Dynamic Programming for Graphs on Surfaces*

Juanjo Rué† Ignasi Sau‡ Dimitrios M. Thilikos§

Abstract

We provide a framework for the design and analysis of dynamic programming algorithms for surface-embedded graphs on n vertices and branchwidth at most k. Our technique applies to general families of problems where standard dynamic programming runs in $2^{O(k\cdot\log k)}\cdot n$ steps. Our approach combines tools from topological graph theory and analytic combinatorics. In particular, we introduce a new type of branch decomposition called *surface cut decomposition*, generalizing sphere cut decompositions of planar graphs introduced by Seymour and Thomas, which has nice combinatorial properties. Namely, the number of partial solutions that can be arranged on a surface cut decomposition can be upper-bounded by the number of non-crossing partitions on surfaces with boundary. It follows that partial solutions can be represented by a single-exponential (in the branchwidth k) number of configurations. This proves that, when applied on surface cut decompositions, dynamic programming runs in $2^{O(k)}\cdot n$ steps. That way, we considerably extend the class of problems that can be solved in running times with a *single-exponential dependence* on branchwidth and unify/improve most previous results in this direction.

Keywords: analysis of algorithms; parameterized algorithms; graphs on surfaces; branchwidth; dynamic programming; polyhedral embeddings; non-crossing partitions.

1 Introduction

One of the most important parameters in the design and analysis of graph algorithms is the branchwidth of a graph. Branchwidth, together with its twin parameter of treewidth, can be seen as a measure of the topological resemblance of a graph to a tree. Its algorithmic importance dates back in the celebrated theorem of Courcelle (see e.g. [8]), stating that graph problems expressible in Monadic Second Order Logic can be solved in $f(\mathbf{bw})\cdot n$ steps

*The results of this paper were announced in the extended abstract "*Dynamic Programming for Graphs on Surfaces. Proceedings of ICALP'2010, volume 6198 of LNCS, pages 372-383*", which is a combination of the algorithmic framework presented in this paper and the enumerative results that can be found in [33].

†Laboratorie d'Informatique, École Polytechnique, 91128 Palaiseau-Cedex, France. Supported by the European Research Council under the European Community's 7th Framework Programme, ERC grant agreement 208471 - ExploreMaps project. E-mail: `rue1982@lix.polytechnique.fr`.

‡AlGCo project-team, CNRS, Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), Montpellier, France. Supported by projects ANR Agape and ANR Gratos. E-mail: `ignasi.sau@lirmm.fr`.

§Department of Mathematics, National and Kapodistrian University of Athens, Greece. Supported by the project "Kapodistrias" (ΑΠ 02839/28.07.2008) of the National and Kapodistrian University of Athens. E-mail: `sedthilk@math.uoa.gr`.

(here **bw** is the branchwidth[1] and n is the number of vertices of the input graph). Using parameterized complexity terminology, this implies that a large number of graph problems are fixed-parameter tractable when parameterized by the branchwidth of their input graph. As the bounds for $f(\mathbf{bw})$ provided by Courcelle's theorem are huge, the design of tailor-made dynamic programming algorithms for specific problems so that $f(\mathbf{bw})$ is a simple – preferably a *single-exponential* – function, became a natural (and unavoidable) ingredient for many results on graph algorithms (see [3,5,14,37]). In this paper, we provide a general framework for the design and analysis of dynamic programming algorithms for graphs embedded in surfaces where $f(\mathbf{bw}) = 2^{O(\mathbf{bw})}$.

Dynamic programming. Dynamic programming is applied in a bottom-up fashion on a rooted branch decomposition the input graph G, that roughly is a way to decompose the graph into a tree structure of edge bipartitions (the formal definition is in Section 2). Each bipartition defines a separator S of the graph called *middle set*, of cardinality bounded by the branchwidth of the input graph. The decomposition is routed in the sense that one of the parts of each bipartition is the "lower part of the middle set", i.e., the so-far processed one. For each graph problem, dynamic programming requires the suitable definition of tables encoding how potential (global) solutions of the problem are restricted to a middle set and the corresponding lower part. The size of these tables reflects the dependence on $k = |S|$ in the running time of the dynamic programming.

Designing the tables for each middle set S is not always an easy task and may vary considerably due to the particularities of each problem. The simplest cases are problems such as Vertex Cover and Dominating Set, where the certificate of the solution is a set of vertices whose choice is not restricted by some global condition. This directly yields the desired $2^{O(k)}$ upper bound on their size. For other problems, such as Longest Path, Cycle Packing, or Hamiltonian Cycle, things are more complicated as the tables encode *pairings of vertices of* S, which are $2^{\Theta(k \log k)}$ many. However, for such problems one can do better for *planar graphs* following the approach introduced in [16]. The idea in [16] is to use a special type of branch decomposition called *sphere cut decomposition* (introduced in [36]) that can guarantee that the pairings are *non-crossing* pairings around a virtual edge-avoiding cycle (called *noose*) of the plane where G is embedded. This restricts the number of tables corresponding to a middle set S by the k-th Catalan number, which is *single-exponential* in k. The same approach was extended for graphs embedded in surfaces of genus γ [13]. The idea in [13] was to perform a *planarization* of the input graph by splitting the potential solution into at most γ pieces and then applying the sphere cut decomposition technique of [16] to a more general version of the problem where the number of pairings is still bounded by some Catalan number (see also [15] for the application of this technique for more general graphs).

A wider family of problems are those where the tables of dynamic programming encode *connected packings* of S into sets, i.e., collections of subsets of S that are pairwise disjoint and where each subset is a connected part of a partial solution (see Section 3 for the formal definitions). Throughout this paper, we call these problems *connected packing-*

[1]The original statement of Courcelle's theorem used the parameter of treewidth instead of branchwidth. The two parameters are approximately equivalent, in the sense that one is a constant-factor approximation of the other.

encodable. Typical problems of this type are Connected Vertex Cover, Connected Dominating Set, Feedback Vertex Set (FVS), or Steiner Tree, where the connected components of a potential solution can be encoded by a collection of disjoint subsets of S, each of *arbitrary cardinality*. Here, the general bound on the table size is given by the k-th Bell number, and thus it can again be $2^{\Theta(k \cdot \log k)}$. To exemplify the differences between distinct types of dynamic programming encodings, we accompany this paper with an Appendix are presented (an expert reader may safely skip these examples). Unfortunately, for the latter category of problems, none of the current techniques has been able to drop the $2^{\Theta(k \cdot \log k)}$ bound to a single-exponential one for graphs embedded in surfaces. It is worth mentioning that, according to the recent lower bounds given by Lokshtanov *et al.* [27], the bound $2^{\Theta(k \cdot \log k)}$ is best possible in *general graphs* for some parameterized problems like Disjoint Paths, unless the Exponential Time Hypothesis (ETH) fails.

Our results. In this paper, we follow a different approach in order to design single-exponential (in **bw**) algorithms for graphs embedded in surfaces. In particular, we deviate significantly from the planarization technique of [13], which is not able to tackle problems whose solutions are encoded by general packings. Instead, we extend the concept of sphere cut decomposition from planar graphs to generic surfaces, and we exploit directly the combinatorial structure of the potential solutions in the topological surface. Our approach permits us to provide in a unified way a single-exponential (in **bw**) time analysis for all aforementioned problems. Examples of other such problems are Maximum Leaf Spanning Tree, Maximum Full-Degree Spanning Tree, Maximum Leaf Tree, Maximum d-Degree-Bounded Connected Subgraph, Metric TSP, or Maximum d-Degree-Bounded Connected Induced Subgraph and all the variants studied in [35]. Our results are formally described in Section 3 and imply all the results in [13, 16], with running times whose dependence on genus is better than the ones in [13], as discussed in Section 9.

Our techniques. For our results we enhance the current technology of dynamic programming using, among others, tools from topological graph theory. Our goal is to define a special type of branch decomposition of embedded graphs with nice topological properties, which we call *surface cut decomposition*. Moreover, we prove that such decomposition can be constructed in single-exponential time. Surface cut decompositions are based on the concept of *polyhedral decomposition*, which can be constructed in polynomial time. In the middle sets of a surface cut decomposition, all vertices, except possibly a set of cardinality $O(\gamma)$, are situated along a set of $O(\gamma)$ nooses of the surface with $O(\gamma)$ common points. This topological property of the middle sets is the source of the single-exponentiality of the size of the tables in dynamic programming: they correspond to non-crossing packings of a set where all its vertices, except possibly a set of cardinality $O(\gamma)$, lie on the boundary of a surface. Our next step is to reduce the problem of counting such packings to the counting of non-crossing partitions of vertices on the boundary of the same surface. Then, the single-exponential bound follows by the recent enumerative results of [33].

For performing dynamic programming, our approach resides in a common preprocessing step that is to construct a *surface cut decomposition*. Then, what remains is just to run a problem-specific dynamic programming algorithm on such a decomposition. The

exponential bound on the size of the tables of the dynamic programming algorithm follows as a result of the enumeration analysis in Section 8.

Very recently, a new framework for obtaining *randomized* single-exponential algorithms parameterized by treewidth in general graphs has appeared in [9]. This framework is based on a dynamic programming technique named Cut&Count, which seems applicable to most connected packing-encodable problems, like Connected Vertex Cover, Connected Dominating Set, Feedback Vertex Set, or Steiner Tree. The randomization in the algorithms of [9] comes from the usage a probabilistic result called the Isolation Lemma [30], whose derandomization is a challenging open problem [4]. Therefore, the existence of *deterministic* single-exponential algorithms parameterized by treewidth for connected packing-encodable problems in general graphs remains wide open. Our results for graphs on surfaces, as well as their generalization to any proper minor-free graph family [34], can be seen as an intermediate step towards an eventual positive answer to this question.

Organization of the paper. In Section 2, we give the definitions of the main topological and graph theoretical concepts and tools that we use in this paper. In Section 3, we define formally the class of connected packing-encodable problems and we formally settle the combinatorial problem of their enumeration. In Section 4, we define the concept of a polyhedral decomposition. In section 5, we give some results on the behavior of certain width parameters on surfaces and in Section 6, we prove some graph-topological results. The concept of a surface-cut decompositions, as well as the algorithm for its construction, are given in Section 7. The enumeration results of the paper are presented in Section 8. Finally, some conclusions and open problems are given in Section 9.

2 Preliminaries

Graphs. We use standard graph terminology, see for instance [12]. All graphs are finite and undirected. Given a graph G and an edge $e \in E(G)$, let G/e be the graph obtained from G by contracting e, removing loops and parallel edges. 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. Given a vertex u with degree two, by *dissolving* u we denote the operation of replacing u and its two incident edges by an edge between its neighbors.

Topological surfaces. In this paper, surfaces are compact and their boundary is homeomorphic to a finite set (possibly empty) of disjoint circles. We denote by $\beta(\Sigma)$ the number of connected components of the boundary of a surface Σ. The Surface Classification Theorem [29] asserts that 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 $h > 0$ *cross-caps* to the sphere, hence obtaining a non-orientable surface $\mathbb{P}_h$ with Euler characteristic $\chi(\mathbb{P}_h) = 2 - h$. A subset Π of a surface Σ is *surface-separating* if $\Sigma \setminus \Pi$ has at least two connected components.

