instead of considering the (q, k, k)-improved clique unbreakable case, we will make the simplifying assumption that G has no separator (L, R) of order at most k, such that for every x, y in $L \cap R$ the minimum order of an x-y separation is at least $k + 1$.

Again we are given as input a graph G with a distinguished vertex set D and the task is to find a isomorphism invariant star decomposition (T, χ) with central bag B such that $D \subseteq B$ and B is (q, k)-unbreakable in G.

There are two cases, either D is small ($|D| \leqslant k$) or D is large. If D is small and there exists a pair x, y such that the minimum order of an x-y separation in G is at most k, we can proceed in a similar manner as the corresponding case in the algorithm of [26].

On the other hand, if D is small and no such pair exists, then D yields precisely the type of separation in G that we just assumed does not exist. Therefore this case does not apply. This is the only place in the algorithm where we use the assumption that G is (q, k, k)-improved clique unbreakable.

If B is large and breakable (there exists a separation (L, R) of order $i < \min(|L \cap B|, |R \cap B|)$ then we can use the "notion of stable separators" defined in [26] and make progress. Finally, when B is large and unbreakable, we observe that the idea of important separator extension used in designing an algorithm for Minimum Bisection [7] gives an unbreakable bag B in an isomorphism invariant way.

In either case we are able to find a star decomposition (T, χ) where the central bag B is either small or unbreakable. When it is small we can brute force, while when it is big and unbreakable, the unpumped graph $G^\star$ is unbreakable and so we can solve the problem for the unbreakable graph $G^\star$ and lift the solution back to G using the lifting algorithm. This concludes the proof sketch of Theorem 1.4.

2.2 Canonizing Unbreakable Graphs with an Excluded Minor

In this section we provide an intuitive overview of the proof of Theorem 1.6. That is, given an unbreakable H-minor-free graph G, we would like to partition the vertex set of G into a rigid part V_{genus} and a bounded-treewidth part V_{tw} in an isomorphism-invariant way.

To simplify the exposition, we first focus on proving Theorem 1.6 under a stronger assumption that G actually has bounded genus. This case already allows us to show most of the key conceptual steps used in the argument for the general case of unbreakable H-minor-free graphs. Then, we briefly discuss traps, issues, and caveats that arise when working in the general setting with near-embeddings rather than embeddings.

Graphs of Bounded Genus. In this section we sketch how to prove the following statement, which is weaker variant of Theorem 1.6 tailored to the bounded genus case.

Theorem 2.1. *There exist computable functions $f_{\mathrm{cut}}, f_{\mathrm{tw}}, f_{\mathrm{time}}$ and an algorithm $\mathcal{A}$ with the following specification. The algorithm $\mathcal{A}$ is given as input a graph G and integers g and q with a promise that G is of Euler genus at most g and is (q,k)-unbreakable for $k = f_{\mathrm{cut}}(g)$. Then $\mathcal{A}$ runs in time bounded by $f_{\mathrm{time}}(g,q) \cdot n^{O(1)}$ and computes an isomorphism-invariant partition of $V(G)$ into $V_{\mathrm{tw}} \uplus V_{\mathrm{genus}}$ and a nonempty isomorphism-invariant family $\mathcal{F}_{\mathrm{genus}}$ of size $O(|E(G)|)$ of bijections $V_{\mathrm{genus}} \mapsto [|V_{\mathrm{genus}}|]$ such that $G[V_{\mathrm{tw}}]$ is of treewidth at most $f_{\mathrm{tw}}(g) \cdot q$.*

Without loss of generality assume that $q \geqslant k \geqslant 2$. If G is of treewidth $O(q)$, then we can return $V_{\mathrm{genus}} = \emptyset$. Hence, we assume that the treewidth of G is much larger than q.

