Faster Parameterized Algorithms for Minor Containment*

Isolde Adler[1]**, Frederic Dorn[2]**, Fedor V. Fomin[2]**,
Ignasi Sau[3]***, and Dimitrios M. Thilikos[4]†

[1] Institut für Informatik, Goethe-Universität, Frankfurt, Germany. E-mail: iadler@informatik.uni-frankfurt.de

[2] Department of Informatics, University of Bergen, Norway. E-mail: {frederic.dorn,fedor.fomin}@ii.uib.no

[3] AlGCo project-team, CNRS, LIRMM, Montpellier, France. E-mail: ignasi.sau@lirmm.fr

[4] Department of Mathematics, National and Kapodistrian University of Athens, Greece. E-mail: sedthilk@math.uoa.gr

Abstract. The H-Minor containment problem asks whether a graph G contains some fixed graph H as a minor, that is, whether H can be obtained by some subgraph of G after contracting edges. The derivation of a polynomial-time algorithm for H-Minor containment is one of the most important and technical parts of the Graph Minors Theory of Robertson and Seymour and it is a cornerstone for most of the algorithmic application of this theory. H-Minor containment for graphs of bounded branchwidth is a basic ingredient of this algorithm. The currently fastest solution to this problem, based on the ideas introduced by Robertson and Seymour, was given by Hicks in [*Networks, 43(1):1–9, 2004*], providing an algorithm that in time $\mathcal{O}(3^{k^2} \cdot (h+k-1)! \cdot m)$ decides if a graph G with m edges and branchwidth k, contains a fixed graph H on h vertices as a minor. In this work we improve the dependence on k of Hicks' result by showing that checking if H is a minor of G can be done in time $\mathcal{O}(2^{(2k+1)\cdot\log k} \cdot h^{2k} \cdot 2^{2h^2} \cdot m)$. We set up an approach based on a combinatorial object called *rooted packing*, which captures the properties of the subgraphs of H that we seek in our dynamic programming algorithm. This formulation with rooted packings allows us to speed up the algorithm when G is embedded in a fixed surface, obtaining the first algorithm for minor containment testing with single-exponential dependence on branchwidth. Namely, it runs in time $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$, with $n = |V(G)|$. Finally, we show that slight modifications of our algorithm permit to solve some related problems within the same time bounds, like induced minor or contraction containment.

Keywords: graph minors, branchwidth, graph minor containment, parameterized complexity, dynamic programming, graphs on surfaces.

---

* An extended abstract of this work appeared in the *Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT), volume 6139 of LNCS, pages 322-333, Bergen, Norway, June 2010.*

** Supported by the Norwegian Research Council.

*** Supported by AGAPE (ANR-09-BLAN-0159) and GRATOS (ANR-09-JCJC-0041-01) French projects.

† Supported by the project "Kapodistrias" (ΑΠ 02839/28.07.2008) of the National and Kapodistrian University of Athens (project code: 70/4/8757).

1 Introduction

Robertson and Seymour asserted Wagner's conjecture by showing that each minor-closed graph property can be characterized by a finite set of forbidden minors [26, 28]. Suppose that $\mathbf{P}$ is a property on graphs that is *minor-closed*, that is, if a graph has this property then all its minors have it too. Graph Minors Theory implies that there is a *finite* set $\mathcal{F}$ of *forbidden minors* such that a graph G has property $\mathbf{P}$ if and only if G does not have any of the graphs in $\mathcal{F}$ as a minor. This result also has a strong impact on algorithms, since it implies that testing for minor closed properties can be done in polynomial time, namely by finitely many calls to an $\mathcal{O}(n^3)$-time algorithm (introduced in [26]) checking whether the input graph G contains some graph H (called the *pattern*) as a minor. (Recently, Kawarabayashi *et al.* have presented a quadratic algorithm for this problem [21].) As a consequence, several graph problems have been shown to have polynomial-time algorithms, some of which were previously not even known to be decidable [16]. However, these algorithmic results are *non-constructive*. This triggered an ongoing quest in the Theory of Algorithms since then, next to the simplification of the the 23-papers proof of the Graph Minors Theorem, for extracting constructive algorithmic results out of Graph Minors (e.g., [2,4,7,22]) and for making its algorithmic proofs practical. Minor containment is one of the important steps in the technique of minor-closed property testing. Unfortunately the hidden constants in the polynomial-time algorithm of [26] are immense even for very simple patterns, which makes the algorithm absolutely impractical.

A basic algorithmic tool introduced in the Graph Minors series is *branchwidth*, which serves (together with its twin parameter of *treewidth*) as a measure of the topological resemblance of a graph to the structure of a tree. The algorithmic importance of branchwidth resides in Courcelle's theorem [3], stating that all graph problems expressible by some Monadic Second Order Logic (MSOL) formula ϕ can be solved in $f(\mathbf{bw}(G), \phi) \cdot n$ steps (we denote by $\mathbf{bw}(G)$ the branchwidth of the graph G). As minor checking (for fixed patterns) can be expressed in MSOL, we obtain the existence of a $f(k,h)\cdot|V(G)|$ step algorithm for the following (parameterized) problem (throughout the paper, we let $n = |V(G)|$, $m = |E(G)|$, and $h = |V(H)|$):

H-Minor Containment
Input: A graph G (the host graph).
Parameter: $k = \mathbf{bw}(G)$.
Question: Does G contain a minor isomorphic to H (the pattern graph)?

Such an algorithm is one of the basic subroutines required by the algorithm in [26] (that is, for the non-parameterized version of minor containment), and every attempt to improve its practicability requires the improvement of the parameter dependence $f(k,h)$. A significant step in this direction was done by Hicks [20], who provided an $\mathcal{O}(3^{k^2} \cdot (h+k-1)! \cdot m)$ step algorithm for H-Minor Containment, exploiting the ideas sketched by Robertson and Seymour in [26]. Note that when H is not fixed, determining whether G contains H as a minor is NP-complete even if G has bounded branchwidth [24].

The objective of this paper is to provide parameterized algorithms for the H-Minor Containment problem with better parameter dependence.

Our results. We present an algorithm for H-Minor Containment with running time $\mathcal{O}(2^{(2k+1)\cdot\log k}\cdot h^{2k}\cdot 2^{2h^2}\cdot m)$, where k is the branchwidth of G, which improves the bound that follows from [26] (explicitly described in [20]). When we restrict the host graph to be embeddable in a fixed surface, we provide an algorithm with running time $2^{\mathcal{O}(k)}\cdot h^{2k}\cdot 2^{\mathcal{O}(h)}\cdot n$. This is the first algorithm for H-Minor Containment with single-exponential dependence on branchwidth. Finally, we show how to modify our algorithm to explicitly find, within the same time bounds, a minor of H in G, as well as for solving some related problems, like induced minor or contraction containment.

Our techniques. We introduce a dynamic programming technique based on a combinatorial object called *rooted packing* (defined in Subsection 3.1). Rooted packings capture how potential models of H (defined in Section 2) are intersecting the separators that the algorithm is processing. It is worth mentioning here that the notion of rooted packing is related to the notion of *folio* introduced by Robertson and Seymour in [26], see Subsection 3.1 for more details. We present the algorithm for general host graphs in Subsection 3.2. When the host graph G is embedded in a surface (see Section 4), this formulation with rooted packings allows us to apply the framework introduced in [29] to obtain algorithms for H-Minor Containment that are single-exponential in k. In this framework we use a new type of branch decomposition, called *surface cut decomposition* (see Subsection 4.2 and [29]), which generalizes sphere cut decompositions for planar graphs introduced by Seymour and Thomas [31]. Our algorithms are robust, in the sense that slight variations permit us to solve several related problems within the same time bounds (see Section 5). Finally, we present some lines for further research in Section 6.