As a conclusion, our surfaces are determined, up to homeomorphism, by their orientability, their Euler characteristic, and the number of connected components of their boundary. For computational simplicity, it is convenient to work with the *Euler genus* $\gamma(\Sigma)$ of a surface Σ, which is defined as $\gamma(\Sigma) = 2 - \chi(\Sigma)$.

Graphs embedded in surfaces. Our main reference for graphs on surfaces is the monograph of Mohar and Thomassen [29]. For a graph G we use the notation (G, τ) to denote that τ is an embedding of G in Σ (that is, a drawing without edge crossings), whenever the surface Σ is clear from the context. An embedding has *vertices*, *edges*, and *faces*, which are zero-, one-, and two-dimensional open sets, and are denoted $V(G)$, $E(G)$, and $F(G)$, respectively. The degree $\mathrm{d}(v)$ of a vertex v is the number of edges incident with v, counted with multiplicity (loops are counted twice).

For a graph G, the *Euler genus* of G, denoted $\gamma(G)$, is the smallest Euler genus among all surfaces in which G can be embedded. Determining the Euler genus of a graph is an NP-hard problem [38], hence we assume throughout the paper that we are given an already embedded graph. An *O-arc* is a subset of Σ homeomorphic to $\mathbb{S}^1$. A subset of Σ meeting the drawing only at vertices of G 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. Many results in topological graph theory rely on the concept of *representativity* [32, 36], also called *face-width*, which is a parameter that quantifies local planarity and density of embeddings. The representativity $\mathbf{rep}(G, \tau)$ of a graph embedding (G, τ) is the smallest length of a non-contractible (i.e., non null-homotopic) noose in Σ. We call an embedding (G, τ) *polyhedral* [29] if G is 3-connected and $\mathbf{rep}(G, \tau) \geq 3$, or if G is a clique and $1 \leq |V(G)| \leq 3$. With abuse of notation, we also say in that case that the graph G itself is polyhedral.

For a given embedding (G, τ), we denote by (G^*, τ) its dual embedding. Thus G^* is the geometric dual of G. Each vertex v (resp. face r) in (G, τ) corresponds to some face v^* (resp. vertex r^*) in (G^*, τ). Also, given a set $S \subseteq E(G)$, we denote by S^* the set of the duals of the edges in S. Let (G, τ) be an embedding and let (G^*, τ) be its dual. We define the *radial graph embedding* (R_G, τ) of (G, τ) (also known as *vertex-face graph embedding*) as follows: R_G is an embedded bipartite graph with vertex set $V(R_G) = V(G) \cup V(G^*)$. For each pair $e = \{v, u\}$, $e^* = \{u^*, v^*\}$ of dual edges in G and G^*, R_G contains edges $\{v, v^*\}$, $\{v^*, u\}$, $\{u, u^*\}$, and $\{u^*, v\}$. Mohar and Thomassen [29] proved that, if $|V(G)| \geq 4$, the following conditions are equivalent: *(i)* (G, τ) is a polyhedral embedding; *(ii)* (G^*, τ) is a polyhedral embedding; and *(iii)* (R_G, τ) has no multiple edges and every 4-cycle of R_G is the border of some face. The *medial graph embedding* (M_G, τ) of (G, τ) is the dual embedding of the radial embedding (R_G, τ) of (G, τ). Note that (M_G, τ) is a Σ-embedded 4-regular graph.

Tree-like decompositions of graphs. Let G be a graph on n vertices. A *branch decomposition* (T, μ) of a graph G consists of an unrooted 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., $\mathbf{mid}(e) := V(G_1) \cap V(G_2)$. The *width* of (T, μ) is the maximum order of the middle sets over all edges of T, i.e., $\mathbf{w}(T, \mu) := \max\{|\mathbf{mid}(e)| \mid e \in T\}$. An optimal branch decomposition of G is defined by a tree T and a bijection μ which give the minimum width, the *branchwidth*, denoted by $\mathbf{bw}(G)$.

Let $G = (V, E)$ be a connected graph. For $S \subseteq V$, we denote by $\delta(S)$ the set of all edges with an end in S and an end in $V \setminus S$. Let $\{V_1, V_2\}$ be a partition of V. If $G[V \setminus V_1]$ and $G[V \setminus V_2]$ are both non-null and connected, we call $\delta(V_1)$ a *bond* of G [36].

A *carving decomposition* (T, μ) is similar to a branch decomposition, only with the difference that μ is a bijection between the leaves of the tree and the vertex set of the graph G. For an edge e of T, the counterpart of the middle set, called the *cut set* $\mathbf{cut}(e)$, contains the edges of G with endvertices in the leaves of both subtrees. The counterpart of branchwidth is *carvingwidth*, and is denoted by $\mathbf{cw}(G)$. In a *bond carving decomposition*, every cut set is a bond of the graph. That is, in a bond carving decomposition, every cut set separates the graph into two connected components.

Let G_1 and G_2 be graphs with disjoint vertex-sets and let $k \geq 0$ be an integer. For $i = 1, 2$, let $W_i \subseteq V(G_i)$ form a clique of size k and let G_i' $(i = 1, 2)$ be obtained from G_i by deleting some (possibly no) edges from $G_i[W_i]$ with both endvertices in W_i. Consider a bijection $h : W_1 \to W_2$. We define a *clique sum* G of G_1 and G_2, denoted by $G = G_1 \oplus_k G_2$, to be the graph obtained from the union of G_1' and G_2' by identifying w with $h(w)$ for all $w \in W_1$. The integer k is called the *size* of the clique sum. Given a set of graphs $\mathcal{G}$ and an integer $\ell \geq 0$, we define the *ℓ-clique sum closure* of $\mathcal{G}$ as the set of graphs $\mathcal{G}_\ell$ recursively defined as follows: every graph in $\mathcal{G}$ is also in $\mathcal{G}_\ell$, and if $G_1 \in \mathcal{G}$, $G_2 \in \mathcal{G}_\ell$, and $G_3 = G_1 \oplus_k G_2$ with $0 \leq k \leq \ell$, then $G_3 \in \mathcal{G}_\ell$.

3 Connected packing-encodable problems

The standard dynamic programming approach on branch decompositions requires the so called *rooted* branch decomposition, defined as a triple (T, μ, e_r), where (T, μ) is a branch-decomposition of G such that T is a tree rooted on a leaf v_l of T incident with some edge e_r. We slightly abuse notation by insisting that no edge of G is assigned to v_l and thus $\mathbf{mid}(e_r) = \emptyset$ (for this, we arbitrarily pick some edge of a branch decomposition, subdivide it and then connect by e_r the subdivision vertex with a new leaf v_l). The edges of T are oriented towards the root e_r and for each edge $e \in E(T)$ we denote by E_e the edges of G that are mapped to leaves of T that are descendants of e. We also set $G_e = G[E_e]$ and we denote by $L(T)$ the edges of T that are incident with leaves of T. Given an edge e whose tail is a non-leaf vertex v, we denote by $e_1, e_2 \in E(T)$ the two edges heading at v (we call them *children* of e). When the tail of an edge of T is also a leaf of T then we call it *leaf-edge*.

Typically, dynamic programming on a rooted branch decomposition (T, μ, e_r) of a graph G associates some suitable combinatorial structure $\mathsf{struct}(e)$ with each edge e of T such that the knowledge of $\mathsf{struct}(e_r)$ makes it possible to determine the solution to the problem. Roughly speaking, $\mathsf{struct}(e)$ encodes all the ways that the possible certificates of a partial solution on graph G_e may be restricted to $\mathbf{mid}(e)$. The computation of $\mathsf{struct}(e)$ is done bottom-up by first providing $\mathsf{struct}(e)$ when e is a leaf-edge of T and then giving

a recursive way to construct $\mathsf{struct}(e)$ from $\mathsf{struct}(e_1)$ and $\mathsf{struct}(e_2)$, where e_1 and e_2 are the children of e.

The encoding of struct is commonly referred as the "tables" of the dynamic programming algorithm. It is desirable that the size of the tables, as well as the time to process them, is bounded by $f(|\mathbf{mid}(e)|) \cdot n^{O(1)}$, where f is a function not depending on n. This would give a polynomial-time algorithm for graphs of fixed branchwidth. In technical terms, this means that the problem is *Fixed Parameter Tractable* (FPT), when parameterized by the branchwidth of the input graph (for more on Fixed Parameter Tractability, see [17, 20, 31]). A challenge in the design of such algorithms is to reduce the contribution of branchwidth to the size of their tables and therefore to simplify f as much as possible. As indicated by the lower bounds in [7, 25, 26], for many problems like Independent Set, Dominating Set, or q-Coloring for fixed $q \geq 3$, f is not expected to be better than single-exponential in general graphs.

Before we proceed with the description of the family of problems that we examine in this paper, we need some definitions. Let G be a graph and let S be a set of vertices of G. We denote by $\mathcal{G}$ the collection of all subgraphs of G. Each $H \in \mathcal{G}$ defines a packing $\mathcal{P}_S(H)$ of S such that two vertices $x, y \in S$ belong to the same set of $\mathcal{P}_S(H)$ if x, y belong to the same connected component of H. We say that $H_1, H_2 \in \mathcal{G}$ are *S-equivalent* if $\mathcal{P}_S(H_1) = \mathcal{P}_S(H_2)$, and we denote it by $H_1 \equiv_S H_2$. Let $\overline{\mathcal{G}}_S$ the collection of all subgraphs of G modulo the equivalence relation $\equiv_S$. We define the set of all *connected packings of S with respect to G* as the collection

$$\Psi_G(S) = \{\mathcal{P}_S(H) \mid H \in \overline{\mathcal{G}}_S\}.$$

Notice that each member of $\Psi_G(S)$ can indeed be seen as a packing of S, as its sets may not necessarily meet all vertices of S.

In this paper we consider graph problems that can be solved by dynamic programming algorithms on branch decompositions for which the size of $\mathsf{struct}(e)$ is upper-bounded by $2^{O(|\mathbf{mid}(e)|)} \cdot |\Psi_{G_e}(\mathbf{mid}(e))| \cdot n^{O(1)}$. We call these problems *connected packing-encodable*. We stress that our definition of connected packing-encodable problem assumes the existence of *an* algorithm with this property, but there may exist other algorithms whose tables are much bigger. In the introduction, we gave a long list of problems that belong to this category and, in the Appendix, we make a full description on how to do dynamic programming for one of them. For these problems, dynamic programming has a single-exponential dependance on branchwidth if and only if $\Psi_{G_e}(\mathbf{mid}(e))$ contains a single-exponential number of packings, i.e., $|\Psi_{G_e}(\mathbf{mid}(e))| = 2^{O(|\mathbf{mid}(e)|)}$.

However, in general the number of different connected packings that could be created during the dynamic programming is not necessarily smaller than the number of the non-connected ones. Therefore, it may linearly depend on the k-th Bell number, where k is the branchwidth of the input graph. This implies that, in general, $|\Psi_{G_e}(\mathbf{mid}(e))| = 2^{O(k \log k)}$ is the best upper bound we may achieve for connected packing-encodable problems, at least for deterministic algorithms. The purpose of this paper is to show that, for such problems, this bound can be reduced to a single-exponential one when their input graphs have bounded genus. In Section 7, we define the concept of a surface cut decomposition, which is a key tool for the main result of this paper, resumed as follows.