Consider Tutte's decomposition of G into 3-connected components[1]. Since G is (q,k)-unbreakable and has treewidth much larger than q, it follows that there is a unique 3-connected component of G that has more than q vertices and treewidth much larger than q, while all the other 3-connected components are of size at most q. The same conclusion holds for the graph $G-X$ for any set $X \subseteq V(G)$ of size at most $k-2$: If (A,B) is a separation of order at most 2 in $G-X$, then $(A \cup X, B \cup X)$ is a separation of order at most k in G and, hence, either $|A \setminus B| \leqslant q$ or $|B \setminus A| \leqslant q$. For any set $X \subseteq V(G)$ of size at most $k-2$, by G^X we denote the unique 3-connected component of $G-X$ that has more than q vertices. Note that the treewidth assumption implies that the treewidth of G^X is in fact much larger than q.

Our goal is to define V_{genus} so that $G[V_{\mathrm{genus}}]$ is almost rigid — admits only few automorphisms — and this rigidity will be derived from the rigidity of embedded graphs. Observe that if Π is an embedding of a connected graph G in some surface, then an automorphism of (G,Π) is fixed by fixing only the image of one edge with distinguished endpoint and an indication which side of the edge is "left" and which is "right". This gives at most $4|E(G)|$ automorphisms.

Unfortunately, a graph can have many different embeddings in a surface of Euler genus g, even assuming it is 3-connected. However, the number of different embeddings drops if the embeddings have large face-width (the minimum number of vertices on a noncontractible face-vertex noose, i.e., a closed curve without self-intersections):

Theorem 2.2 ([36]). *There is a function $f_{\mathrm{ue}}(g) \in O(\log g/\log\log g)$ such that if G is a 3-connected graph and Π is an embedding of G of Euler genus g and face-width at least $f_{\mathrm{ue}}(g)$, then g is equal to the (minimum) Euler genus of G and Π is the unique embedding of Euler genus g.*

The first idea would be to apply Theorem 2.2 to $G^{\emptyset}$ — the unique large 3-connected component of G. That is, consider an embedding Π of $G^{\emptyset}$ of minimum possible Euler genus. If the face-width of this embedding is at least $f_{\mathrm{ue}}(g)$, then we may simply set $V_{\mathrm{genus}} = V(G^{\emptyset})$. Then Π is the unique embedding of $G[V_{\mathrm{genus}}]$ of minimum Euler genus, which gives rise to an isomorphism-invariant family of at most $4|E(G)|$ automorphisms of G, from which a suitable family $\mathcal{F}_{\mathrm{genus}}$ can be easily derived. Note that by (q,k)-unbreakability, every connected component of $G[V_{\mathrm{tw}}] = G - V_{\mathrm{genus}}$ has at most q vertices, hence in particular $G[V_{\mathrm{tw}}]$ has treewidth at most q.

However, it may happen that Π has face-width smaller than $f_{\mathrm{ue}}(g)$ and Theorem 2.2 cannot be applied. But then there is a set $X_1 \subseteq V(G^{\emptyset})$ of size at most $f_{\mathrm{ue}}(g)$ such that $G^{\emptyset} - X_1$ has strictly smaller Euler genus than $G^{\emptyset}$. This implies that G^{X_1} has strictly smaller Euler genus than $G^{\emptyset}$.

We can iterate this process: if we take any minimum Euler genus embedding Π_1 of G^{X_1} and it turns out that Π_1 has small face-width, there is a set X_2 of small cardinality such that $G^{X_1 \cup X_2}$ has strictly smaller Euler genus. It will be useful later to allow subsequent steps in this process to ask for larger and larger face-width (and thus allowing larger cut sets $X_2, X_3, \ldots$). Note that the number of steps is bounded by the Euler genus of G.

More formally, let us define a function $\kappa\colon \{0,1,\ldots,g,g+1\} \to \mathbb{N}$ by setting $\kappa(0) = 0$ and $\kappa(\gamma+1) = p_\kappa(\kappa(\gamma))$ for a polynomial $p_\kappa(x) = x + f_{\mathrm{ue}}(g) + c \cdot (x+g+1)^4$, where c is a sufficiently large constant. We set $k = f_{\mathrm{cut}}(g) = \kappa(g+1) := p_\kappa(\kappa(g)) \in 2^{2^{O(g)}}$.