2 Definitions

Graphs and minors. We use standard graph terminology, see for instance [8]. All the graphs considered in this article are simple and undirected. Given a graph G, we denote the vertex set of G by $V(G)$ and the edge set of G by $E(G)$. A graph F is a subgraph of a graph G, denoted by $F\subseteq G$, if $V(F)\subseteq V(G)$ and $E(F)\subseteq E(G)$. For a subset $X\subseteq V(G)$, we use $G[X]$ to denote the subgraph of G *induced* by X, i.e., $V(G[X]) := X$ and $E(G[X]) := \{\{u,v\}\subseteq X \mid \{u,v\}\in E(G)\}$. For a subset $Y\subseteq E(G)$, we let $G[Y]$ be the graph with $V(G[Y]) := \{v\in V(G) \mid v\in e \text{ for some } e\in Y\}$ and $E(G[Y]) := Y$. *Hypergraphs* generalize graphs by allowing edges to be arbitrary subsets of the vertex set. Let $\mathcal{G}$ be a hypergraph. A *path* in $\mathcal{G}$ is a sequence $v_1,\ldots,v_n$ of vertices of $\mathcal{G}$, such that for every two consecutive vertices there exists a distinct hyperedge of $\mathcal{G}$ containing both. In this way, the notions of connectivity, connected component, etc. are transferred from graphs to hypergraphs. Given a subset $S\subseteq V(G)$, we define $N_G[S]$ to be the set of vertices of $V(G)$ at distance at most 1 from at least one vertex of S. If $S=\{v\}$, we simply use the notation $N_G[v]$. We also define $N_G(v)=N_G[v]\setminus\{v\}$ and $E_G(v)=\{\{v,u\} \mid u\in N_G(v)\}$. Let $e=\{x,y\}\in E(G)$. Given a graph G and an edge $e\in E(G)$, let $G\backslash e := (V(G),E(G)\setminus\{e\})$ be the graph obtained from G by deleting e, and let G/e be the graph obtained from G by *contracting* e, i.e.,

$$G/e = \big((V(G)\setminus\{x,y\})\,\dot\cup\,\{v_{x,y}\},(E(G)\setminus(E_G(x)\cup E_G(y)))\cup\{\{v_{xy},z\} \mid z\in N_G[\{x,y\}]\}\big),$$

where $v_{xy} \notin V(G)$ is a new vertex, not contained in $V(G)$. If H can be obtained from a subgraph of G by a (possibly empty) sequence of edge contractions, we say that H is a *minor* of G. If H can be obtained from an induced subgraph of G (resp. the whole graph G) by a (possibly empty) sequence of edge contractions, we say that H is an *induced minor* (resp. a *contraction*) of G.

Branch decompositions. A *branch decomposition* (T, μ) of a graph G consists of a ternary tree T (i.e., all internal vertices are of degree three) and a bijection $\mu : L \to E(G)$ from the set L of leaves of T to the edge set of G. We define for every edge e of T the *middle set* $\mathbf{mid}(e) \subseteq V(G)$ as follows: Let T_1 and T_2 be the two connected components of $T \setminus \{e\}$. Then let G_i be the graph induced by the edge set $\{\mu(f) : f \in L \cap V(T_i)\}$ for $i \in \{1, 2\}$. The *middle set* is the intersection of the vertex sets of G_1 and G_2, i.e., $\mathbf{mid}(e) = V(G_1) \cap V(G_2)$. Note that for each $e \in E(T)$, $\mathbf{mid}(e)$ is a separator of G (unless $\mathbf{mid}(e) = \emptyset$). The *width* of (T, μ) is the maximum order of the middle sets over all edges of T, i.e., $\mathit{width}(T, \mu) := \max\{|\mathbf{mid}(e)| : e \in E(T)\}$. The *branchwidth* of G is defined as $\mathbf{bw}(G) := \min\{\mathit{width}(T, \mu) \mid (T, \mu)$ branch decomposition of $G\}$. Intuitively, a graph has small branchwidth if it is close to being a tree.

In our algorithms, we need to root a branch decomposition (T, μ) of G. For this, we pick an arbitrary edge $e^* \in E(T)$, we subdivide it by adding a new vertex v_{new} and then add a new vertex r and make it adjacent to v_{new}. We extend μ by setting $\mu(r) = \emptyset$ (thereby slightly extending the definition of a branch decomposition). Now vertex r is the root. For each $e \in E(T)$, let T_e be the tree of the forest $T \backslash e$ that does not contain r as a leaf (i.e., the tree that is "below" e in the rooted tree T) and let E_e be the edges that are images, via μ, of the leaves of T that are also leaves of T_e. Let $G_e := G[E_e]$. Observe that, if $e_r = \{v_{\text{new}}, r\}$, then $G_{e_r} = G$ unless G has isolated vertices.

Models. A *model* of H in G [26] is a mapping ϕ, that assigns to every edge $e \in E(H)$ an edge $\phi(e) \in E(G)$, and to every vertex $v \in V(H)$ a non-empty connected subgraph $\phi(v) \subseteq G$, such that

(i) the graphs $\{\phi(v) \mid v \in V(H)\}$ are mutually vertex-disjoint and the edges $\{\phi(e) \mid e \in E(H)\}$ are pairwise distinct;
(ii) for $e = \{u, v\} \in E(H)$, $\phi(e)$ has one end-vertex in $V(\phi(u))$ and the other in $V(\phi(v))$.

Thus, H is isomorphic to a minor of G if and only if there exists a model of H in G.

Remark 1. We can assume that for each vertex $v \in V(H)$, the subgraph $\phi(v) \subseteq G$ is a tree. Indeed, if for some $v \in V(H)$, $\phi(v)$ is not a tree, then by replacing $\phi(v)$ with a spanning tree of $\phi(v)$ we obtain another model with the desired property.

For each $v \in V(H)$, we call the graph $\phi(v)$ a *vertex-model* of v. With slight abuse of notation, the subgraph $M \subseteq G$ defined by the union of $\{\phi(v) \mid v \in V(H)\}$ and $\{\phi(e) \mid e \in E(H)\}$ is also called a *model* of H in G. For each edge $e \in E(H)$, the edge $\phi(e) \in E(G)$ is called a *realization* of e.

Potential models. In the course of dynamic programming along a branch decomposition, we will need to search for potential models of subgraphs of H in G, which we proceed to define. For graphs $\bar{H}$ and G, a set $R \subseteq V(\bar{H})$, and a (possibly empty) set $X \subseteq E(\bar{H}[R])$, an (R, X)-*potential model* of

$\bar{H}$ in G is a mapping ϕ, that assigns to every edge $e \in (E(\bar{H}) \setminus E(\bar{H}[R])) \cup X$ an edge $\phi(e) \in E(G)$, and to every vertex $v \in V(\bar{H})$ a non-empty subgraph $\phi(v) \subseteq G$, such that

(i) the graphs $\{\phi(v) \mid v \in V(\bar{H})\}$ are mutually vertex-disjoint and the edges $\{\phi(e) \mid e \in E(\bar{H})\}$ are pairwise distinct;
(ii) for every $e = \{u, v\} \in (E(\bar{H}) \setminus E(\bar{H}[R])) \cup X$, the edge $\phi(e)$ has one end-vertex in $V(\phi(u))$ and the other in $V(\phi(v))$;
(iii) for every $v \in V(\bar{H}) \setminus R$ the graph $\phi(v)$ is connected in G.

For the sake of intuition, we can think of an (R, X)-potential model of $\bar{H}$ as a candidate of becoming a model of $\bar{H}$ in further steps of the dynamic programming, if the missing edges (that is, those in $E(\bar{H}[R]) \setminus X$) can be realized, and if the graphs $\{\phi(v) \mid v \in R\}$ get eventually connected.

We say that ϕ is an *R-potential model* of $\bar{H}$ in G, if ϕ is an (R, X)-potential model of $\bar{H}$ in G for some $X \subseteq E(\bar{H}[R])$, and we say that ϕ is a *potential model* of $\bar{H}$ in G, if ϕ is an *R-potential model* of $\bar{H}$ in G for some $R \subseteq V(\bar{H})$. Note that a $\emptyset$-potential model of $\bar{H}$ in G is a model of $\bar{H}$ in G. Slightly abusing notation, we also say that the subgraph $M \subseteq G$ defined by the union of $\{\phi(v) \mid v \in V(\bar{H})\}$ and $\{\phi(e) \mid e \in (E(\bar{H}) \setminus E(\bar{H}[R])) \cup X\}$ is an *(R, X)-potential model* of $\bar{H}$ in G.