Theorem 3.1 *Every connected packing-encodable problem whose input graph G is embedded in a surface of Euler genus γ, and has branchwidth at most k, can be solved by a dynamic programming algorithm on a surface cut decomposition of G with tables of size $\gamma^{O(k)} \cdot k^{O(\gamma)} \cdot \gamma^{O(\gamma)} \cdot n^{O(1)}$.*

In Section 7, we prove (Theorem 7.2) that, given a graph G embedded in a surface of Euler genus γ, a surface cut decomposition of G of width $O(\mathbf{bw}(G)+\gamma)$ can be constructed in $2^{O(\mathbf{bw}(G))} \cdot n^3$ steps. Therefore, we conclude the following result.

Theorem 3.2 *Every connected packing-encodable problem whose input graph G is embedded in a surface of Euler genus γ, and has branchwidth at most k, can be solved in $\gamma^{O(k)} \cdot k^{O(\gamma)} \cdot \gamma^{O(\gamma)} \cdot n^{O(1)}$ steps.*

Given a parameterized problem with parameter k, an algorithm that solves it in time $2^{O(k)} \cdot n^{O(1)}$ is called *single-exponential* FPT*-algorithm.* As finding an optimal embedding of a graph of genus γ can be solved in $f(\gamma) \cdot n$ steps [28], we can restate Theorem 3.2 as follows.

Corollary 3.3 *Every connected packing-encodable problem on graphs of fixed genus has a single-exponential* FPT*-algorithm, when parameterized by the branchwidth of its input.*

4 Polyhedral decompositions

We introduce in this section *polyhedral decompositions* of graphs embedded in surfaces. Let G be an embedded graph, and let N be a noose in the surface. Similarly to [6], we use the notation $G \rightarrowtail\!\!\!\!\circ N$ for the graph obtained by cutting G along the noose N and gluing a disk on the obtained boundaries.

Definition 4.1 *Given a graph $G = (V, E)$ embedded in a surface of Euler genus γ, a* polyhedral decomposition *of G is a set of graphs $\mathcal{G} = \{H_1, \ldots, H_\ell\}$ together with a set of vertices $A \subseteq V$ such that*

- $|A| = O(\gamma)$*;*
- *H_i is a minor of $G[V \setminus A]$, for $i = 1, \ldots, \ell$;*
- *H_i has a polyhedral embedding in a surface of Euler genus at most γ, for $i = 1, \ldots, \ell$;*
- *$G[V \setminus A]$ belongs to the 2-clique sum closure of $\mathcal{G}$.*

Observation 4.2 *Note that an embedded graph H is not polyhedral if and only if there exists a noose N of length at most two in the surface in which H is embedded, such that either N is non-contractible or $V(H) \cap N$ separates H. Indeed, if H has representativity at most two, then there exists a non-contractible noose N of length at most two. Otherwise, since H is not polyhedral, H has a minimal separator S of size at most two. It is then easy to see that there exists a noose containing only vertices of S.*

Algorithm 1 provides an efficient way to construct a polyhedral decomposition, as it is stated in Proposition 4.3. In the algorithm, the addition of an edge $\{u, v\}$ represents the existence of a path in G between u and v that is not contained in the current component.

Algorithm 1 Construction of a polyhedral decomposition of an embedded graph G

Input: A graph G embedded in a surface of Euler genus γ.
Output: A polyhedral decomposition of G.

$A = \emptyset$, $\mathcal{G} = \{G\}$ (the elements in $\mathcal{G}$, which are embedded graphs, are called *components*).
while $\mathcal{G}$ contains a non-polyhedral component H **do**
  Let N be a noose as described in Observation 4.2 in the surface in which H is embedded, and let $S = V(H) \cap N$.
  if N is non-surface-separating **then**
    Add S to A, and replace in $\mathcal{G}$ component H with $H[V(H) \setminus S] \nearrow N$.
  if N is surface-separating **then**
    Let H_1, H_2 be the subgraphs of $H \nearrow N$ corresponding to the two surfaces occurring after splitting H
    if $S = \{u\} \cup \{v\}$ and $\{u, v\} \notin E(H)$ **then**
      Add the edge $\{u, v\}$ to H_i, $i = 1, 2$.
    Replace in $\mathcal{G}$ component H with the components of $H \nearrow N$ containing at least one edge of H.
return $(\mathcal{G}, A)$.

Proposition 4.3 *Given a graph G on n vertices embedded in a surface, Algorithm 1 constructs a polyhedral decomposition of G in $O(n^3)$ steps.*

Proof: We first prove that the the output $(\mathcal{G}, A)$ of Algorithm 1 is indeed a polyhedral decomposition of G, and then we analyze the running time.

Let us see that each component of $\mathcal{G}$ is a minor of $G[V \setminus A]$. Indeed, the only edges added to G by Algorithm 1 are those between two non-adjacent vertices u, v that separate a component H into several components $H_1, \ldots, H_\ell$. For each component H_i, $i = 1, \ldots, \ell$, there exists a path between u and v in $H \setminus H_i$ (provided that the separators of size 1 have been already removed, which can we assumed without loss of generality), and therefore the graph obtained from H_i by adding the edge $\{u, v\}$ is a minor of H, which is inductively a minor of $G[V \setminus A]$. Also, each component of $\mathcal{G}$ is polyhedral by definition of the algorithm.

As a non-separating noose is necessarily non-contractible, each time some vertices are moved to A, the Euler genus of the surfaces strictly decreases [29, Lemma 4.2.4]. Therefore, $|A| = O(\gamma)$.

By the construction of the algorithm, it is also clear that each component of $\mathcal{G}$ has a polyhedral embedding in a surface of Euler genus at most γ. Finally, $G[V \setminus A]$ can be constructed by joining the graphs of $\mathcal{G}$ applying clique sums of size at most two.

Thus, $(\mathcal{G}, A)$ is a polyhedral decomposition of G according to Definition 4.1.

We now analyze the running time of the algorithm. Separators of size at most two can be found in $O(n^2)$ steps [24]. A noose with respect to a graph H corresponds to a cycle in the radial graph of H, hence can also be found[2] in $O(n^2)$ (using that the number of edges of a bounded-genus graph is linearly bounded by its number of vertices). Since each time that we find a *small* separator we decrease the size of the components, the running time of the algorithm is $O(n^3)$. □

[2] A shortest non-contractible cycle can be found in $2^{O(\gamma \log \gamma)} n^{4/3}$ steps [6]. This running time improves on $O(n^3)$ for a big range of values of γ.

5 Width parameters of graphs on surfaces

In this section we state some definitions and auxiliary results about several width parameters of graphs on surfaces, to be applied in Section 7 for building surface cut decompositions. In the same spirit of [23, Theorem 1] we can prove the following lemma. We omit the proof here since the details are very similar[3] to the proof in [23].

Lemma 5.1 *Let (G,τ) and (G^*,τ) be dual polyhedral embeddings in a surface of Euler genus γ and let (M_G,τ) be the medial graph embedding. Then* $\max\{\mathbf{bw}(G),\mathbf{bw}(G^*)\} \leq \mathbf{cw}(M_G)/2 \leq 6\cdot\mathbf{bw}(G)+2\gamma+O(1)$. *In addition, given a branch decomposition of G of width at most k, a carving decomposition of M_G of width at most $12k$ can be found in linear time.*

We would like to point out that in Lemma 5.1 we need the embeddings to be polyhedral.

Lemma 5.2 (folklore) *The removal of a vertex from a non-acyclic graph decreases its branchwidth by at most* 1.

Lemma 5.3 *Let G be a graph and let $\mathcal{G}$ be a collection of graphs such that G can be constructed by joining graphs in $\mathcal{G}$ applying clique sums of size at most two. Given branch decompositions $\{(T_H,\mu_H) \mid H\in\mathcal{G})\}$, we can compute in linear time a branch decomposition (T,μ) of G such that $\mathbf{w}(T,\mu)\leq\max\{2,\{\mathbf{w}(T_H,\mu_H)\mid H\in\mathcal{G}\}\}$. In particular, $\mathbf{bw}(G)\leq\max\{2,\{\mathbf{bw}(H)\mid H\in\mathcal{G}\}\}$.*

Proof: Note that if G_1 and G_2 are graphs with no vertex (resp. a vertex, an edge) in common, then $G_1\cup G_2 = G_1\oplus_0 G_2$ (resp. $G_1\oplus_1 G_2$, $G_1\oplus_2 G_2$). To prove Lemma 5.3, we need the following two lemmata.

Lemma 5.4 *Let G_1 and G_2 be graphs with at most one vertex in common. Then* $\mathbf{bw}(G_1\cup G_2)=\max\{\mathbf{bw}(G_1),\mathbf{bw}(G_2)\}$.

Proof: Assume first that G_1 and G_2 share one vertex v. Clearly $\mathbf{bw}(G_1\cup G_2)\geq\max\{\mathbf{bw}(G_1),\mathbf{bw}(G_2)\}$. Conversely, for $i=1,2$, let (T_i,μ_i) be a branch decomposition of G_i such that $\mathbf{w}(T_i,\mu_i)\leq k$. For $i=1,2$, let T_i^v be the minimal subtree of T_i containing all the leaves u_i of T_i such that v is an endvertex of $\mu_i(u_i)$. For $i=1,2$, we take an arbitrary edge $\{a_i,b_i\}$ of T_i^v, we subdivide it by adding a new vertex w_i, and then we build a tree T from T_1 and T_2 by adding the edge $\{w_1,w_2\}$. We claim that $(T,\mu_1\cup\mu_2)$ is a branch decomposition of $G_1\cup G_2$ of width at most k. Indeed, let us compare the middle sets of $(T,\mu_1\cup\mu_2)$ to those of (T_1,μ_1) and (T_2,μ_2). First, it is clear that the vertices of $V(G_1)\cup V(G_2)-\{v\}$ appear in $(T,\mu_1\cup\mu_2)$ in the same middle sets as in (T_1,μ_1) and (T_2,μ_2). Secondly, $\mathbf{mid}(\{w_1,w_2\})=\{v\}$, since v is a cut-vertex of $G_1\cup G_2$. Also, for $i=1,2$, $\mathbf{mid}(\{a_i,w_i\})=\mathbf{mid}(\{w_i,b_i\})=\mathbf{mid}(\{a_i,b_i\})$, and the latter has size at most k as $\mathbf{w}(T_i,\mu_i)\leq k$. For all other edges e of T_i, $i=1,2$, $\mathbf{mid}(e)$ is exactly the same in T and in T_i, since if $e\in E(T_i^v)$ then $v\in\mathbf{mid}(e)$ in both T and T_i, and if $e\in E(T_i\setminus T_i^v)$ then $v\notin\mathbf{mid}(e)$ in both T and T_i.

[3]The improvement in the multiplicative factor of the Euler genus is obtained by applying more carefully Euler's formula in the proof analogous to that of [23, Lemma 2].