Let $0 \leqslant \gamma \leqslant g$. A set $X \subseteq V(G)$ is a *potential deletion set for* γ if $|X| \leqslant \kappa(\gamma)$ and the Euler genus of G^X is at most $g-\gamma$. Let $0 \leqslant \gamma_0 \leqslant g$ be the maximum integer such that there exists a potential deletion set for γ_0; note that γ_0 exists as $X = \emptyset$ is a potential deletion set for $\gamma = 0$. Let $\kappa_0 \leqslant \kappa(\gamma_0)$ be the minimum size of a potential deletion set for γ_0. A potential deletion set for γ_0 of size κ_0 shall be called a *deletion set.*

Let $\mathcal{X}$ be the family of all deletion sets. Let $X_0 = \bigcap \mathcal{X}$ be the set of those vertices of G that are contained in every deletion set; clearly $|X_0| \leqslant \kappa_0$. We define also $Z := \bigcup \mathcal{X}$ to be the set of all vertices contained in any deletion set. We remark that known algorithms for Genus Vertex Deletion (e.g., [25]) can be modified to compute γ_0, κ_0, X_0, Z, and an arbitrary element of $\mathcal{X}$ in FPT time when parameterized by g.

In this overview we assume $\gamma_0 < g$; the case $\gamma_0 = g$ (i.e., we reach a planar graph in the end of the process described above) can be handled very similarly, using the fact that a 3-connected plane graphs has a unique planar embedding. By the choice of γ_0, for every $X \in \mathcal{X}$ the graph G^X admits an embedding Π^X of Euler genus $g-\gamma_0$ and face-width at least $f_{\mathrm{ue}}(g) + c\cdot(\kappa_0+g+1)^4$, i.e., much larger than both κ_0 and $f_{\mathrm{ue}}(g)$. In particular, Π^X has the minimum possible Euler genus, which is $g-\gamma_0$, and is the unique embedding of G^X of this Euler genus.

Although it can be easily seen that the treewidth of $G - V(G^X)$ is $O(q+\kappa_0)$ and the uniqueness of the embedding of G^X makes its automorphism group simple, we cannot return $V_{\mathrm{genus}} = V(G^X)$, because the choice of $X \in \mathcal{X}$ is not isomorphism-invariant. The crux of the approach lies in showing that the graph G^Z can be defined similarly as G^X, while now Z — the union of all deletion sets — is defined in an isomorphism-invariant way. Namely, G^Z is the unique large 3-connected component of $G-Z$, which can be well defined despite the fact that the size of Z may be much larger than k. Then G^Z has similar embedding properties as G^X, while it turns out that $G - V(G^Z)$ still has low treewidth. Thus we can return $V_{\mathrm{genus}} = V(G^Z)$. Intuitively, the key to proving the viability

[1]The 3-connected components of a graph are the torsos of the bags of its Tutte's decomposition.

of this strategy is to show that two elements $X, X' \in \mathcal{X}$ cannot differ too much from each other, and therefore $Z = \bigcup_{X \in \mathcal{X}} X$ is structurally not much different from every single $X \in \mathcal{X}$.

For $X \in \mathcal{X}$ and $w \in V(G) \setminus X$, we define the *projection* $\pi^X(w) \subseteq V(G^X)$ as follows. If $w \in V(G^X)$, then $\pi^X(w) = \{w\}$. Otherwise, letting C_w be the component of $G - X - V(G^X)$ containing w, we set $\pi^X(w) = N_{G-X}(C_w)$. Note that $|N_{G-X}(C_w)| \in \{0, 1, 2\}$ and if $|N_{G-X}(C_w)| = 2$, then G^X contains an edge connecting the two elements of $N_{G-X}(C_w)$.

For $X \in \mathcal{X}$, we define the set Γ^X of *attachment points* as follows. For every $v \in X \setminus X_0$, we define the set $\Gamma^X(v)$ of attachment points of v as the union of $\pi^X(u)$ over all $u \in N_G(v) \setminus X$. Finally, we define $\Gamma^X = \bigcup_{v \in X \setminus X_0} \Gamma^X(v)$.

The first step in our analysis of $\mathcal{X}$ is to observe that the attachment points are local in the following sense.