3 Dynamic programming for general graphs

Roughly speaking, in each edge of the branch decomposition, the tables of our dynamic programming algorithm store all the potential models of H in the graph processed so far. While the vertex-models of H are required to be connected in G, in potential models they may have several connected components, and we need to keep track of them. In order to do so, we introduce *rooted packings* of the middle sets (defined in Subsection 3.1). A rooted packing encodes the trace of the components of a potential model in the middle set, together with a mapping of the components to vertices of H. We denote the *empty set* by $\emptyset$ and the *empty function* by $\varnothing$.

3.1 Rooted packings

Let $S \subseteq V(H)$ be a subset of the vertices of the pattern H, and let $R \subseteq S$. Given a middle set $\mathbf{mid}(e)$ corresponding to an edge e of a branch decomposition (T, μ) of G, we define a *rooted packing* of $\mathbf{mid}(e)$ as a quintuple $\mathbf{rp} = (\mathcal{A}, S, R, \psi, \chi)$, where $\mathcal{A}$ is a (possible empty) collection of mutually disjoint non-empty subsets of $\mathbf{mid}(e)$ (that is, a *packing* of $\mathbf{mid}(e)$), $\psi : \mathcal{A} \to R$ is a surjective mapping (the *rooting*) assigning vertices of R to the sets in $\mathcal{A}$, and $\chi : R \times R \to \{0, 1\}$ is a binary symmetric function between pairs of vertices in R.

The intended meaning of a rooted packing $(\mathcal{A}, S, R, \psi, \chi)$ is as follows. In a given middle set $\mathbf{mid}(e)$, a packing $\mathcal{A}$ represents the intersection of the connected components of the potential model with $\mathbf{mid}(e)$. The subsets $R, S \subseteq V(H)$ and the function χ indicate that we are looking for an $(R, \{\{u, v\} \mid u, v \in R, \chi(u, v) = 1\})$-potential model M of $H[S]$ in G_e. Intuitively, the function χ captures which edges of $H[S]$ have been realized so far. Since we allow the vertex-models intersecting $\mathbf{mid}(e)$ to be disconnected, we need to keep track of their connected components. The subset $R \subseteq S$ tells us which vertex-models intersect $\mathbf{mid}(e)$, and the function ψ associates the sets

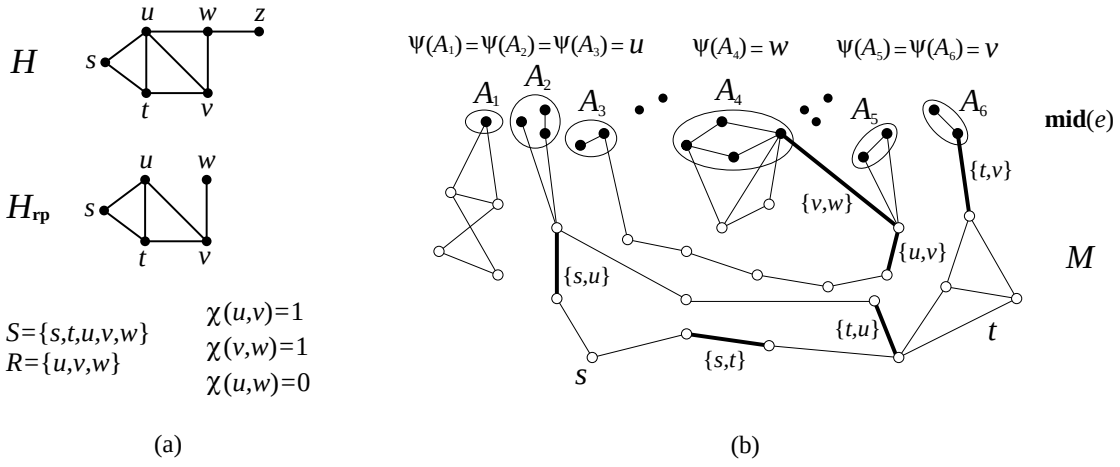

Fig. 1. (a) A pattern H and a subgraph $H_{\mathbf{rp}} \subseteq H$ associated with a rooted packing $\mathbf{rp} = (\mathcal{A}, S, R, \psi, \chi)$. We have $V(H) = \{s, t, u, v, w, z\}, S = \{s, t, u, v, w\}$, and $R = \{u, v, w\}$. The function χ is given by $\chi(u, v) = \chi(v, w) = 1$ and $\chi(u, w) = 0$, which defines the edges in $H_{\mathbf{rp}}$. (b) An R-potential model $M \subseteq G_e$ corresponding to the rooted packing $\mathbf{rp}$ of $\mathbf{mid}(e)$. Full dots represent vertices in $\mathbf{mid}(e)$, and the ovals indicate the subsets of the packing $\mathcal{A} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$. Above the ovals, the coloring ψ is shown. The thick edges in M correspond to realizations of edges in $E(H_{\mathbf{rp}})$, which are explicitly labeled in the figure. Note that the vertex-models in M corresponding to vertices $s, t \in S \setminus R$ are connected, as required.

in $\mathcal{A}$ with the vertices in R. We can think of ψ as a coloring that colors the subsets in $\mathcal{A}$ with colors given by the vertices in R. Note that several subsets in $\mathcal{A}$ can have the same color $u \in R$, which means that the vertex-model of u in G_e is not connected yet, but it may get connected in further steps of the dynamic programming, if the necessary edges appear from other branches of the branch decomposition of G. Note that we distinguish between two types of edges of $H[S]$, namely those with both end-vertices in R, and the rest. The key observation is that if the desired R-potential model of $H[S]$ exists, then all the edges in $E(H[S]) \setminus E(H[R])$ must have already been realized in G_e. Indeed, as $\mathbf{mid}(e)$ is a separator of G and no vertex-model of a vertex in $S \setminus R$ intersects $\mathbf{mid}(e)$, the edges in $E(H[S]) \setminus E(H[R])$ cannot appear in $G \setminus G_e$. Therefore, we make sure that the edges in $E(H[S]) \setminus E(H[R])$ have already been realized, and we only need to keep track of the edges in $E(H[R])$. In other words, for two distinct vertices $u, v \in R$, we let $\chi(u, v) = 1$ if and only if $\{u, v\} \in E(H)$ and there exist two subsets $A, B \in \mathcal{A}$, with $\psi(A) = u$ and $\psi(B) = v$, such that there is an edge in the potential model M between a vertex in A and a vertex in B. In that case, it means that we have a realization of the edge $\{u, v\} \in E(H)$ in $M \subseteq G_e$. A rooted packing $\mathbf{rp} = (\mathcal{A}, S, R, \psi, \chi)$ defines a unique subgraph $H_{\mathbf{rp}}$ of H, with $V(H_{\mathbf{rp}}) = S$ and $E(H_{\mathbf{rp}}) = E(H[S]) \setminus E(H[R]) \cup \{\{u, v\} \mid u, v \in R, \chi(u, v) = 1\}$. An example of the intended meaning of a rooted packing is illustrated in Fig. 1.

As mentioned in Section 1, the notion of rooted packing is related to the notion of folio introduced by Robertson and Seymour in [26] (see also [21]). More precisely, a graph G together with r vertices $v_1, \ldots, v_r \in V(G)$ is called a *rooted graph*. The *folio* of a rooted graph $(G, v_1, \ldots, v_r)$

is the set of all rooted graphs $(H, u_1, \ldots, u_r)$ such that there exists a model ϕ of H in G with $v_i \subseteq V(\phi(u_i))$ for $1 \leq i \leq r$. Using this terminology, given two rooted graphs $(G, v_1, \ldots, v_r)$ and $(H, u_1, \ldots, u_r)$, if we set $R = \{u_1, \ldots, u_r\}$ and we assume that $\{v_1, \ldots, v_r\} \subseteq \mathbf{mid}(e)$ for some edge e of a branch decomposition (T, μ) of G, a rooted packing $(\mathcal{A}, S, R, \psi, \chi)$ of $\mathbf{mid}(e)$ can be seen as a refined data structure that can be used to determine whether $(H, u_1, \ldots, u_r)$ belongs to the folio of $(G, v_1, \ldots, v_r)$. In the remainder of the article we will not use this interpretation.