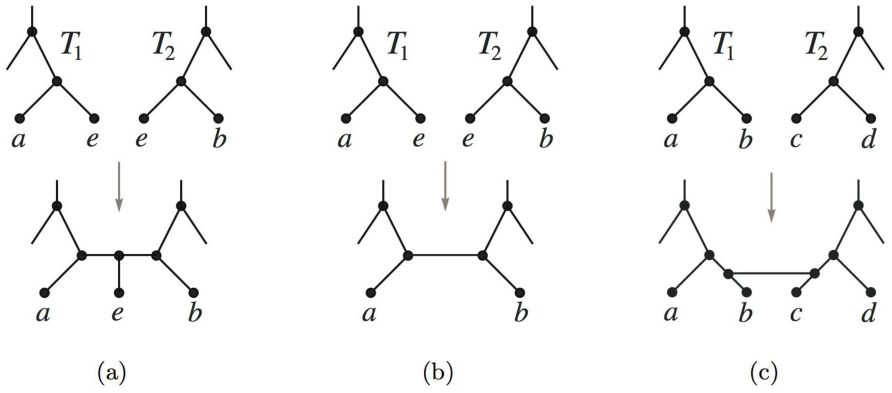

Figure 1: Merging branch decompositions (T_1, μ_1) and (T_2, μ_2) of two components H_1 and H_2 in a polyhedral decomposition $(\mathcal{G}, A)$ of $G = (V, E)$. There are three cases: (a) H_1 and H_2 share two vertices v_1, v_2 and the edge $e = \{v_1, v_2\}$ is in E; (b) H_1 and H_2 share two vertices v_1, v_2 and $e = \{v_1, v_2\}$ is *not* in E; (c) H_1 and H_2 share one vertex v.

If G_1 and G_2 share no vertices, we can merge two branch decompositions (T_1, μ_1) and (T_2, μ_2) by subdividing a pair of arbitrary edges, without increasing the width. □

Lemma 5.5 (Fomin and Thilikos [22]) *Let G_1 and G_2 be graphs with one edge f in common. Then* $\mathbf{bw}(G_1 \cup G_2) \leq \max\{\mathbf{bw}(G_1), \mathbf{bw}(G_2), 2\}$. *Moreover, if both endvertices of f have degree at least two in at least one of the graphs, then* $\mathbf{bw}(G_1 \cup G_2) = \max\{\mathbf{bw}(G_1), \mathbf{bw}(G_2)\}$.

It remains only to show how to merge the branch decompositions (T_1, μ_1), (T_2, μ_2) of two graphs H_1, H_2 in $\mathcal{G}$. We distinguish four cases:

(a) H_1 and H_2 share two vertices v_1, v_2, and the edge $e = \{v_1, v_2\} \in E(G)$. We take the leaves in T_1 and T_2 corresponding to e, we identify them, and we add a new edge whose leave corresponds to e (see Figure 1(a)).

(b) H_1 and H_2 share two vertices v_1, v_2, and the edge $e = \{v_1, v_2\} \notin E(G)$. We take the leaves in T_1 and T_2 corresponding to e, we identify them, and we dissolve the common vertex (see Figure 1(b)).

(c) H_1 and H_2 share one vertex v. We take two edges b, c in T_1, T_2 whose leaves correspond to edges containing v, we subdivide them and add a new edge between the newly created vertices (see Figure 1(c)).

(d) H_1 and H_2 share no vertices. We do the construction of case (c) for any two edges of the two branch decompositions.

The above construction does not increase the branchwidth by Lemmata 5.4 and 5.5.

□

Observation 5.6 *Let $G = (V, E)$ be a graph, and let $A \subseteq V$. Given a branch decomposition (T', μ') of $G[V \setminus A]$, we can obtain a branch decomposition (T, μ) of G with*

$\mathbf{w}(T,\mu) \leq \mathbf{w}(T',\mu') + |A|$ *recursively as follows: First, for each edge* $\{u,v\} \in E(G)$ *with* $u \in V \setminus A$ *and* $v \in A$*, we choose an edge* $e \in G[V \setminus A]$ *containing* u*, and we replace the leaf of* T' *corresponding to* e *with two incident pendant edges whose leaves correspond to edges* $\{u,v\}$ *and* e*, respectively. Finally, for each edge* $\{u,v\} \in E(G)$ *with* $u,v \in A$*, we take and arbitrary edge of* T'*, subdivide it, and add a new edge whose leave corresponds to edge* $\{u,v\}$*. It can be easily checked that the size of the middle sets has increased by at most* $|A|$*.*

Given an embedded graph G and a carving decomposition (T,μ) of its medial graph M_G, we define a *radial decomposition* (T^*,μ^*) of the dual graph R_G, where $T^* = T$ and μ^* is a bijection from the leaves of T to the set of faces of R_G defined as follows: for each edge $e \in E(T)$, $\mu^*(e) = f$, where f is the face in R_G corresponding to the vertex $u_f \in V(M_G)$ such that $\mu(e) = u_f$. Each edge $e \in E(T^*)$ partitions the faces of R_G into two sets F_1 and F_2. We define the *border set* of e, denoted $\mathbf{bor}(e)$, as the set of edges of R_G that belong to both F_1 and F_2. Note that F_1 and F_2 may intersect also in vertices, not only in edges.

If (T,μ) is a bond carving decomposition of M_G, then the associated radial decomposition (also called *bond*) has nice connectivity properties. Indeed, in a bond carving decomposition, every cut set partitions the vertices of M_G into two subsets V_1, V_2 such that both $M_G[V_1]$ and $M_G[V_2]$ are non-null and connected. This property, seen in the radial decomposition of R_G, implies that each edge $e \in E(T^*)$ corresponds to a partition of the faces of R_G into two sets F_1 and F_2, namely *black* and *white* faces (naturally partitioning the edges into *black*, *white*, and *grey*), such that it is possible to reach any black (resp. white) face from any black (resp. white) face by only crossing black (resp. white) edges. In other words, the union of all black (resp. white) faces and edges is a connected set.

Observation 5.7 *Recall that all the faces of a radial graph* R_G *are* tiles*, that is, each face has exactly 4 edges. Also, each one of those tiles corresponds to a pair of dual edges* e *and* e^* *of* G *and* G^**, respectively. Given a carving decomposition* (T,μ) *of* M_G *(or equivalently, a radial decomposition* (T^*,μ^*) *of* R_G*), one can obtain in a natural way branch decompositions of* G *and* G^* *by redefining the bijection* μ *from the leaves of* T *to the edges of* G *(or* G^**) that correspond to the faces of* R_G*.*

6 Some topological results

In this section we state two topological lemmata and some definitions that will be used in Section 7. Given a collection $\mathcal{S}$ of sets, we denote their union by $\bigcup\mathcal{S} = \bigcup_{S\in\mathcal{S}} S$.

Given a graph G embedded in a surface of Euler genus γ, its dual G^* and a spanning tree C^* of G^*, we call $C = \{e \in E(G) \mid e^* \in E(C^*)\}$ a *spanning cotree* of G. We define a *tree-cotree partition* (cf. [18]) of an embedded graph G to be a triple (T,C,X) where T is a spanning tree of G, C is a spanning cotree of G, $X \subseteq E(G)$, and the three sets $E(T)$, C, and X form a partition of $E(G)$. Eppstein proved [18, Lemma 3.1] that if T and C^* are forests such that $E(T)$ and C are disjoint, we can make T become part of a spanning tree T' and C become part of a spanning cotree disjoint from T', extending T and C to a tree-cotree decomposition. We can now announce the following lemma from [18, Lemma 3.2].

Lemma 6.1 (Eppstein [18]) *If (T, C, X) is a tree-cotree decomposition of a graph G embedded in a surface of Euler genus γ, then $|X| = O(\gamma)$.*

Let Σ be a surface and let $\mathcal{N}$ be a finite collection of O-arcs in Σ pairwise intersecting at a finite zero-dimensional subsets (i.e., points) of Σ. For a point $p \in \Sigma$, let $\mathcal{N}(p)$ be the number of O-arcs in $\mathcal{N}$ containing p, and let $P(\mathcal{N}) = \{p \in \Sigma : \mathcal{N}(p) \geq 2\}$; note that by assumption $P(\mathcal{N})$ is a finite set of points of Σ. Then we define

$$\theta(\mathcal{N}) = \sum_{p \in P(\mathcal{N})} (\mathcal{N}(p) - 1) \ .$$

Lemma 6.2 *Let Σ be a surface without boundary with $\gamma(\Sigma) = \gamma$. Let $\mathcal{N}$ be a collection of $O(\gamma)$ O-arcs in Σ pairwise intersecting at finite zero-dimensional subsets of Σ, and such that $\Sigma \setminus \bigcup \mathcal{N}$ has two connected components. Then $\theta(\mathcal{N}) = O(\gamma)$.*

Proof: In order to prove the lemma, we define from $\mathcal{N}$ the following (multi)graph H embedded in Σ: we first add a vertex v_p in H for every point p in Σ such that $\mathcal{N}(p) \geq 2$. We call such points *repeated*. We now distinguish four cases according to the number of repeated points in an O-arc. First, for each O-arc N with at least three repeated points, we order cyclically the repeated points in N, and the same ordering applies to the corresponding vertices in H. Then, we add an edge in H between each two consecutive vertices in that ordering. For each O-arc with exactly two repeated points p and q, we add two parallel edges in H between v_p and v_q. For each O-arc with exactly one repeated point p, we add in H a loop at vertex v_p. Finally, for each O-arc N with no repeated points, we add to H a new vertex v_N with a loop. Visually, H is the graph embedded in Σ corresponding to the union of the O-arcs in $\mathcal{N}$.

In order to prove the result, by the construction of H it is enough to prove that $\sum_{v \in V(H)} (\mathrm{d}_H(v) - 2) = O(\gamma)$. By assumption, H separates Σ into two connected components Σ' and Σ''. Let $H_1, H_2, \ldots, H_r$ be the maximal connected subgraphs of H. In particular, $r \leq |\mathcal{N}| = O(\gamma)$ by hypothesis. Some of these connected subgraphs may be incident with Σ' but not with Σ'', or vice-versa. Additionally, there is at least one connected subgraph H_i incident with both connected components. Without loss of generality we assume that the subgraphs $H_1, H_2, \ldots, H_p$ are incident only with Σ', $H_{p+1}, \ldots, H_q$ are incident with both components, and $H_{q+1}, \ldots, H_r$ are incident only with Σ''. It is clear that there exists a path joining a vertex in H_i with a vertex in H_{i+1} if $1 \leq i \leq q-1$ or $p+1 \leq i \leq r-1$.

From graphs $H_1, H_2, \ldots, H_p, \ldots, H_q$ (the ones which are incident with Σ') we construct a new graph G_1 in the following inductive way: we start taking H_q and H_{q-1}, and a path joining a vertex in H_q to a vertex in H_{q-1}. This path exists because H_q and H_{q-1} are incident with Σ'. Consider the graph obtained from H_q and H_{q-1} by adding an edge that joins this pair of vertices. Then, we delete H_q and H_{q-1} from the initial list and add this new connected graph. This procedure is done $q-1$ times. At the end, we obtain a connected graph G' incident with both Σ' and Σ'' where each vertex has degree at least three. Finally, we apply the same procedure with $G', H_{q+1}, \ldots, H_r$, obtaining a connected graph G. Observe also that

$$\sum_{v \in V(H)} (\mathrm{d}_H(v) - 2) \ \leq \ \sum_{v \in V(G)} (\mathrm{d}_G(v) - 2) \ < \ \sum_{v \in V(G)} \mathrm{d}_G(v) \ = \ 2|E(G)| \ .$$

In what follows, we obtain upper bounds for $2|E(G)|$. Observe that H defines a pair of faces over Σ, not necessarily disks. In the previous construction of G, every time we add an edge we either subdivide a face into two parts or not. Consequently, the number of faces that G defines over Σ is at most $2 + |\mathcal{N}|$. The next step consists in reducing the surface in the following way: let f be a face determined by G over Σ. If f is contractible, we do nothing. If it is not, there is a non-contractible cycle $\mathbb{S}^1$ contained on f. Let Σ_1 be the connected component of $\Sigma \nsim \mathbb{S}^1$ which contains G. Then G defines a decomposition of Σ_1, $\gamma(\Sigma_1) \leq \gamma$, and the number of faces has been increased by at most one. Observe that for each operation $\nsim$ we reduce the Euler genus and we create at most one face. As the Euler genus is finite, so is the number of $\nsim$ operations. This gives rise to a surface Σ_s with $\gamma(\Sigma_s) \leq \gamma$, and such that all faces determined by G are contractible. Additionally, the number of faces that G determines over Σ_s is smaller than $2 + |\mathcal{N}| + \gamma$.