Lemma 2.3 (informal statement). *For every $X \in \mathcal{X}$, there is a set $\Lambda^X \subseteq \Gamma^X$ of size $O((g + \kappa_0 + 1)^2)$ such that Γ^X is covered by the union of radial balls in Π^X of radius 9 centered at vertices of Λ^X.*

The proof of Lemma 2.3 goes roughly as follows: if for some $v \in X$ the set $\Gamma^X(v)$ cannot be covered by a small number of radial balls, then one can pack many vertex-disjoint radial balls of radius 3 around the attachment points. Using the 3-connectivity of G^X, one can argue that these radial balls give rise to many K_5 minor models intersecting only at v. This implies $v \in X_0$, a contradiction.

Using the above, we prove that any two elements of $\mathcal{X}$ need to be somewhat similar, in the sense that their attachment points are close to each other.

Lemma 2.4 (informal statement). *Consider any $X, X' \in \mathcal{X}$. Then for every $w \in X' \setminus X$, the elements of $\pi^X(w)$ are within radial distance $O((g + \kappa_0)^2)$ from Λ^X in Π^X.*

For the proof of Lemma 2.4, assume there is $w \in X' \setminus X$ with $\pi^X(w)$ radially far from Λ^X (and thus also far from Γ^X). Since κ_0 is still significantly smaller than k, the graph $G^{X' \cup X}$ is well-defined and it has a unique embedding $\Pi^{X \cup X'}$ with large face-width and of the same Euler genus $g - \gamma_0$. One can now argue that a big part of the graph "between" Λ^X and $\pi^X(w)$ is embedded in the same way in embeddings Π^X, $\Pi^{X'}$, and $\Pi^{X \cup X'}$. This gives ground for a replacement argument: if Π^X is able to embed $\pi^X(w)$, and the area between $\pi^X(w)$ and Λ^X is embedded in the same way in Π^X and in $\Pi^{X'}$, then one can modify the embedding $\Pi^{X'}$ using a part of the embedding Π^X to obtain an embedding of $G^{X' \setminus \{w\}}$. This embedding can be turned into an embedding of $G - (X' \setminus \{w\})$ of Euler genus $g - \gamma_0$, contradicting the minimality of $X' \in \mathcal{X}$.

With Lemma 2.4 established, we can proceed to the analysis of $Z = \bigcup_{X \in \mathcal{X}} X$. Lemma 2.4 shows that for any fixed $X \in \mathcal{X}$, $Z \setminus X$ is contained in $O((g + \kappa_0)^2)$ radial balls in Π^X, each of radius $O((g + \kappa_0)^2)$. First, since $|X| = \kappa_0$, this gives a bound on the treewidth of $G[Z]$ using the fact that graphs of bounded genus have bounded local treewidth — the treewidth is bounded linearly in the radial radius. Second, it shows that $G^X \setminus Z$, with the embedding inherited from G^X, still has large face-width. This allows us to define G^Z as the unique 3-connected component of $G - Z$ with Euler genus $g - \gamma_0$ and large face-width of the embedding.

The face-width lower bound makes the embedding of G^Z unique, imposing a very rigid structure on the automorphism group of G^Z. With some technical care, the treewidth bound on $G[Z]$ can be lifted to a treewidth bound on $G - V(G^Z)$. Since the definition of Z is isomorphism-invariant, our definition of G^Z is also isomorphism invariant. So all requirements are satisfied to set $V_{\text{genus}} = V(G^Z)$ and $V_{\text{tw}} = V(G - V(G^Z))$. This finishes the sketch of the proof of Theorem 2.1.

The General Case. We now discuss how the strategy presented above can be lifted to the general case of H-minor-free graphs. Here, the main tool will be the Structure Theorem of Robertson and Seymour. We first need to recall some terminology.

A *tangle* $\mathcal{T}$ of order p in a graph G is a family of separations of order less than p in G such that (i) for every separation (A, B) of order less than p, either (A, B) or (B, A) is in $\mathcal{T}$, (ii) for every three elements $(A_1, B_1), (A_2, B_2), (A_3, B_3) \in \mathcal{T}$, we have $A_1 \cup A_2 \cup A_3 \neq V(G)$. If $(A, B) \in \mathcal{T}$, then A is the *small side* and B is the *big side* of the separation (A, B). For a vertex subset X and a tangle $\mathcal{T}$, by $\mathcal{T} - X$ we denote the set of all separations (A, B) of $G - X$ such that $(A \cup X, B \cup X) \in \mathcal{T}$. It is straightforward to check that if $\mathcal{T}$ has order $p > |X|$, then $\mathcal{T} - X$ is a tangle of order $p - |X|$.