In the sequel, it will be convenient to think of a packing $\mathcal{A}$ of $\mathbf{mid}(e)$ as a hypergraph $\mathcal{G} = (\mathbf{mid}(\mathrm{e}), \mathcal{A})$. Note that, by definition, $\mathcal{A}$ is a matching in $\mathcal{G}$. We use the notation $\bigcup \mathcal{A} := \bigcup_{X \in \mathcal{A}} X$.

Operations with rooted packings. Let $\mathbf{rp}_1 = (\mathcal{A}_1, S_1, R_1, \psi_1, \chi_1)$ and $\mathbf{rp}_2 = (\mathcal{A}_2, S_2, R_2, \psi_2, \chi_2)$ be rooted packings of two middle sets $\mathbf{mid}(e_1)$ and $\mathbf{mid}(e_2)$, such that e_1 and e_2 are the children edges of an edge $e \in E(T)$. We say that $\mathbf{rp}_1$ and $\mathbf{rp}_2$ are *compatible* if

(i) $E(H_{\mathbf{rp}_1}) \cap E(H_{\mathbf{rp}_2}) = \emptyset$;
(ii) $S_1 \cap S_2 = R_1 \cap R_2$;
(iii) for any $A_1 \in \mathcal{A}_1$ and $A_2 \in \mathcal{A}_2$ such that $A_1 \cap A_2 \neq \emptyset$, we have $\psi_1(A_1) = \psi_2(A_2)$.

In other words, two rooted packings $\mathbf{rp}_1$ and $\mathbf{rp}_2$ are compatible if the edge-sets of the corresponding subgraphs $H_{\mathbf{rp}_1}$ and $H_{\mathbf{rp}_2}$ are disjoint, if their intersection is given by the intersection of R_1 and R_2, and if their colorings coincide in the common part. Note that whether two rooted packings are compatible can be easily checked in time linear in the sizes of the middle sets.

Given two hypergraphs H_1 and H_2 of H, we define $H_1 \cup H_2$ as the graph with vertex set $V(H_1) \cup V(H_2)$ and edge set $E(H_1) \cup E(H_2)$. Given two compatible rooted packings $\mathbf{rp}_1 = (\mathcal{A}_1, S_1, R_1, \psi_1, \chi_1)$ and $\mathbf{rp}_2 = (\mathcal{A}_2, S_2, R_2, \psi_2, \chi_2)$, we define $\mathbf{rp}_1 \oplus \mathbf{rp}_2$ as the rooted packing $(\mathcal{A}, S, R, \psi, \chi)$, where

- $\mathcal{A}$ is the packing of $\mathbf{mid}(e)$ defined by the connected components of the hypergraph $(\mathbf{mid}(e_1) \cup \mathbf{mid}(e_2), \mathcal{A}_1 \cup \mathcal{A}_2)$. In other words, the sets of the packing $\mathcal{A}$ are the vertex sets corresponding to the connected components of the hypergraph $(\mathbf{mid}(e_1) \cup \mathbf{mid}(e_2), \mathcal{A}_1 \cup \mathcal{A}_2)$;
- $S = S_1 \cup S_2$;
- $R = R_1 \cup R_2$;
- for any subset $A \in \mathcal{A}$, $\psi(A)$ is defined as

$$\psi(A) = \begin{cases} \psi_1(A_1), & \text{if there exists } A_1 \in \mathcal{A}_1 \text{ such that } A \cap A_1 \neq \emptyset. \\ \psi_2(A_2), & \text{if there exists } A_2 \in \mathcal{A}_2 \text{ such that } A \cap A_2 \neq \emptyset. \end{cases}$$

  Note that the mapping ψ is well-defined. Indeed, if there exist both $A_1 \in \mathcal{A}_1$ and $A_2 \in \mathcal{A}_2$ intersecting a subset A, then by definition of $\mathcal{A}$ it holds $A_1 \cap A_2 \neq \emptyset$, and therefore $\psi_1(A_1) = \psi_2(A_2)$ because by assumption the rooted packings $\mathbf{rp}_1$ and $\mathbf{rp}_2$ are compatible;
- for any two vertices $u, v \in R$, $\chi(u, v)$ is defined as

$$\chi(u, v) = \begin{cases} 1, & \text{if either } u, v \in R_1 \text{ and } \chi_1(u, v) = 1, \text{ or } u, v \in R_2 \text{ and } \chi_2(u, v) = 1. \\ 0, & \text{otherwise.} \end{cases}$$

Note that if $\mathbf{rp}_1$ and $\mathbf{rp}_2$ are two compatible rooted packings, then $H_{\mathbf{rp}_1 \oplus \mathbf{rp}_2} = H_{\mathbf{rp}_1} \cup H_{\mathbf{rp}_2}$.

If $(\mathcal{A}, S, R, \psi, \chi)$ is a rooted packing of a middle set $\mathbf{mid}(e)$ and $B \subseteq \mathbf{mid}(e)$, we define $(\mathcal{A}, S, R, \psi, \chi)|_B$ as the rooted packing $(\mathcal{A}', S', R', \psi', \chi')$ of B, where

- $\mathcal{A}' = \{X \cap B \mid X \in \mathcal{A}\} \setminus \{\emptyset\}$;
- $S' = S$;
- for a set $X \cap B \in \mathcal{A}'$ with $X \in \mathcal{A}$ we let $\psi'(X \cap B) = \psi(X)$;
- R' is defined as the image of ψ', that is, $R' = \{\psi'(A) \mid A \in \mathcal{A}'\}$;
- χ' is defined at the restriction of χ to $R' \times R'$, that is, for two vertices $u, v \in R'$, $\chi'(u, v) = \chi(u, v)$.

Note that the property of being a rooted packing is closed under the two operations defined above.

How to encode a potential model. Let $\mathscr{P}_e$ be the collection of all rooted packings $(\mathcal{A}, S, R, \psi, \chi)$ of $\mathbf{mid}(e)$. We use the notation $\mathcal{C}(F)$ for the set of connected components of a graph (or hypergraph) F. Given a rooted packing $(\mathcal{A}, S, R, \psi, \chi) \in \mathscr{P}_e$, we define the boolean variable $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$, encoding whether G_e contains a potential model with the conditions given by the rooted packing. Namely, the variable is set to `true` if the required potential model exists, and to `false` otherwise:

$$\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi) = \begin{cases} \texttt{true}, & \text{if there exist a subgraph } M \subseteq G_e \text{ and a partition of} \\ & V(M) \text{ into } |S| \text{ sets } \{V_u \mid u \in S\} \text{ such that} \\ & \text{(i) for every } u \in S \setminus R, \quad |\mathcal{C}(M[V_u])| = 1 \text{ and } V_u \cap \mathbf{mid}(e) = \emptyset; \\ & \text{(ii) for every } u \in R, \\ & \quad \{V(M') \cap \mathbf{mid}(e) \mid M' \in \mathcal{C}(M[V_u])\} = \psi^{-1}(u); \\ & \text{(iii) for every two vertices } u, v \in S \text{ with } \{u, v\} \in E(H) \\ & \quad \text{and such that } \{u, v\} \not\subseteq R, \text{ there exist } u^* \in V_u \text{ and} \\ & \quad v^* \in V_v \text{ such that } \{u^*, v^*\} \in E(M); \\ & \text{(iv) for every two vertices } u, v \in R, \chi(u, v) = 1 \text{ if} \\ & \quad \text{and only if } \{u, v\} \in E(H) \text{ and there exist} \\ & \quad u^* \in V_u \text{ and } v^* \in V_v \text{ such that } \{u^*, v^*\} \in E(M). \\ \texttt{false}, & \text{otherwise.} \end{cases}$$

Note that since $\mathcal{A}$ does not contain the empty set, in (ii) we implicitly require every connected component of V_u to have a non-empty intersection with $\mathbf{mid}(e)$.