G defines a map on Σ_s (i.e., all faces are contractible), and consequently we can apply Euler's formula. Then $|F(G)|+|V(G)| = |E(G)|+2-\gamma(\Sigma_s)$. Then, as $|F(G)| \leq 2+|\mathcal{N}|+\gamma$, we obtain that $|E(G)| + 2 - \gamma(\Sigma_s) = |V(G)| + |F(G)| \leq |V(G)| + 2 + |\mathcal{N}| + \gamma$. The degree of each vertex is at least three, and thus $3|V(G)| \leq 2|E(G)|$. Substituting this condition in the previous equation, we obtain

$$|E(G)| + 2 - \gamma(\Sigma_s) \;\leq\; |V(G)| + 2 + |\mathcal{N}| + \gamma \;\leq\; \frac{2}{3}|E(G)| + 2 + |\mathcal{N}| + \gamma.$$

Isolating $|E(G)|$, we get that $2|E(G)| \leq 6|\mathcal{N}|+6\gamma(\Sigma_s)+6\gamma \leq 6|\mathcal{N}|+12\gamma$. As by hypothesis $|\mathcal{N}| = O(\gamma)$, the previous bound yields the desired result. □

7 Surface cut decompositions

Sphere cut decompositions [36] have proved to be very useful to analyze the running time of algorithms based on dynamic programming over branch decompositions on planar graphs [14–16, 35]. In this section we generalize sphere cut decompositions to graphs on surfaces; we call them *surface cut decompositions*.

Definition 7.1 *Given a graph G embedded in a surface Σ with $\gamma(\Sigma) = \gamma$, a* surface cut decomposition *of G is a branch decomposition (T, μ) of G such that there exists a polyhedral decomposition $(\mathcal{G}, A)$ of G with the following property: for each edge $e \in E(T)$, either $|\mathbf{mid}(e) \setminus A| \leq 2$, or there exists a graph $H \in \mathcal{G}$ such that*

- $\mathbf{mid}(e) \setminus A \subseteq V(H)$;
- *the vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $O(\gamma)$ nooses of Σ pairwise intersecting only at subsets of $\mathbf{mid}(e) \setminus A$.*
- $\theta(\mathcal{N}) = O(\gamma)$.
- $\Sigma \setminus \bigcup \mathcal{N}$ *contains exactly two connected components, such that the graph $G_e \setminus A$ is embedded in the closure of one of them.*

Note that a sphere cut decomposition is a particular case of a surface cut decomposition when $\gamma = 0$, by taking $A = \emptyset$, $\mathcal{G}$ containing only the graph G itself, and all the vertices of each middle set contained in a single noose. We provide now an algorithm to construct

a surface graph decomposition of an embedded graph. The proof of Theorem 7.2 uses Proposition 4.3 and all the results of Sections 5 and 6.

Algorithm 2 Construction of a surface cut decomposition of an embedded graph G

Input: An embedded graph G.
Output: A surface cut decomposition of G.

Compute a polyhedral decomposition $(\mathcal{G}, A)$ of G, using Algorithm 1.
for each component H of $\mathcal{G}$ **do**
1. Compute a branch decomposition (T'_H, μ'_H) of H, using [2, Theorem 3.8].
2. Transform (T'_H, μ'_H) to a carving decomposition (T^c_H, μ^c_H) of the medial graph M_H, using Lemma 5.1.
3. Transform (T^c_H, μ^c_H) to a *bond* carving decomposition (T^b_H, μ^b_H) of M_H, using [36].
4. Transform (T^b_H, μ^b_H) to a branch decomposition (T_H, μ_H) of H, using Observation 5.7.

Construct a branch decomposition (T, μ) of G by merging, using Lemma 5.3, the branch decompositions $\{(T_H, \mu_H) \mid H \in \mathcal{G}\}$, and by adding the edges of G with at least one endvertex in A, using Observation 5.6.
return (T, μ).

Theorem 7.2 *Given a graph G on n vertices embedded in a surface of Euler genus γ, with $\mathbf{bw}(G) \leq k$, Algorithm 2 constructs, in $2^{3k+O(\log k)} \cdot n^3$ steps, a surface cut decomposition (T, μ) of G of width at most $27k + O(\gamma)$.*

Proof: We prove, in this order, that the output (T, μ) of Algorithm 2 is indeed a surface cut decomposition of G, then that the width of (T, μ) is at most $27\mathbf{bw}(G) + O(\gamma)$, and finally the claimed running time.

(T, μ) is a surface cut decomposition of G. We shall prove that all the properties of Definition 7.1 are fulfilled. First note that, as $(\mathcal{G}, A)$ is a polyhedral decomposition of G, we have that $|A| = O(\gamma)$.

By construction, it is clear that (T, μ) is a branch decomposition of G. In (T, μ), there are some edges that have been added in the last step of Algorithm 2, in order to merge branch decompositions of the graphs in $\mathcal{G}$, with the help of Lemma 5.3. Let e be such an edge. Since $(\mathcal{G}, A)$ is a polyhedral decomposition of G, any two graphs in $\mathcal{G}$ share at most two vertices, hence $|\mathbf{mid}(e) \setminus A| \leq 2$.

All other edges of (T, μ) correspond to an edge of a branch decomposition of some polyhedral component $H \in \mathcal{G}$. Let henceforth e be such an edge. Therefore, $\mathbf{mid}(e) \setminus A \subseteq V(H)$. To complete this part of the proof, we prove in a sequence of three claims that the remaining conditions of Definition 7.1 hold.

Claim 7.3 *The vertices in $\mathbf{mid}(e) \setminus A$ are contained in a set $\mathcal{N}$ of $O(\gamma)$ nooses.*

Proof: The proof uses the tree-cotree partition defined in Section 6.

Recall that e is an edge that corresponds to a branch decomposition (T_H, μ_H) of a polyhedral component H of $\mathcal{G}$. The branch decomposition (T_H, μ_H) of H has been built by Algorithm 2 from a bond carving decomposition of its medial graph M_H, or equivalently from a bond radial decomposition of its radial graph R_H. Due to the fact that the carving

decomposition of M_H is bond, edge e partitions the vertices of M_H into two sets – namely, *black* and *white* vertices – each one inducing a connected subgraph of M_H. There are three types of edges in R_H: *black*, *white*, and *grey*, according to whether they belong to faces of the same color (black or white) or not. Therefore, the corresponding black and white faces also induce connected subgraphs of R_H, in the sense that it is always possible to reach any black (resp. white) face from any black (resp. white) face only crossing black (resp. white) *edges*.

Let F be the set of grey edges of R_H. Since each edge of R_H contains a vertex from H and another from H^*, the vertices in $\mathbf{mid}(e)$ are contained in $R_H[F]$, so it suffices to prove that $R_H[F]$ can be partitioned into a set of $O(\gamma)$ cycles (possibly sharing some vertices). Note that each cycle in the radial graph R_H corresponds to a noose in the surface.

To this end, first note that in $R_H[F]$ all vertices have even degree. Indeed, let $v \in V(R_H[F])$, and consider a clockwise orientation of the edges incident with v in $R_H[F]$. Each such edge alternates from a black to a white face, or viceversa, so beginning from an arbitrary edge and visiting all others edges in the clockwise order, we deduce that the number of edges incident with v is necessarily even.

Therefore, $R_H[F]$ can be partitioned into a set of cycles. Let us now bound the number of such cycles. Since the subgraph induced by the black (resp. white) faces of R_H is connected, we can consider in M_H a spanning tree T_B^* (resp. T_W^*) corresponding to the black (resp. white) faces of R_H. Merge both trees by adding a new edge e_0^*, and let T^* be the resulting tree. Let T be a spanning tree of R_H disjoint from T^* (in the sense that there is no pair of dual edges e and e^* with $e \in E(T)$ and $e^* \in E(T^*)$); such a spanning tree exists by [18, Lemma 3.1]. Now consider the tree-cotree partition (T, T^*, X), where X is the set of edges of R_H that are neither in T nor in T^*.

Each edge of T^*, except e_0^*, corresponds to two faces of R_H of the same color. Therefore, the set $F \in E(R_H)$ of edges separating faces of different color is contained in $T \cup \{e_0\} \cup X$. Since T is a tree, each cycle of $R_H[F]$ uses at least one edge in $\{e_0\} \cup X$. Therefore, $R_H[F]$ can be partitioned into at most $1 + |X|$ cycles. The result follows from the fact that (T, T^*, X) is a tree-cotree partition, and therefore $|X| = O(\gamma)$ by Lemma 6.1. □

Claim 7.4 *Let $\mathcal{N}$ be the set of nooses constructed in the proof of Claim 7.3. Then $\bigcup\mathcal{N}$ separates Σ into two connected components.*

Proof: By Claim 7.3, the vertices in $\mathbf{mid}(e) \setminus A$ are contained in $\bigcup\mathcal{N}$. The claim holds from the fact that for each component H of $\mathcal{G}$, (T_H^b, μ_H^b) is a bond carving decomposition of M_H, and by taking into account the discussion before Observation 5.7. □

Note that the collection of nooses constructed in the proof of Claim 7.3 is finite and its elements pairwise intersect only at subsets of $\mathbf{mid}(e) \setminus A$, as required. In particular, for this collection $\mathcal{N}$ of nooses, the parameter $\theta(\mathcal{N})$ is well-defined (see Section 6).

Claim 7.5 *Let $\mathcal{N}$ be the set of nooses constructed in the proof of Claim 7.3. Then $\theta(\mathcal{N}) = O(\gamma)$.*

Proof: By Claim 7.4, $\bigcup\mathcal{N}$ separates Σ into two connected components. The claim then holds by Lemma 6.2. □

The width of (T, μ) is at most $27 \cdot \mathbf{bw}(G) + O(\gamma)$. For simplicity, let $k = \mathbf{bw}(G)$. By Proposition 4.3, each polyhedral component H is a minor of G, hence $\mathbf{bw}(H) \leq k$ for all $H \in \mathcal{G}$. In Step 1 of Algorithm 2, we compute a branch decomposition (T'_H, μ'_H) of H of width at most $k' = \frac{9}{2}k$, using Amir's algorithm [2, Theorem 3.8]. In Step 2, we transform (T'_H, μ'_H) to a carving decomposition (T^c_H, μ^c_H) of the medial graph M_H of H of width at most $12k'$, using Lemma 5.1. In Step 3, we transform (T^c_H, μ^c_H) to a *bond* carving decomposition (T^b_H, μ^b_H) of M_H of width at most $12k'$, using the algorithm of [36]. Then, using Observation 5.7, we transform in Step 4 (T^b_H, μ^b_H) to a branch decomposition (T_H, μ_H) of H. By the proof of Claim 7.3, the discrepancy between $\mathbf{w}(T_H, \mu_H)$ and $\mathbf{w}(T^b_H, \mu^b_H)/2$ is at most the bound provided by Lemma 6.2, i.e., $O(\gamma)$. Therefore, $\mathbf{w}(T_H, \mu_H) \leq 6k' + O(\gamma) = 27k + O(\gamma)$, for all $H \in \mathcal{G}$.