Let us come back to the proof of Theorem 1.6. The threshold k will be set later, but let us imagine for a moment that it is defined. Then note that the unbreakability assumption on G in Theorem 1.6 allows us to define the *unbreakability tangle* of order $k+1$: the tangle consists of all separations (A, B) of order at most k with $|A| \leqslant q(k)$. This defines a tangle as long as $|G| > 3q(k)$, and otherwise we can return $V_{\text{tw}} = V(G)$ and $V_{\text{genus}} = \emptyset$.

In the general case we will work with *near-embeddings* of graphs; in this matter, we mostly follow the the notation from [9]. Given a graph G, a *near-embedding* of G consists of:

(1) a set $\mathbf{A} \subseteq V(G)$, called the *apices*;
(2) a partition of the graph $G - \mathbf{A}$ into edge-disjoint subgraphs

$$G - \mathbf{A} = G_0 \cup \bigcup_{i=1}^{k^\circ} G_i^\circ \cup \bigcup_{i=1}^{k^\triangle} G_i^\triangle;$$

(3) an embedding Π of G_0 into a (compact and connected) surface Σ.

The following properties have to be satisfied:

(1) Every vertex of $G - \mathbf{A}$ that belongs to at least two subgraphs of the partition, belongs to G_0.
(2) For every $1 \leqslant i \leqslant k^\triangle$, $|V(G_i^\triangle) \cap V(G_0)| \leqslant 3$. (The graphs $G_i^\triangle$ are henceforth called *dongles*.)
(3) The graphs $(G_i^\circ)_{i=1}^{k^\circ}$ are pairwise vertex-disjoint.
(4) For every $1 \leqslant i \leqslant k^\circ$, the set $V(G_i^\circ) \cap V(G_0)$ can be enumerated as $\{v_{i,1}^\circ, \ldots, v_{i,n_i^\circ}^\circ\}$ so that G_i° admits a path decomposition $(\mathbf{B}_{i,1}^\circ, \ldots, \mathbf{B}_{i,n_i^\circ}^\circ)$ with $v_{i,j}^\circ \in \mathbf{B}_{i,j}^\circ$ for every $1 \leqslant j \leqslant n_i^\circ$. The graph G_i° is called a *vortex* and the vertices $v_{i,j}^\circ$ are called the *society vertices* of the vortex G_i°. The number of society vertices, n_i°, is the *length* of a vortex.
(5) On the surface Σ there exist closed discs with disjoint interiors $(D_i^\triangle)_{i=1}^{k^\triangle}$ and $(D_i^\circ)_{i=1}^{k^\circ}$ such that Π embeds G_0 into the closure of $\Sigma \setminus \left(\bigcup_{i=1}^{k^\triangle} D_i^\triangle \cup \bigcup_{i=1}^{k^\circ} D_i^\circ\right)$ and the following conditions hold:
   - For every $1 \leqslant i \leqslant k^\triangle$, the vertices of G_0 that lie on the boundary of $D_i^\triangle$ are exactly the vertices of $V(G_i^\triangle) \cap V(G_0)$.

- For every $1 \leqslant i \leqslant k^{\circ}$, the vertices of G_0 that lie on the boundary of D_i° are exactly the vertices of $V(G_i^{\circ}) \cap V(G_0)$. Furthermore, these vertices lie around the boundary of D_i° in the order $v_{i,1}^{\circ}, \ldots, v_{i,n_i^{\circ}}^{\circ}$.

We say that a near-embedding $\mathcal{E}$ *captures* a tangle $\mathcal{T}$ if for each separation of $\mathcal{T} - \mathbf{A}$, the big side of this separation is not contained in any dongle $G_i^{\triangle}$ or in any vortex G_i°.