The following lemma follows immediately from the definitions.

Lemma 1. *Let G and H be graphs, let e be an edge in a rooted branch decomposition of G.*

1. *If $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi) = \texttt{true}$, then G_e contains an $(R, \{\{u, v\} \mid u, v \in R, \chi(u, v) = 1\})$-potential model of $H[S]$.*
2. *If G_e contains an R-potential model of $H[S]$, then there exist $\mathcal{A}$, ψ, and χ such that $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi) = \texttt{true}$.*

3. G contains a minor isomorphic to H if and only if some middle set $\mathbf{mid}(e)$ satisfies $\mathbf{mod}_e(\emptyset, V(H), \emptyset, \varnothing, \varnothing) = \texttt{true}$.

3.2 The algorithm

Let us now see how the values of $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$ can be explicitly computed using dynamic programming over a branch decomposition of G.

First, let e, e_1, e_2 be three edges of T that are incident to the same vertex and such that e is closer to the root of T than the other two. The value of $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$ is then given by:

$$\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi) = \begin{cases} \texttt{true}, & \text{if there exist two compatible rooted packings } (\mathcal{A}_1, S_1, R_1, \psi_1, \chi_1) \\ & \text{and } (\mathcal{A}_2, S_2, R_2, \psi_2, \chi_2) \text{ of } \mathbf{mid}(e_1) \text{ and } \mathbf{mid}(e_2), \text{ such that} \\ & \text{(i) } \mathbf{mod}_{e_1}(\mathcal{A}_1, S_1, R_1, \psi_1, \chi_1) = \mathbf{mod}_{e_2}(\mathcal{A}_2, S_2, R_2, \psi_2, \chi_2) = \texttt{true}; \\ & \text{(ii) } \bigcup \mathcal{A}_1 \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)) = \bigcup \mathcal{A}_2 \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2))\,; \\ & \text{(iii) } (\mathcal{A}, S, R, \psi, \chi) = (\mathcal{A}_1, S_1, R_1, \psi_1, \chi_1) \oplus (\mathcal{A}_2, S_2, R_2, \psi_2, \chi_2)|_{\mathbf{mid}(e)}; \\ & \text{(iv) Let } (\mathcal{A}', S, R', \psi', \chi') = (\mathcal{A}_1, S_1, R_1, \psi_1, \chi_1) \oplus (\mathcal{A}_2, S_2, R_2, \psi_2, \chi_2). \\ & \quad \text{Then, for each } u \in (R_1 \cup R_2) \setminus R, \ |\psi'^{-1}(u)| = 1. \\ \texttt{false}, & \text{otherwise.} \end{cases}$$

We have shown above how to compute $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$ for e being an internal edge of T. Finally, suppose that $e_{\text{leaf}} = \{x, y\} \in E(T)$ is an edge such that x is a leaf of T. Let $\mu(x) = \{v_1, v_2\} \in E(G)$, and let u and v be two arbitrary distinct vertices of H. Then

$$\mathbf{mod}_{e_{\text{leaf}}}(\mathcal{A}, S, R, \psi, \chi) = \begin{cases} \texttt{true}, & \text{if} \quad \mathcal{A} = \{\{v_1, v_2\}\},\ S = R = \{u\}, \\ & \qquad \psi(\{v_1, v_2\}) = u, \text{ and } \chi(u, u) = 0, \\ & \text{or} \quad \mathcal{A} = \{\{v_i\}\},\ S = \{u, v\},\ R = \{u\}, \\ & \qquad \psi(\{v_i\}) = u, \text{ and } \chi(u, u) = 0, \text{ for } i \in \{1, 2\}, \\ & \text{or} \quad \mathcal{A} = \{\{v_i\}\},\ S = R = \{u\}, \\ & \qquad \psi(\{v_i\}) = u, \text{ and } \chi(u, u) = 0, \text{ for } i \in \{1, 2\}, \\ & \text{or} \quad \mathcal{A} = \{\{v_1\}, \{v_2\}\},\ S = R = \{u, v\}, \\ & \qquad \psi(\{v_1\}) = u,\ \psi(\{v_2\}) = v,\ \chi(u, u) = \chi(v, v) = 0, \\ & \qquad \text{and } \chi(u, v) = \chi(v, u) = 1 \text{ only if } \{u, v\} \in E(H), \\ & \text{or} \quad \mathcal{A} = \emptyset,\ S = \{u, v\},\ R = \emptyset \text{ and } \psi = \chi = \varnothing, \\ & \text{or} \quad \mathcal{A} = \emptyset,\ S = \{u\},\ R = \emptyset \text{ and } \psi = \chi = \varnothing, \\ & \text{or} \quad \mathcal{A} = S = R = \emptyset \text{ and } \psi = \chi = \varnothing. \\ \texttt{false}, & \text{otherwise.} \end{cases}$$

Correctness of the algorithm. By Lemma 1, G contains a minor isomorphic to H if and only if for some middle set $\mathbf{mid}(e)$, $\mathbf{mod}_e(\emptyset, V(H), \emptyset, \varnothing, \varnothing) = \texttt{true}$. Observe that if $e_r = \{v_{\text{new}}, r\}$, we can assume that $\mathcal{A} = R = \emptyset$ and that $\psi = \chi = \varnothing$.

Given three edges e, e_1, e_2 as described above, we shall now see that the formula to compute $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$ is correct. Indeed, condition (i) guarantees that the required compatible models in G_{e_1} and G_{e_2} exist, while condition (ii) assures that the packings $\mathcal{A}_1$ and $\mathcal{A}_2$ contain the same vertices in the intersection of both middle sets. Condition (iii) says that the rooted packing of $\mathbf{mid}(e)$ can be obtained by first merging the two rooted packings of $\mathbf{mid}(e_1)$ and $\mathbf{mid}(e_2)$, and then projecting the obtained rooted packing to $\mathbf{mid}(e)$. Finally, condition (iv) imposes that each of the vertices in $R_1 \cup R_2$ that has been forgotten in $\mathbf{mid}(e)$ induces a single connected component in the desired potential model. This is indeed necessary, as the vertex-models of these forgotten vertices will not be updated anymore, so they need to be already connected. For each such vertex $u \in (R_1 \cup R_2) \setminus R$, the connectivity of the vertex-model of u is captured by the number of subsets colored u in the packing obtained by merging the packings $\mathcal{A}_1$ and $\mathcal{A}_2$. Indeed, the vertex-model of u is connected in M if and only if there is a single connected component colored u in the merged packing.

Suppose now that $e_{\text{leaf}} = \{x, y\} \in E(T)$ is a leaf-edge. Then $\mathbf{mid}(e_{\text{leaf}}) \subseteq \{v_1, v_2\}$ and $|S| \leq 2$. Let us discuss the formula to compute $\mathbf{mod}_{e_{\text{leaf}}}(\mathcal{A}, S, R, \psi, \chi)$. In the first case, $\mathcal{A} = \{\{v_1, v_2\}\}$, so both v_1 and v_2 must be mapped to the same vertex in S. The second and third case are similar, except that one of the two vertices v_1, v_2 is either not present in $\mathbf{mid}(e)$ or we omit it. In the fourth case we have $\mathcal{A} = \{\{v_1\}, \{v_2\}\}$, so each vertex in $\mathbf{mid}(e)$ corresponds to a distinct vertex of H, say, to u and v, respectively. We must distinguish two cases. Namely, if $\{u, v\} \in E(H)$, then the edge $\{v_1, v_2\} \in E(G)$ is a realization of $\{u, v\} \in E(H)$, so in this case we can set $\chi(u, v) = \chi(v, u) = 1$. Otherwise, we set $\chi(u, v) = \chi(v, u) = 0$. Finally, in the cases $\mathcal{A} = \emptyset$, we omit the whole middle set, and we set $R = \emptyset$.