Then, we merge the branch decompositions of the polyhedral components, using Lemma 5.3, and finally we add the edges of G with at least one endvertex in A, using Observation 5.6, to obtain a branch decomposition (T, μ) of G.

Combining the discussion above with Lemmata 5.2 and 5.3 and Observation 5.6, and using that $|A| = O(\gamma)$, we get that

$$\begin{aligned} \mathbf{w}(T, \mu) &\leq \max\{2, \{\mathbf{w}(T_H, \mu_H) \mid H \in \mathcal{G}\}\} + |A| \\ &\leq 27k + O(\gamma) + |A| \\ &= 27k + O(\gamma). \end{aligned}$$

Algorithm 2 runs in $2^{3k+O(\log k)} \cdot n^3$ time. We analyze sequentially the running time of each step. First, we compute a polyhedral decomposition of G using Algorithm 1 in $O(n^3)$ steps, by Proposition 4.3. Then, we run Amir's algorithm in each component in Step 1, which takes $O(2^{3k}k^{3/2}n^2)$ time [2, Theorem 3.8]. We would like to stress that this step is the only non-polynomial procedure in the construction of surface cut decompositions. Step 2 can be done in linear time by Lemma 5.1. Step 3 can be done in $O(n^2)$ time [36]. Step 4 takes linear time by Observation 5.7. Merging the branch decompositions can clearly be done in linear time. Finally, since any two elements in $\mathcal{G}$ share at most two vertices, the overall running time is the claimed one. □

8 Upper-bounding the size of the tables

In this section we show that by using surface cut decompositions in order to solve connected packing-encodable problems in surface-embedded graphs, one can guarantee single-exponential upper bounds on the size of the tables of dynamic programming algorithms. Then Theorem 3.2 follows directly by the definition of a connected packing-encodable problem and the following lemma.

Lemma 8.1 *Let G be a graph embedded in a surface Σ without boundary and Euler genus γ, and let (T, μ) be a surface cut decomposition of G of width at most k. Then for every $e \in E(T)$, $|\Psi_{G_e}(\mathbf{mid}(e))| = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot \gamma^{O(k)}$.*

Before we give the proof of the above lemma, we first need to define formally the notion of non-crossing partitions on surfaces with boundary and then to prove some lemmata that

combine elements from topology and combinatorics.

A *non-crossing partition* of a set of size k, from a combinatorial point of view, is a partition of the set $\{1, 2, \ldots, k\}$ with the following property: if $\{a, b, c, d\} \subseteq \{1, 2, \ldots, k\}$ with $1 \leq a < b < c < d \leq k$ and some subset in the partition contains a and c, then no other subset contains both b and d. One can represent such a partition on a disk by placing k points on the boundary of the disk, labeled consecutively, and drawing each subset as a convex polygon (also called *block*) on the points belonging to the subset. Then, the "non-crossing" condition is equivalent to the fact that the blocks are pairwise disjoint. See Figure 2 for some examples.

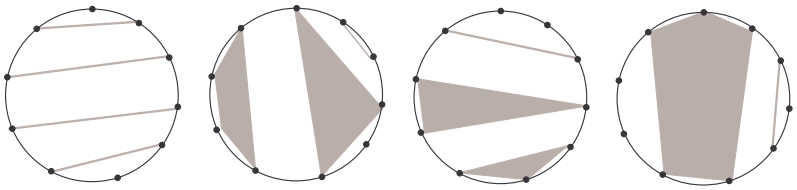

Figure 2: Non-crossing partitions on a disk, which enumerate the number of partial solutions on planar graphs when using sphere cut decompositions.

The enumeration of non-crossing partitions on a disk is one of the first non-trivial problems in enumerative combinatorics: it is well-known (see e.g. [19]) that the the number of non-crossing partitions of $\{1, 2, \ldots, k\}$ on a disk is equal to the Catalan number $C(k) = \frac{1}{k+1}\binom{2k}{k} \sim \frac{4^k}{k^{3/2}\sqrt{\pi}} = O(4^k)$. This is the main combinatorial property exploited to obtain single-exponential dynamic programming algorithms on planar graphs using sphere cut decompositions [16, 35, 36].

The generalization of the notion of non-crossing partition to surfaces of higher genus is not as straignforward as in the case of the disk, and must be defined carefully. We consider pairs (Σ, S) where Σ is a surface whose boundary has $\beta(\Sigma)$ connected components, each one homeomorphic to a simple circle, and S is a set of vertices on this boundary. A *partition family* for the pair (Σ, S) is a collection $\mathfrak{B}$ of mutually non-intersecting connected subsets of Σ, such that each vertex in S belongs to some set in $\mathfrak{B}$.

Actually the concept of a partition family is not enough for our purposes, as we have to incorporate the presence of the set of vertices A arising from a polyhedral decomposition. This set of vertices plays is some sense the role of *apices* in Graph Minors theory, and this is why we also call these vertices *apices*. For this reason we consider pairs of the form $(\Sigma \cup \Gamma_A, S \cup A)$ where Σ is a surface with boundary, S is a set of vertices on this boundary, A is a vertex set not on the boundary (the apices), and Γ_A is the closed set containing the points of the graph C_A obtained if we take a complete graph with A as vertex set and add to it S together with all edges between the vertices of A and S. We require that $\Gamma_A \cap \Sigma = S$ and we see the set Γ_A as "flying above" the surface Σ. That way, we call the edges of C_A *flying edges*, and we treat them as *closed subsets* of Γ_A by adding to them the two endpoints of their boundary. We use the notation Σ^A to denote $\Sigma \cup \Gamma_A$ (clearly $\Sigma = \Sigma^\emptyset$). To extend the definition of partition family, we take a partition family $\mathfrak{B}_\Sigma$ of

Σ and, on top of it, we consider a set $\mathfrak{E}_A$ of flying edges where each apex is incident with some edge in $\mathfrak{E}_A$. An *extended partition family* for (Σ^A, S) is a collection $\mathfrak{B}_{\Sigma^A}$ of subsets of $\Sigma^A \cup S$ defined as

$$\mathfrak{B}_{\Sigma^A} = \{C \mid C \text{ is a connected component of the set } \mathbf{U}(\mathfrak{B}_\Sigma \cup \mathfrak{E}_A)\} \ ,$$

where $\mathfrak{B}_\Sigma$ and $\mathfrak{E}_A$ are taken as before. For simplicity, we may drop the index of a collection $\mathfrak{B}_\Sigma$ or $\mathfrak{B}_{\Sigma^A}$ when it is clear from the context whether it refers to Σ or to Σ^A.

Notice that each partition family $\mathfrak{B}$ for $(\Sigma^A, S \cup A)$ defines a partition of $S \cup A$ as follows.

$$\mathcal{R}(\mathfrak{B}) = \{(S \cup A) \cap B \mid B \in \mathfrak{B}\}.$$

We say that two extended partition families $\mathfrak{B}_1$ and $\mathfrak{B}_2$ for $(\Sigma^A, S \cup A)$ are *equivalent* if $\mathcal{R}(\mathfrak{B}_1) = \mathcal{R}(\mathfrak{B}_2)$ and we denote it by $\mathfrak{B}_1 \equiv \mathfrak{B}_2$. The set of the *non-crossing partitions with apices* of the set $S \cup A$ (where S and A are vertices embedded in Σ^A as before), denoted by $\Pi_{\Sigma^A}(S \cup A)$, is the set of equivalence classes of the extended partition families for $(\Sigma^A, S \cup A)$ with respect to the relation $\equiv$.

We define $\Pi_\Sigma(S) = \Pi_{\Sigma^\emptyset}(S \cup \emptyset)$, and note that, if Σ is the disk and $|S| = k$, then $|\Pi_\Sigma(S)|$ is the k-th Catalan number and therefore $|\Pi_\Sigma(S)| = O(4^k)$. The asymptotic enumeration of $|\Pi_\Sigma(S)|$ for general surfaces is quite a complicated problem. However, its behavior for surfaces Σ where $\gamma(\Sigma)$ and $\beta(\Sigma)$ are bounded is not significantly different from the disk in what concerns its exponential growth. In particular it holds that $\lim_{|S|\to\infty} |\Pi_\Sigma(S)|^{1/|S|} = 4$ and this is a consequence of the following enumerative result from [33].

Theorem 8.2 *Let Σ be a surface with boundary. Then the number $|\Pi_\Sigma(S)|$, for $|S| = k$, verifies*

$$|\Pi_\Sigma(S)| \leq_{k\to\infty} \frac{C(\Sigma)}{\Gamma\left(3/2\gamma(\Sigma) + \beta(\Sigma) - 3\right)} \cdot k^{3/2\gamma(\Sigma)+\beta(\Sigma)-4} \cdot 4^k \ , \tag{1}$$

where $C(\Sigma)$ is a function depending only on Σ that is bounded by $\gamma(\Sigma)^{O(\gamma(\Sigma))}$, and Γ is the Gamma function: $\Gamma(u) = \int_0^\infty t^{u-1}e^{-t}dt$.

The above result, which is critical for our analysis, has been proved using tools from analytic combinatorics (see [19]): singularity analysis over expressions obtained by the symbolic method. Actually, we prefer to translate it to the following looser form that is more convenient for our algorithmic purposes.

Corollary 8.3 *Let Σ be a surface with boundary and let S be a set of k vertices in the boundary of Σ. Let also γ be an integer such that $\gamma(\Sigma), \beta(\Sigma) \leq \gamma$. Then $|\Pi_\Sigma(S)| = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot 4^k$.*

For every set S we define $\mathcal{B}(S)$ as the collection of all its partitions. Recall that if $|S| = l$, then $|\mathcal{B}(S)|$ is the l-th Bell number and that $|\mathcal{B}(S)| = 2^{O(l \log l)}$. Also, given a collection $\mathcal{C} = \{S_1, \ldots, S_q\}$ of subsets of S and a subset $S' \subseteq S$, we denote by $\mathcal{C}|_{S'}$ the collection of all non-empty sets in $\{S_1 \cap S', \ldots, S_q \cap S'\}$. Clearly, if $\mathcal{C}$ is a partition of S, then $\mathcal{C}|_{S'}$ is a partition of S'.