The following statement is the main structural result of the Graph Minors series and appears as (3.1) in [35].

Theorem 2.5. *For every nonplanar graph H there exist integers $k_H, \alpha_H \geqslant 0$ such that for every H-minor-free graph G and every tangle $\mathcal{T}$ of G of order at least k_H, G admits a near-embedding capturing $\mathcal{T}$ where the number of apices, the Euler genus of the surface, the number of vortices, and the maximum adhesion size of a vortex are all bounded by α_H.*

Thus, if we set the threshold k to be larger than $k_0 = k_H + \alpha_H + 1$, then Theorem 2.5 asserts that there exists a near-embedding $\mathcal{E}$ of G that captures the unbreakability tangle where the number of apices and the maximum adhesion size of a vortex is bounded by k_0. This in particular implies that every dongle and every bag of a vortex has size bounded by $q_0 \coloneqq q(k_0)$.

To imitate the role of κ from the bounded genus case, consider the following refinement process. Recall that Π is the embedding of G_0 in the near-embedding $\mathcal{E}$. If the face-width of Π is small (i.e., bounded by a function of q_0), or if two vortices are close in the radial distance in Π (again, meaning that the radial distance is bounded as a function of q_0), we can move all vertices appearing on the corresponding curve on the surface to the apex set, decreasing either the Euler genus or the number of vortices. If such a curve passes through a vortex or a disc corresponding to a dongle, we move also to the apex set the entire dongle or two bags of a vortex — the ones at society vertices where the curve entered and left the disc of a vortex. In this way we have obtained a new near-embedding $\widetilde{\mathcal{E}}$, which uses fewer vortices or a simpler surface, but where the number of apices has grown to at most some function of q_0 — call it k_1. Denote $q_1 = q(k_1)$ and iterate this process, as in the bounded genus case. The number of iterations is bounded by $O(\alpha_H)$, since every iteration either decreases the genus or the number of vortices. So eventually the process stops after some $\iota \leqslant O(\alpha_H)$ iterations, reaching a near-embedding that has at most k_ι apices, but has face-width and the radial distance between vortices larger than any function of q_ι fixed in advance. Here we stop; we will analyze the family of all such near-embeddings, and call them henceforth *optimal*.

Note that the argument above uses the unbreakability assumption only for a bounded — in terms of k_H, α_H, and the function q — initial values q. The bound on the number of those values constitutes our threshold k.

At this moment, most of the reasoning for bounded genus can be adjusted. First, there is a well-defined notion of *universal apices*: these are vertices of G that are apices in all optimal-near embeddings. Then we can prove an analog of Lemma 2.3, which shows that in every optimal near-embedding, the attachment points of non-universal apices appear locally on the surface: up to technical details, they can be covered by a bounded number of bounded-radius balls. Second, leveraging on that, we prove an analog of Lemma 2.4: for any two optimal near-embeddings $\mathcal{E}$ and $\widetilde{\mathcal{E}}$, if we additionally assume that $\widetilde{\mathcal{E}}$ has inclusion-wise minimal set of vertices contained in vortices, then every vertex that is an apex or is in a vortex in $\widetilde{\mathcal{E}}$, is also either an apex of $\mathcal{E}$ or is close (in radial distance) to a vortex or an attachment point of a non-universal apex of $\mathcal{E}$.

Finally, this allows us to argue the following. Define Z to be the set of all vertices of G that can be apices or be contained in vortices in an optimal near-embedding with an inclusion-wise minimal set of vertices contained in vortices. Then $G - Z$ contains a unique 3-connected component H of large treewidth. Further, any optimal near-embedding of G as above projects to a near-embedding of H without vortices or apices, but into the same surface and with still large face-width. This allows us to use Theorem 2.2 to be able to proclaim $V(H)$ to be the V_{genus} part. As before, we are left with arguing that the graph $G - V_{\text{genus}} = G[V_{\text{tw}}]$ has bounded treewidth. Again, this follows from the fact that surface-embedded graphs have locally bounded treewidth, which we use in conjunction with the radial bounds provided by the analog of Lemma 2.4.