Running time. The size of the tables of the dynamic programming over a branch decomposition of the input graph G determines the running time of our algorithms. For $e \in E(T)$, let $|\mathbf{mid}(e)| \leq k$, and let $h = |V(H)|$. To bound the size of the tables in e, namely $|\mathscr{P}_e|$, we discuss each element appearing in a rooted packing $(\mathcal{A}, S, R, \psi, \chi)$ of $\mathbf{mid}(e)$ separately:

- Bound on the number of $\mathcal{A}$'s: The number of ways a set of k elements can be partitioned into non-empty subsets is well-known as the k-th *Bell number* [17], and it is denoted by B_k. The number of packings of a set of k elements can be expressed in terms of the Bell numbers as

$$\sum_{i=0}^{k} \binom{k}{i} B_{k-i} \;=\; B_{k+1} \;\leq\; 2^{k \cdot \log k} \;,$$

  where the equality is a well-known recursive formula of the Bell numbers, and the inequality follows from $B_k \leq \frac{e^k - 1}{(\log k)^k} \cdot k!$ [17].
- Bound on the number of S's: the number of subsets of $V(H)$ is $2^{|V(H)|} = 2^h$.
- Bound on the number of R's: for a fixed $S \subseteq V(H)$, the number of subsets of S is at most 2^h.
- Bound on the number of ψ's: ψ is a mapping from subsets of $\mathbf{mid}(e)$ to vertices in R, so the number of such mappings for a fixed packing of $\mathbf{mid}(e)$ is at most h^k.
- Bound on the number of χ's: χ is a symmetric function from $R \times R$ to $\{0, 1\}$, so for a fixed R with $|R| \leq h$, the number of choices for χ is at most $2^{h^2/2}$.

Summarizing, for each edge $e \in E(T)$, we have that

$$|\mathscr{P}_e| \;\leq\; 2^{k\cdot\log k}\cdot h^k \cdot 2^{h^2/2}\cdot 2^{2h} \;\leq\; 2^{k\cdot\log k}\cdot h^k\cdot 2^{h^2}\;.$$

At each edge e of the branch decomposition, in order to compute all the values $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$, we test all the possibilities of combining compatible rooted packings of the two middle sets $\mathbf{mid}(e_1)$ and $\mathbf{mid}(e_2)$. The operations $(\mathcal{A}_1, S_1, R_1, \psi_1, \chi_1) \oplus (\mathcal{A}_2, S_2, R_2, \psi_2, \chi_2)$ and $(\mathcal{A}, S, R, \psi, \chi)|_B$ take $\mathcal{O}(|\mathbf{mid}(e)|)$ time, as well as testing whether two rooted packings are compatible. That is, these operations just incur a multiplicative term $\mathcal{O}(k) = \mathcal{O}(2^{\log k})$ in the running time. Hence, from the above discussion we conclude the following theorem.

Theorem 1. *Given a general host graph G with $|E(G)| = m$, a pattern H with $|V(H)| = h$, and a branch decomposition of G of width at most k, we can decide whether G contains a minor isomorphic to H in $\mathcal{O}(2^{(2k+1)\cdot\log k}\cdot h^{2k}\cdot 2^{2h^2}\cdot m)$ time.*

4 Speed up for graphs on surfaces

In this section we present a speed up of the algorithm described in Section 3 when the host graph G is embedded in a fixed surface Σ. Note that as the genus of a graph can only decrease by taking minors, we may assume that the pattern H can also be embedded in Σ. Namely, if $\mathbf{bw}(G) \leq k$, we improve the running time from $2^{\mathcal{O}(k\cdot\log k + h^2)+2k\cdot\log h}\cdot m$ (cf. Theorem 1) to $2^{\mathcal{O}(k+h)+2k\cdot\log h}\cdot n$ (cf. Theorem 4). That is, for fixed H, the running time of the algorithm becomes *single-exponential* in the branchwidth of the host graph. Let us briefly discuss this improved running time more in detail. If both G and H are planar, then it is known that if $\mathbf{bw}(G) > c\cdot h$ for some small constant c, then G contains H as a minor [19, 27]. Therefore, we may assume that $k = \mathcal{O}(h)$, and in that case there is no improvement in terms of k in Theorem 4 with respect to Theorem 1 (as we discuss in Subsection 4.3, the improvement in the term $2^{\mathcal{O}(h^2)}$ follows immediately from Euler's formula). The real improvement is when the host graph and the pattern are not planar, that is, when $\Sigma \neq \mathbb{S}^2$. In this case there is no bound on $\mathbf{bw}(G)$ to assure the existence of H as a minor, as shown by the following example. Let G be the disjoint union of an arbitrarily big grid and an arbitrary number of disjoint K_5's, and let $H = K_{3,3}$. Then G has arbitrarily big branchwidth and arbitrarily big genus, but it does not contain H as a minor. That is, even if G has grid-minors of size $\Omega(\mathbf{bw}(G))$ [6], a big grid does not certify a minor isomorphic to H. Therefore, no assumption can be made about k with respect to h, so the improvement given by Theorem 4 is indeed significant when $\Sigma \neq \mathbb{S}^2$.

We do not modify at all the dynamic programming algorithm presented in Section 3. Instead, our approach consists in analyzing more carefully its running time when G and H are embedded in a surface. The key idea is to use a special type of branch decomposition called *surface cut decomposition*, which has been recently introduced by Rué *et al.* [29], generalizing sphere cut decompositions of planar graphs (defined by Seymour and Thomas [31] and exploited algorithmically for doing dynamic programming for the first time in Dorn *et al.* [14]) to arbitrary surfaces. We first provide some preliminaries in Subsection 4.1, then we define surface cut decompositions in Subsection 4.2, and finally we present the improved analysis in Subsection 4.3.

4.1 Preliminaries

According to the Surface Classification Theorem [25], a compact and connected surface without boundary is determined, up to homeomorphism, by its Euler characteristic $\chi(\Sigma)$ and by whether it is orientable or not. More precisely, orientable surfaces are obtained by adding $g \geq 0$ *handles* to the sphere $\mathbb{S}^2$, obtaining the g-torus $\mathbb{T}_g$ with Euler characteristic $\chi(\mathbb{T}_g) = 2-2g$, while non-orientable surfaces are obtained by adding $c > 0$ *cross-caps* to the sphere, hence obtaining a non-orientable surface $\mathbb{P}_c$ with Euler characteristic $\chi(\mathbb{P}_c) = 2-c$. For computational simplicity, it is convenient to work with the *Euler genus* $\gamma(\Sigma)$ of a surface Σ, which is defined as $\gamma(\Sigma) = 2-\chi(\Sigma)$.

For a graph G, the *Euler genus* of G, denoted by $\gamma(G)$, is the smallest Euler genus among all surfaces in which G can be embedded (i.e., drawn without edge-crossings). Determining the Euler genus of a graph is an NP-hard problem [32], hence we assume that we are given a graph G already embedded in a surface Σ. An *O-arc* is a subset of Σ homeomorphic to $\mathbb{S}^1$. A subset of Σ meeting the drawing of G only at vertices is called *G-normal*. If an O-arc is G-normal, then we call it a *noose*. The *length* of a noose is the number of its vertices.

A *sphere cut decomposition* (T, μ) of a planar graph G is a branch decomposition of G with the following property: for every edge e of T, there exists a noose O_e meeting every face at most once and bounding the two open discs Δ_1 and Δ_2 such that $G_i \subseteq \Delta_i \cup O_e$, $1 \leq i \leq 2$. Thus O_e meets G only in $\mathbf{mid}(e)$ and its length is $|\mathbf{mid}(e)|$. Sphere cut decompositions were defined by Seymour and Thomas [31], and during the last years they have been exhaustively exploited to obtain algorithms in planar graphs with single-exponential dependence on branchwidth, which can then be used to obtain subexponential parameterized algorithms through Bidimensionality Theory [9, 13, 14, 30]. The key observation is that in a planar graph the restriction of a partial solution to a middle set has a *non-crossing* structure, and then the size of the tables can be upper-bounded by the *Catalan numbers* [17].