Lemma 8.4 *Let Σ be a surface with boundary, let S be a set of vertices in the boundary of Σ, and let A be a set of apices. Let also γ and k be integers such that $|A|, \gamma(\Sigma), \beta(\Sigma) \leq \gamma$ and $|S| \leq k$. Then $|\Pi_{\Sigma^A}(S \cup A)| = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot \gamma^{O(k)}$.*

Proof: Let $\mathcal{R} \in \Pi_{\Sigma^A}(S \cup A)$ and let $\mathfrak{B}$ be an extended partition family for $(\Sigma^A, S \cup A)$, where $\mathcal{R}(\mathfrak{B}) = \mathcal{R}$. We define $\mathfrak{B}_\Sigma$ as the set of connected components of the set $(\mathbf{U}\mathfrak{B}) \cap \Sigma$. Notice that $\mathfrak{B}_\Sigma$ is a partition family for (Σ, S) and thus $\mathcal{R}_\Sigma = \mathcal{R}(\mathfrak{B}_\Sigma) \in \Pi_\Sigma(S)$. Notice also that $\mathcal{R}|_A$ is a member of $\mathcal{B}(A)$. We conclude that each $\mathcal{R} \in \Pi_{\Sigma^A}(S \cup A)$ uniquely generates a pair $(\mathcal{R}_\Sigma, \mathcal{R}|_A) \in \Pi_\Sigma(S) \times \mathcal{B}(A)$.

We define $\mathbf{P}_{(\mathcal{R}_\Sigma, \mathcal{R}|_A)}$ as the set of all possible $\mathcal{R}$'s in $\Pi_{\Sigma^A}(S \cup A)$ that can generate a given pair $(\mathcal{R}_\Sigma, \mathcal{R}|_A) \in \Pi_\Sigma(S) \times \mathcal{B}(A)$.

Claim 8.5 $|\mathbf{P}_{(\mathcal{R}_\Sigma, \mathcal{R}|_A)}| \leq (|\mathcal{R}|_A| + 1)^{|\mathcal{R}_\Sigma|}$.

Proof: We use the notation $\mathcal{R}|_A = \{A_1, \ldots, A_q\}$. Let $\mathcal{R} \in \mathbf{P}_{(\mathcal{R}_\Sigma, \mathcal{R}|_A)}$. By the above definitions, for each $i \in \{1, \ldots, p\}$, there is a unique set, say $P^{(i)}$, of $\mathcal{R}$ containing A_i as a subset. Moreover, there is a (possibly empty) subset, say $\mathcal{B}^{(i)}$, of $\mathcal{R}_\Sigma$ such that $P^{(i)} \setminus A_i = \mathbf{U}\mathcal{B}^{(i)}$. Notice that $\{\mathcal{B}^{(1)}, \ldots, \mathcal{B}^{(i)}\}$ is a packing of $\mathcal{R}_\Sigma$ (not necessarily a partition of $\mathcal{R}_\Sigma$, as some sets of $\mathcal{R}_\Sigma$ may appear directly as sets in $\mathcal{R}$). This means that each $\mathcal{R} \in \mathbf{P}_{(\mathcal{R}_\Sigma, \mathcal{R}|_A)}$ corresponds to some packing of $\mathcal{R}_\Sigma$ and some bijection of its sets to some of the elements of $\mathcal{R}|_A$. This corresponds to the partial functions from the set $\mathcal{R}_\Sigma$ to the set $\mathcal{R}|_A$, that is the claimed upper bound. □

The rest of the proof is based on the fact that

$$|\Pi_{\Sigma^A}(S \cup A)| \leq \sum_{\substack{(\mathcal{R}_\Sigma, \mathcal{R}|_A) \in \\ \Pi_\Sigma(S) \times \mathcal{B}(A)}} |\mathbf{P}_{(\mathcal{R}_\Sigma, \mathcal{R}|_A)}|.$$

Recall now that $|\mathcal{B}(A)| \leq |A|^{|A|} \leq \gamma^\gamma$. Also, from Corollary 8.3, it holds that $|\Pi_\Sigma(S)| = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot 4^k$. The Claim above implies that $\mathbf{P}_{(\mathcal{R}_\Sigma, \mathcal{R}|_A)} \leq (\gamma + 1)^k$, as every packing in $\Pi_\Sigma(S)$ has at most $|S| \leq k$ sets and every packing in $\mathcal{B}(A)$ has at most $|A| \leq \gamma$ sets. The proof of the lemma is completed by putting all these facts together. □

Let G be a graph and let S be a subset of $V(G)$. We define $\Pi_G(S)$ as the set of all partitions in $\Psi_G(S)$, formally,

$$\Pi_G(S) = \{\mathcal{R} \mid \mathcal{R} \in \Psi_G(S) \text{ and } \mathbf{U}\,\mathcal{R} = S\}.$$

Lemma 8.6 *Let G be a graph and let $S' \subseteq S \subseteq V(G)$. Then $|\Pi_G(S')| \leq |\Pi_G(S)|$.*

Proof: In order to prove the lemma, let us define an injective application $i : \Pi_G(S') \hookrightarrow \Pi_G(S)$. Let $\mathcal{R} \in \Pi_G(S')$, which implies by definition (see Section 3) that there exists a subgraph $H \subseteq G$ whose connected components define the packing $\mathcal{R}$ of S'. We define $i(\mathcal{R})$ as the packing of S given by the same subgraph H. It is then easy to check that if $\mathcal{R}_1, \mathcal{R}_2 \in \Pi_G(S')$ with $\mathcal{R}_1 \neq \mathcal{R}_2$, then $i(\mathcal{R}_1) \neq i(\mathcal{R}_2)$. □

Lemma 8.7 *Let G' be a graph with a set $S' \subseteq V(G')$ and an edge $e = \{x, y\}$ whose endvertices are both vertices of S'. Let also G be the graph obtained from G' after the contraction of e to a vertex v_e, and let $S = S' \setminus \{x, y\} \cup \{v_e\}$. Then $|\Pi_G(S)| \leq |\Pi_{G'}(S')|$.*

Proof: Similarly to the proof of Lemma 8.7, let us define an injection $i : \Pi_G(S) \hookrightarrow \Pi_{G'}(S')$. Let $\mathcal{R} \in \Pi_G(S)$, and let H be a subgraph of G whose connected components

define the packing $\mathcal{R}$ of S. We distinguish two cases. First, if $v_e \notin V(H)$, we define $i(\mathcal{R})$ to be the packing of S' given by the connected components of H. Otherwise, if $v_e \in V(H)$, let $H' \subseteq G'$ be the graph obtained from H by removing v_e and adding x, y, the edge $\{x, y\}$, and all the edges in G' between x, y and the neighbors of v_e in H. In this case we define $i(\mathcal{R})$ to be the packing of S' given by the connected components of H'. It is again easy to check that if $\mathcal{R}_1, \mathcal{R}_2 \in \Pi_G(S)$ with $\mathcal{R}_1 \neq \mathcal{R}_2$, then $i(\mathcal{R}_1) \neq i(\mathcal{R}_2)$. □

The following observation gives the obvious way to enumerate packings from partitions.

Observation 8.8 *Let G be a graph and let $S \subseteq V(G)$. Then $\Psi_G(S) = \bigcup_{S' \subseteq S} \Pi_G(S')$.*

Combining Lemma 8.6 and Observation 8.8 we obtain the following.

Observation 8.9 *Let G be a graph and let $S' \subseteq S \subseteq V(G)$. Then $|\Psi_G(S')| \leq |\Psi_G(S)|$.*

Let H be a graph embedded in a surface Σ with boundary. We denote by $\mathfrak{B}_H$ the collection of connected subsets of Σ corresponding to the connected components of H.

Lemma 8.10 *Let G be a graph containing a set A of vertices such that $G \setminus A$ is embedded in a surface Σ. Let also S be the set of vertices of G that lie on the boundary of Σ. Then $|\Pi_G(S \cup A)| \leq |\Pi_{\Sigma^A}(S \cup A)|$.*

Proof: It is enough to prove that for every partition $\mathcal{R}$ in $\Pi_G(S \cup A)$ there is an extended partition family $\mathfrak{B}$ for $(\Sigma^A, S \cup A)$ such that $\mathcal{R}(\mathfrak{B}) = \mathcal{R}$. For this, consider a subgraph H of G where $\mathcal{P}_{S \cup A}(H) = \mathcal{R}$. As $\mathcal{R} \in \Pi_G(S \cup A)$, it holds that $\mathbf{U}\mathcal{R} = S \cup A$ and therefore $\mathbf{U}\mathcal{R} \subseteq V(H)$. As $H \setminus A$ can be embedded in Σ, the set $\mathfrak{B}_{H \setminus A}$ is a partition family for (Σ, S). Let now H_A be the subgraph of H formed by its edges that are not embedded in Σ. Observe that H_A is isomorphic to a subgraph of C_A and therefore its edges can be seen as a collection $\mathfrak{E}_A$ of flying edges where each apex vertex is contained in some edge of $\mathfrak{E}_A$. Let $\mathfrak{B}$ be the connected components of the set $\mathbf{U}(\mathfrak{B}_{H \setminus A} \cup \mathfrak{E}_A)$. Clearly, $\mathfrak{B}$ is an extended partition family for $(\Sigma^A, S \cup A)$. It is now easy to verify that $\mathcal{R}(\mathfrak{B}) = \mathcal{R}$ and the lemma follows. □

Lemma 8.11 *Let G be a graph containing a set A of vertices such that $G \setminus A$ is embedded in a surface Σ with boundary. Let also S be the set of vertices of G that lie on the boundary of Σ and $A' \subseteq A$. Then, if $|S| \leq k$ and $|A|, \gamma(\Sigma), \beta(\Sigma) \leq \gamma$, then $|\Psi_G(S \cup A')| = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot \gamma^{O(k)}$.*

Proof: From Observation 8.9, it is enough to prove the lemma for the case where $A' = A$. From Lemmata 8.4 and 8.10, it follows that $|\Pi_G(S \cup A)| = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot \gamma^{O(k)}$. From Lemma 8.6, we obtain that $|\Pi_G(W)| \leq |\Pi_G(S \cup A)| = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot \gamma^{O(k)}$ for every $W \subseteq S \cup A$. Therefore, from Observation 8.8, $|\Psi_G(S \cup A)| \leq 2^{|S|+|A|} \cdot \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot \gamma^{O(k)} = \gamma^{O(\gamma)} \cdot k^{O(\gamma)} \cdot \gamma^{O(k)}$ and the lemma follows. □

Let Σ be a surface without boundary, and let $\mathcal{N}$ be a set of O-arcs in Σ pairwise intersecting at zero-dimensional subsets of Σ. Then the closure of each connected component of $\Sigma \setminus \mathbf{U}\mathcal{N}$ is called a *pseudo-surface*. Notice that the boundary of a pseudo-surface is a subset of $\mathcal{N}$ and that the definition of the parameter $\theta(\mathcal{N})$ introduced in Section 6 can

be naturally extended to pseudo-surfaces. If Σ is a pseudo-surface with boundary given by a finite set $\mathcal{N}$ of O-arcs pairwise intersecting at finite zero-dimensional subsets of Σ, note that Σ is a surface with boundary if and only if $\theta(\mathcal{N}) = 0$. Note also that the closure of each of the two connected components in the last condition of Definition 7.1 is a pseudo-surface.