Note that in the plan sketched above, we cannot directly apply Theorem 2.2 to H, as H is not a completely embeddable graph: the near-embedding inherited from a near-embedding of G has no apices or vortices, but still may contain dongles. To circumvent this issue, we prove that in 3-connected graphs, the dongles can be chosen in a canonical way so that we may essentially speak about the *unique* near-embedding of H. We remark that the canonical choice of dongles is also crucially used in the replacement argument underlying the proof of the analog of Lemma 2.4.

REFERENCES

[1] László Babai. 2016. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016.* ACM, 684–697. https://doi.org/10.1145/2897518.2897542

[2] László Babai and Eugene M. Luks. 1983. Canonical Labeling of Graphs. In *STOC.* 171–183.

[3] Hans L. Bodlaender. 1990. Polynomial Algorithms for Graph Isomorphism and Chromatic Index on Partial k-Trees. *J. Algorithms* 11, 4 (1990), 631–643.

[4] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. 2016. Designing FPT Algorithms for Cut Problems Using Randomized Contractions. *SIAM J. Comput.* 45, 4 (2016), 1171–1229. https://doi.org/10.1137/15M1032077

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

[6] Marek Cygan, Paweł Komosa, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, Saket Saurabh, and Magnus Wahlström. 2021. Randomized Contractions Meet Lean Decompositions. *ACM Trans. Algorithms* 17, 1 (2021), 6:1–6:30. https://doi.org/10.1145/3426738

[7] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2014. Minimum bisection is fixed parameter tractable. In *STOC.* 323–332.

[8] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2019. Minimum Bisection Is Fixed-Parameter Tractable. *SIAM J. Comput.* 48, 2 (2019), 417–450. https://doi.org/10.1137/140988553

[9] Reinhard Diestel, Ken-ichi Kawarabayashi, Theodor Müller, and Paul Wollan. 2012. On the excluded minor structure theorem for graphs of large tree-width. *J. Comb. Theory, Ser. B* 102, 6 (2012), 1189–1210. https://doi.org/10.1016/j.jctb.2012.07.001

[10] Michael Elberfeld and Pascal Schweitzer. 2017. Canonizing Graphs of Bounded Tree Width in Logspace. *ACM Trans. Comput. Theory* 9, 3 (2017), 12:1–12:29. https://doi.org/10.1145/3132720

[11] I. S. Filotti and Jack N. Mayer. 1980. A Polynomial-time Algorithm for Determining the Isomorphism of Graphs of Fixed Genus (Working Paper). In *STOC.* 236–243.

[12] Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. 2011. Finding topological subgraphs is fixed-parameter tractable. In *STOC.* 479–488.

[13] Martin Grohe and Dániel Marx. 2012. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. In *STOC.* 173–192.

[14] Martin Grohe and Daniel Neuen. 2020. Recent Advances on the Graph Isomorphism Problem. *CoRR* abs/2011.01366 (2020). arXiv:2011.01366 https://arxiv.org/abs/2011.01366

[15] Martin Grohe and Daniel Neuen. 2021. Isomorphism, canonization, and definability for graphs of bounded rank width. *Commun. ACM* 64, 5 (2021), 98–105. https://doi.org/10.1145/3453943

[16] Martin Grohe, Daniel Neuen, and Pascal Schweitzer. 2018. A Faster Isomorphism Test for Graphs of Small Degree. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*, Mikkel Thorup (Ed.). IEEE Computer Society, 89–100. https://doi.org/10.1109/FOCS.2018.00018

[17] Martin Grohe, Daniel Neuen, Pascal Schweitzer, and Daniel Wiebking. 2020. An Improved Isomorphism Test for Bounded-tree-width Graphs. *ACM Trans. Algorithms* 16, 3 (2020), 34:1–34:31. https://doi.org/10.1145/3382082

[18] Martin Grohe and Pascal Schweitzer. 2015. Isomorphism Testing for Graphs of Bounded Rank Width. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*. IEEE Computer Society, 1010–1029. https://doi.org/10.1109/FOCS.2015.66

[19] Martin Grohe, Daniel Wiebking, and Daniel Neuen. 2020. Isomorphism Testing for Graphs Excluding Small Minors. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE, 625–636. https://doi.org/10.1109/FOCS46700.2020.00064

[20] John E. Hopcroft and Robert Endre Tarjan. 1972. Isomorphism of Planar Graphs. In *Complexity of Computer Computations*. 131–152.