4.2 Surface cut decompositions

The approach based on exploiting sphere cut decompositions has been also extended to devise single-exponential algorithms for graphs of bounded genus and graphs excluding a fixed graph as a minor [5, 11, 12]. Roughly speaking, the idea for bounded genus graphs was to perform a *planarization* of the input graph by splitting the potential solutions into at most γ pieces and then applying the sphere cut decomposition technique. Two drawbacks of this approach are that the techniques are problem-dependent and that they are difficult to apply to the general class of problems in which a solution is encoded by a packing of vertices. Recall that a *packing* of a set X is a (possible empty) collection of mutually disjoint non-empty subsets of X.

Very recently, Rué *et al.* [29] have introduced a framework that allows to obtain single-exponential algorithms for a broad class of problems in graphs on surfaces. The main ingredient of this framework is a new type of branch decomposition called surface cut decomposition, which we proceed to define overlooking some technicalities (see [29] for the full definition). A *surface cut decomposition* of a graph G embedded in a surface Σ with Euler genus γ is consists of a branch decomposition (T, μ) of G and a subset $A \subseteq V(G)$, with $|A| = \mathcal{O}(\gamma)$, such that for each $e \in E(T)$,

- either $|\mathbf{mid}(e) \setminus A| \leq 2$,
- or
  - the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $\mathcal{O}(\gamma)$ nooses;
  - these nooses intersect in $\mathcal{O}(\gamma)$ vertices;
  - $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ contains exactly two connected components.

Note that a sphere cut decomposition is a particular case of a surface cut decomposition when G is planar, by taking $A = \emptyset$ and $|\mathcal{N}| = 1$ for each $e \in E(T)$. The importance of surface cut decompositions follows by the following two theorems, which have been proved using techniques from topological graph theory and analytic combinatorics.

Theorem 2 (Rué *et al.* [29]). *Given a graph G on n vertices embedded in a surface of Euler genus γ, with $\mathbf{bw}(G) \leq k$, a surface cut decomposition (T, μ) of G of width at most $27k + \mathcal{O}(\gamma)$ can be constructed in $2^{3k+\mathcal{O}(\log k)} \cdot n^3$ time.*

Theorem 3 (Rué *et al.* [29]). *Given a surface cut decomposition (T, μ) of width at most k of a graph G on n vertices embedded in a surface of Euler genus γ, for each $e \in E(T)$ the number of non-crossing packings of $\mathbf{mid}(e)$ is bounded above by $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\gamma)} \cdot \gamma^{\mathcal{O}(\gamma)}$.*

4.3 Improved analysis

When the host graph G is embedded in a surface Σ of Euler genus γ, we apply Theorem 2 to construct a surface cut decomposition (T, μ) of G of width $\mathcal{O}(\mathbf{bw} + \gamma)$, and then we run the algorithm presented in Section 3.2 on (T, μ). For $e \in E(T)$, let $|\mathbf{mid}(e)| \leq k$, and let $h = |V(H)|$. We discuss again each element appearing in a rooted packing $(\mathcal{A}, S, R, \psi, \chi)$ of $\mathbf{mid}(e)$ separately:

- Bound on the number of $\mathcal{A}$'s: since (T, μ) is a surface cut decomposition, by Theorem 3 the number of packings of $\mathbf{mid}(e)$ associated with a potential model is bounded by $2^{\mathcal{O}(k)} \cdot k^{\mathcal{O}(\gamma)} \cdot \gamma^{\mathcal{O}(\gamma)}$.
- Bound on the number of S's and R's: for both sets, the bound 2^h still holds.
- Bound on the number of ψ's: this bound also remains unchanged, that is, h^k.
- Bound on the number of χ's: the number of choices for χ is bounded by the number of subsets of $E(H[R])$, and since we can assume that H can also be embedded in Σ, it follows by Euler's formula that $|E(H[R])| = \mathcal{O}(h + \gamma)$, so the number of choices for χ is at most $2^{\mathcal{O}(h+\gamma)}$.

It also follows from Euler's formula that $|E(G)| = \mathcal{O}(n + \gamma)$. If we consider that the surface Σ is fixed, then $\gamma = \mathcal{O}(1)$, and from the above analysis we conclude the following theorem.

Theorem 4. *Given a host graph G with $|V(G)| = n$ embedded in a fixed surface, a pattern H with $|V(H)| = h$, and a surface cut decomposition of G of width at most k, we can decide whether G contains a minor isomorphic to H in $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$ time.*

5 Variations

In this section we show how to modify the algorithm presented in Section 3 in order to solve several variants of the H-Minor Containment problem. In all cases, the same times bounds given by Theorem 1 (general host graph) and Theorem 4 (host graph embedded in a surface) still hold.

Induced minor. It is natural to ask whether G contains H as an induced minor (see Section 2 for the definition). In contrast to the dynamic programming presented in Section 3.2, now we must only consider the rooted packings of $\mathbf{mid}(e)$ that define an *induced* potential model in G. Namely, if two adjacent vertices $v_1, v_2 \in V(G)$ belong to two different potential vertex-models of two vertices $u, v \in V(H)$, then the edge $\{u, v\}$ must also belong to the partial minor. This property can be incorporated in the algorithm of Subsection 3.2 by just imposing it in the leaves of the branch decomposition. Namely, if in a leaf corresponding to an edge $\{v_1, v_2\} \in E(G)$ we forbid the rooted packings given by $\mathcal{A} = \{\{v_1\}, \{v_2\}\}$, $S = R = \{u, v\}$, and $\psi(\{v_1\}) = u$, $\psi(\{v_2\}) = v$ with $\chi(u, v) = \chi(v, u) = 0$, then all the potential models will correspond to induced minors. Indeed, since we only merge potential models containing the same vertices in the intersection of two middle sets (see condition (ii) in the computation of $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$ in Subsection 3.2), the property of being an induced model propagates to all the edges of the branch decomposition.

Contraction minor. We now consider the problem of deciding whether H can be obtained from G by just doing edge contractions. That is, we cannot *forget* any vertex in the middle sets. In other words, in a rooted packing $(\mathcal{A}, S, R, \psi, \chi)$ of a middle set $\mathbf{mid}(e)$, $\mathcal{A}$ is restricted to be a *partition* of $\mathbf{mid}(e)$. We also have to assure that no edge of G is forgotten. This can be incorporated in the definition of $\mathbf{mod}_e(\mathcal{A}, S, R, \psi, \chi)$ in Subection 3.1 by replacing the potential model M with the graph G_e itself, and by adding a fifth condition:

(v) for every edge $\{v_1, v_2\} \in E(G_e)$, either both $v_1, v_2 \in V_u$ for some $u \in V(H)$, or $v_1 \in V_u$ and $v_2 \in V_v$ with $\{u, v\} \in E(H)$.

Finally, since all vertices and edges and G must be in the model, the value in the leaves is redefined as

$$\mathbf{mod}_{e_{\text{leaf}}}(\mathcal{A}, S, R, \psi, \chi) = \begin{cases} \texttt{true}, & \text{if} \quad \mathcal{A} = \{\{v_1, v_2\}\},\ S = R = \{u\}, \\ & \qquad \psi(\{v_1, v_2\}) = u, \text{ and } \chi(u, u) = 0, \\ & \text{or} \quad \mathcal{A} = \{\{v_1\}, \{v_2\}\},\ S = R = \{u, v\}, \\ & \qquad \psi(\{v_1\}) = u,\ \psi(\{v_2\}) = v,\ \chi(u, u) = \chi(v, v) = 0, \\ & \qquad \chi(u, v) = \chi(v, u) = 1, \text{ and } \{u, v\} \in E(H). \\ & \text{or} \quad \mathcal{A} = \{\{v_i\}\},\ S = \{u, v\},\ R = \{u\}, \\ & \qquad \psi(\{v_i\}) = u, \text{ and } \chi(u, u) = 0, \text{ for } i \in \{1, 2\}, \\ & \text{or} \quad \mathcal{A} = \{\{v_i\}\},\ S = R = \{u\}, \\ & \qquad \psi(\{v_i\}) = u, \text{ and } \chi(u, u) = 0, \text{ for } i \in \{1, 2\}, \\ & \text{or} \quad \mathcal{A} = S = R = \emptyset \text{ and } \psi = \chi = \varnothing. \\ \\ \texttt{false}, & \text{otherwise.} \end{cases}$$

Note that as $\mathcal{A}$ is restricted to be a partition of $\mathbf{mid}(e_{\text{leaf}})$, the third and fourth (resp. fifth) case can hold only if $|\mathbf{mid}(e_{\text{leaf}})| = 1$ (resp. $\mathbf{mid}(e_{\text{leaf}}) = \emptyset$).