[21] John E. Hopcroft and Robert Endre Tarjan. 1973. A $V \log V$ Algorithm for Isomorphism of Triconnected Planar Graphs. *J. Comput. Syst. Sci.* 7, 3 (1973), 323–331.

[22] John E. Hopcroft and J. K. Wong. 1974. Linear Time Algorithm for Isomorphism of Planar Graphs (Preliminary Report). In *STOC*. 172–184.

[23] Ken-ichi Kawarabayashi. 2015. Graph Isomorphism for Bounded Genus Graphs In Linear Time. *CoRR* abs/1511.02460 (2015). arXiv:1511.02460 http://arxiv.org/abs/1511.02460

[24] Ken-ichi Kawarabayashi and Mikkel Thorup. 2011. The Minimum k-way Cut of Bounded Size is Fixed-Parameter Tractable. In *FOCS*. 160–169.

[25] Tomasz Kociumaka and Marcin Pilipczuk. 2019. Deleting Vertices to Graphs of Bounded Genus. *Algorithmica* 81, 9 (2019), 3655–3691. https://doi.org/10.1007/s00453-019-00592-7

[26] Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2017. Fixed-Parameter Tractable Canonization and Isomorphism Test for Graphs of Bounded Treewidth. *SIAM J. Comput.* 46, 1 (2017), 161–189. https://doi.org/10.1137/140999980

[27] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. 2018. Reducing CMSO Model Checking to Highly Connected Graphs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic (LIPIcs, Vol. 107)*, Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 135:1–135:14. https://doi.org/10.4230/LIPIcs.ICALP.2018.135

[28] Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. 2020. A Parameterized Approximation Scheme for Min k-Cut. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*. IEEE, 798–809. https://doi.org/10.1109/FOCS46700.2020.00079

[29] Eugene M. Luks. 1982. Isomorphism of Graphs of Bounded Valence can be Tested in Polynomial Time. *J. Comput. Syst. Sci.* 25, 1 (1982), 42–65.

[30] Gary L. Miller. 1980. Isomorphism Testing for Graphs of Bounded Genus. In *STOC*. 225–235.

[31] Daniel Neuen. 2020. Hypergraph Isomorphism for Groups with Restricted Composition Factors. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020 (LIPIcs, Vol. 168)*. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 88:1–88:19. https://doi.org/10.4230/LIPIcs.ICALP.2020.88

[32] Daniel Neuen. 2021. Isomorphism Testing Parameterized by Genus and Beyond. In *Proceedings of the 29th Annual European Symposium on Algorithms, ESA 2021 (LIPIcs, Vol. 204)*. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 72:1–72:18. https://doi.org/10.4230/LIPIcs.ESA.2021.72

[33] Michał Pilipczuk, Nicole Schirrmacher, Sebastian Siebertz, Szymon Toruńczyk, and Alexander Vigny. 2021. Algorithms and data structures for problems definable in first-order logic with connectivity predicates. Manuscript.

[34] I.N. Ponomarenko. 1991. The isomorphism problem for classes of graphs closed under contraction. *Journal of Soviet Mathematics* 55, 2 (1991), 1621–1643.

[35] Neil Robertson and Paul D. Seymour. 2003. Graph Minors XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B* 89, 1 (2003), 43–76.

[36] Paul D. Seymour and Robin Thomas. 1996. Uniqueness of highly representative surface embeddings. *Journal of Graph Theory* 23, 4 (1996), 337–349. https://doi.org/10.1002/(SICI)1097-0118(199612)23:4<337::AID-JGT2>3.0.CO;2-S

[37] H. Weinberg. 1966. A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. *Circuit Theory* 13 (1966), 142–148.

[38] Daniel Wiebking. 2020. Graph Isomorphism in Quasipolynomial Time Parameterized by Treewidth. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020 (LIPIcs, Vol. 168)*. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 103:1–103:16. https://doi.org/10.4230/LIPIcs.ICALP.2020.103