Explicitly finding a model. In the case where G contains H as a minor, it is also interesting to explicitly find a model of H in G. The dynamic programming presented in Subsection 3.2 can

be easily made constructive using standard techniques: we keep pointers between table entries and if for some $\mathbf{mid}(e) \in E(T)$ we have $\mathbf{mod}_e(\emptyset, V(H), \emptyset, \varnothing, \varnothing) = \texttt{true}$, we post-process the tree top-down, marking the vertices that appear in the packings of the middle sets. In the leaves we restore the edges of G in the model.

It is also possible to *count* the number of models of H in G by just using a counter without increasing the running time, and applying the techniques of [15] to avoid double counting.

Finding a model of smallest size. Another natural variation of the H-Minor Containment problem is to find a model M of H in G, if it exists, of smallest size. The *size* of M can be taken as $|V(M)|$ or $|E(M)|$, but since by Remark 1 we can assume that each vertex-model is a tree, both versions are equivalent. In order to control the size of the model, we redefine the variable $\mathbf{mod}_e(\mathbf{rp})$ associated with a middle set $\mathbf{mid}(e)$ and a rooted packing $\mathbf{rp}$. Instead of being a boolean variable, in this case $\mathbf{mod}_e(\mathbf{rp})$ stores the smallest size of a potential model in G_e with the conditions given by $\mathbf{rp}$, and it is set to -1 if such a model does not exist. When updating the tables in an edge e, we check all possibilities of merging potential models of the children edges e_1 and e_2, and we keep a `true` option of smallest size. If a model of H is found, we do not stop the algorithm until the root, as a smaller model may still be found. See [30] for a detailed application of similar techniques to degree-constrained subgraph problems.

6 Conclusions and further research

In this paper we presented an algorithm to test whether an input host graph G contains a fixed graph H as a minor. Parameterizing the problem by the branchwidth of G ($\mathbf{bw}$), we improved the best existing algorithm for general host graphs, and we provided the first algorithm with running time single-exponential in $\mathbf{bw}$ when the host graph can be embedded in a surface. Finally, we showed how to modify our algorithm to solve some related problems, like induced minor or contraction containment.

There are a number of interesting lines for further research concerning minor containment problems. First of all, it may be possible to improve the dependence on $h = |V(H)|$ of our algorithms; see [1] for recent advances in this direction for the case when the host graph G is planar, using a different approach. On the other hand, we believe that the dependence on $\mathbf{bw}$ of our algorithm for general host graphs (that is, $2^{\mathcal{O}(\mathbf{bw} \cdot \log \mathbf{bw})}$) is best possible. The recent techniques presented by Lokshtanov *et al.* [23] may provide an answer to this question.

We also believe that the approach we used in Section 4 to obtain single-exponential algorithms when G is embedded in a surface can be extended to more general classes of graphs, like apex-minor-free graphs or general minor-free graphs. In order to do so, a first step could be to generalize the framework developed in [29] to minor-free graphs, which looks like a promising (but highly non-trivial) direction.

It makes sense to consider other parameters, like the size of the desired model of H in G, or the number of vertices one has to remove from G such that the resulting graph does not contain H as a minor. A more ambitious optimization version is to find a subgraph $F \subseteq G$ of the largest size such that F does not contain H as a minor. The dynamic programming approach that we presented seems to get considerably more involved in this case.

Recently, the first steps have been made towards an analog of Bidimensionality Theory for directed graphs [10]. It would be interesting to see whether our dynamic programming algorithms can be adapted to find directed minors in directed graphs.

Finally, a challenging problem concerning minor containment is to provide explicit and hopefully not too big constants depending on h in the polynomial-time algorithm of Robertson and Seymour [26]. Of course, these constants must be superpolynomial in h unless P = NP, as when H is not fixed the problem of deciding whether G contains H as a minor is NP-complete [18].

References

1. I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos. Fast Minor Testing in Planar Graphs. In *Proc. of the 18th Annual European Symposium on Algorithms (ESA)*, 2010. To appear.
2. I. Adler, M. Grohe, and S. Kreutzer. Computing excluded minors. In *Proc. of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 641–650, 2008.
3. B. Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, pages 193–242. 1990.
4. A. Dawar, M. Grohe, and S. Kreutzer. Locally Excluding a Minor. In *Proc. of the 22nd IEEE Symposium on Logic in Computer Science (LICS)*, pages 270–279, 2007.
5. E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.
6. E. D. Demaine and M. T. Hajiaghayi. Graphs excluding a fixed minor have grids as large as treewidth, with combinatorial and algorithmic applications through bidimensionality. In *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 682–689, 2005.
7. E. D. Demaine, M. T. Hajiaghayi, and K.-i. Kawarabayashi. Algorithmic Graph Minor Theory: Decomposition, Approximation, and Coloring. In *Proc. of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 637–646, 2005.
8. R. Diestel. *Graph Theory*, volume 173. Springer-Verlag, 2005.
9. F. Dorn. Planar Subgraph Isomorphism Revisited. In *Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 263–274, 2010.
10. F. Dorn, F. V. Fomin, D. Lokshtanov, V. Raman, and S. Saurabh. Beyond Bidimensionality: Parameterized Subexponential Algorithms on Directed Graphs. In *Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 251–262, 2010.
11. F. Dorn, F. V. Fomin, and D. M. Thilikos. Fast Subexponential Algorithm for Non-local Problems on Graphs of Bounded Genus. In *Proc. of the 10th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 4059 of *LNCS*, pages 172–183, 2006.
12. F. Dorn, F. V. Fomin, and D. M. Thilikos. Catalan structures and dynamic programming in H-minor-free graphs. In *Proc. of the 19th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 631–640, 2008.
13. F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008.
14. F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut branch decompositions. *Algorithmica*, 58(3):790–810, 2010.
15. D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999.

16. M. R. Fellows and M. A. Langston. Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM*, 35(3):727–739, 1988.
17. P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.
18. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
19. Q. P. Gu and H. Tamaki. Improved bound on the planar branchwidth with respect to the largest grid minor size. Technical Report SFU-CMPT-TR 2009-17, Simon Fraiser University, 2009.
20. I. V. Hicks. Branch decompositions and minor containment. *Networks*, 43(1):1–9, 2004.
21. K.-i. Kawarabayashi, Y. Kobayashi, and B. A. Reed. The disjoint paths problem in quadratic time. Submitted for publication, available at `http://research.nii.ac.jp/~k_keniti/quaddp1.pdf`, 2010.
22. K.-i. Kawarabayashi and B. A. Reed. Hadwiger's conjecture is decidable. In *Proc. of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 445–454, 2009.
23. D. Lokshtanov, D. Marx, and S. Saurabh. Slightly Superexponential Parameterized Problems. In *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 760–776, 2011.
24. J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics*, 108:143–364, 1992.
25. B. Mohar and C. Thomassen. *Graphs on surfaces*. John Hopkins University Press, 2001.
26. N. Robertson and P. Seymour. Graph Minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.
27. N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994.
28. N. Robertson and P. D. Seymour. Graph Minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.
29. J. Rué, I. Sau, and D. M. Thilikos. Dynamic Programming for Graphs on Surfaces. In *Proc. of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6198 of *LNCS*, pages 372–383, 2010.
30. I. Sau and D. M. Thilikos. Subexponential Parameterized Algorithms for Degree-constrained Subgraph Problems on Planar Graphs. *Journal of Discrete Algorithms*, 8(3):330–338, 2010.
31. P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.
32. C. Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989.