Lemma 8.12 *Let G be a graph embedded in a pseudo-surface Σ whose boundary is given by a collection $\mathcal{N}$ of nooses of G pairwise intersecting only at vertices of G, and such that $\theta(\mathcal{N}) > 0$. Let S be the set of vertices of G that lie on the boundary of Σ. Then there is a graph G' embedded in a pseudo-surface Σ' with boundary given by a collection $\mathcal{N}'$ of nooses of G', such that*

- *$\theta(\mathcal{N}') = \theta(\mathcal{N}) - 1$;*
- *G is the result of the contraction of an edge in G';*
- *if S' is the set of vertices of G' that lie on the boundary of Σ', then $|S'| = |S| + 1$.*

Proof: Without loss of generality, let $v \in N_1 \cap \ldots \cap N_\ell$, with $N_1, \ldots, N_\ell \in \mathcal{N}$ and $\ell \geq 2$, so by assumption $v \in S \subseteq V(G)$; for an illustration throughout the proof, see Figure 3. We build from Σ a pseudo-surface Σ' by replacing noose N_1 with a noose N_1' obtained from N_1 by slightly deforming it around v in such a way that $v \notin N_1'$ (note that this is clearly possible, as by assumption the nooses intersect only at vertices of G). As the nooses in Σ and in Σ' intersect at the same vertices except for vertex v, we have that $\theta(\mathcal{N}') = \theta(\mathcal{N}) - 1$. We now construct G' from G as follows: We start from the embedding of G in Σ, and we embed it in Σ' in such a way that $v \in N_2 \cap \ldots \cap N_\ell$. Finally, we add a new vertex $v' \in N_1'$ and we add the edge $\{v, v'\}$. By construction, it is clear that G can be obtained from G' by contracting edge $\{v, v'\}$, and that $S' = S \cup \{v'\}$. □

Proof of Lemma 8.1: In case $|\mathbf{mid}(e) \setminus A| \leq 2$, we have that $|\mathbf{mid}(e)| = O(\gamma)$ and the result follows as $|\Pi_{G_e}(\mathbf{mid}(e))| \leq |\mathbf{B}(O(\gamma))| = 2^{O(\gamma \log \gamma)}$. In the remaining case, let H be the graph of the polyhedral decomposition $(\mathcal{G}, A)$ of G that corresponds to edge e. Let also $\mathcal{N}$ be the corresponding set of $O(\gamma)$ nooses meeting all vertices of $\mathbf{mid}(e) \setminus A$. Let also Σ^* be the closure of the connected component of $\Sigma \setminus \bigcup_{N \in \mathcal{N}} N$ where the graph $G_e \setminus A$ is embedded. Clearly, Σ^* is a pseudo-surface with boundary given by a set of nooses $\mathcal{N}$ with $\theta(\mathcal{N}) = O(\gamma)$. By inductively applying Lemmata 8.7 and 8.12, we can assume that Σ^* is a surface with boundary such that $O(|\mathbf{mid}(e)| + \gamma(\Sigma)) = O(k + \gamma)$ of the vertices of G_e lie on this boundary. Then the result follows directly from Lemma 8.11 by setting G_e instead of G, Σ^* instead of Σ, $A \cap \mathbf{mid}(e)$ instead of A', and $A \cap V(G_e)$ instead of A. □

9 Conclusions and open problems

As stated in Theorem 3.2, our results can be summarized as follows: Every connected packing-encodable problem whose input graph G is embedded in a surface of Euler genus γ, and has branchwidth at most k, can be solved in $\gamma^{O(k)} \cdot k^{O(\gamma)} \cdot \gamma^{O(\gamma)} \cdot n^{O(1)}$ steps.

As we mentioned, the problems tackled in [13] can be encoded with pairings, and therefore they can be seen as special cases of packing-encodable problems. As a result

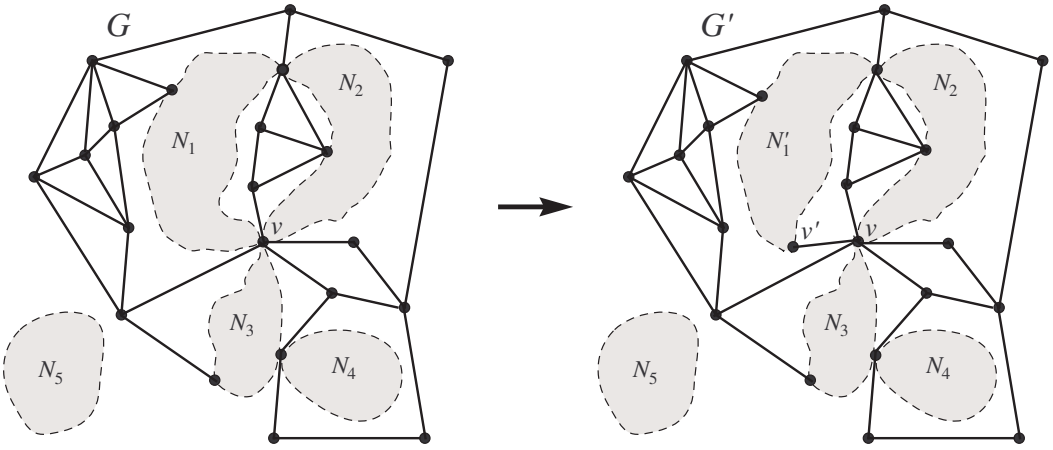

Figure 3: Example of the construction of Σ' and G' in the proof of Lemma 8.12. On the left, we have a graph G (depicted with thick lines) embedded in a pseudo-surface Σ whose boundary is given by the set of nooses $\mathcal{N} = \{N_1, N_2, N_3, N_4, N_5\}$ (in grey) pairwise intersecting at vertices of G, with $\theta(\mathcal{N}) = 4$. On the right, the corresponding graph G' embedded in a pseudo-surface Σ' with boundary given by $\mathcal{N}' = \{N'_1, N_2, N_3, N_4, N_5\}$, and such that $\theta(\mathcal{N}') = 3$. In this example, we have that $|S| = 6$ and $|S'| = 7$.

of this, we reproduce all the results of [13]. Moreover, as our approach does not use planarization, our analysis provides algorithms where the dependence on the Euler genus γ is better than the one in [13]. In particular, the running time of the algorithms in [13] is $2^{O(\gamma \cdot k + \gamma^2 \cdot \log k)} \cdot n^{O(1)}$, while in our case the running time is $2^{O(\log \gamma \cdot k + \gamma \cdot \log k + \gamma \cdot \log \gamma)} \cdot n^{O(1)}$.

Dynamic programming is important for the design of *subexponential* exact or parameterized algorithms. Using the fact that bounded-genus graphs have branchwidth at most $O(\sqrt{\gamma \cdot n})$ [21], we derive the existence of exact algorithms in $O^*(2^{O(\log \gamma \cdot \sqrt{\gamma n} + \gamma \cdot \log n + \gamma \cdot \log \gamma)}$ steps for all connected packing-encodable problems. Moreover, using bidimensionality theory (see [10,11]), one can derive $2^{O(\gamma \cdot \log \gamma \cdot \sqrt{k} + \gamma \cdot \log k)} \cdot n^{O(1)}$ time parameterized algorithms for all bidimensional connected packing-encodable problems, where here k is the corresponding parameter.

Note that the running time of our algorithms is conditioned by the construction of an appropriate surface cut decomposition. This preprocessing step takes $2^{3k+O(\log k)} \cdot n^3$ steps by Theorem 7.2. Finding a preprocessing algorithm with better polynomial dependance remains open. As finding an optimal branch decomposition of a surface-embedded graph in polynomial time is open, it may be even possible that computing an optimal surface cut decomposition can be done in polynomial time.

Sometimes dynamic programming demands even more complicated encodings. We believe that our results can also serve in this direction. For instance, surface cut decompositions have recently been used in [1] for minor containment problems, where tables encode partitions of packings of the middle sets.

A natural extension of our results is to consider more general classes of graphs than bounded-genus graphs. This has been done in [15] for problems where the tables of the algorithms encode pairings of the middle sets. Extending these results for connected packing-encodable problems (where tables encode subsets of the middle sets) using the

planarization approach of [15] appears to be a quite complicated task. We believe that our surface-oriented approach could be more successful in this direction and we find it an interesting, but non-trivial task [34].

Acknowledgement. We would like to thank Sergio Cabello for inspiring discussions and for pointing us to several helpful topological references.

References

[1] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos. Faster Parameterized Algorithms for Minor Containment. In *Proc. of the 12th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 6139 of *LNCS*, pages 322–333, 2010.

[2] E. Amir. Efficient approximation for triangulation of minimum treewidth. In *Proc. of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 7–15, 2001.

[3] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – a survey. *BIT*, 25(1):2–23, 1985.

[4] V. Arvind and P. Mukhopadhyay. Derandomizing the Isolation Lemma and Lower Bounds for Circuit Size. In *Proc. of APPROX-RANDOM*, volume 5171 of *LNCS*, pages 276–289, 2008.

[5] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Proc. of the 15th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 317 of *LNCS*, pages 105–118, 1988.

[6] S. Cabello and B. Mohar. Finding shortest non-separating and non-contractible cycles for topologically embedded graphs. *Discrete and Computational Geometry*, 37:213–235, 2007.

[7] L. Cai and D. W. Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003.

[8] B. Courcelle. The monadic second-order logic of graphs: definable sets of finite graphs. In *Proc. of the 14th International Workshop on Graph-theoretic Concepts in Computer Science (WG)*, volume 344 of *LNCS*, pages 30–53, 1988.

[9] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. Manuscript available at `http://lanl.arxiv.org/abs/1103.0534`.

[10] 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.

[11] E. D. Demaine, M. Hajiaghayi, and D. M. Thilikos. The Bidimensional Theory of Bounded-Genus Graphs. *SIAM Journal on Discrete Mathematics*, 20(2):357–371, 2006.

[12] R. Diestel. *Graph Theory*. Springer-Verlag, Berlin, 3rd edition, 2005.

[13] 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.

[14] F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. In *Proc. of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *LNCS*, pages 15–27, 2007.

[15] 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, Philadelphia, PA, USA, 2008.

[16] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Branch Decompositions. In *Proc. of the 13th Annual European Symposium on Algorithms (ESA)*, volume 3669 of *LNCS*, pages 95–106, 2005.

[17] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, New York, 1999.

[18] D. Eppstein. Dynamic Generators of Topologically Embedded Graphs. In *Proc. of the 14th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 599–608, 2003.

[19] P. Flajolet and R. Sedgewick. *Analytic Combinatorics.* Cambridge Univ. Press, 2008.

[20] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer Verlag, 2006.

[21] F. V. Fomin and D. M. Thilikos. Fast Parameterized Algorithms for Graphs on Surfaces: Linear Kernel and Exponential Speed-Up. In *Proc. of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *LNCS*, pages 581–592, 2004.

[22] F. V. Fomin and D. M. Thilikos. Dominating Sets in Planar Graphs: Branch-Width and Exponential Speed-Up. *SIAM Journal on Computing*, 36(2):281–309, 2006.

[23] F. V. Fomin and D. M. Thilikos. On Self Duality of Pathwidth in Polyhedral Graph Embeddings. *Journal of Graph Theory*, 55(42-54), 2007.

[24] M. R. Henzinger, S. Rao, and H. N. Gabow. Computing Vertex Connectivity: New Bounds from Old Techniques. *Journal of Algorithms*, 34(2):222–250, 2000.

[25] R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. Special issue of FOCS 1998.

[26] D. Lokshtanov, D. Marx, and S. Saurabh. Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal. In *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, page to appear, 2011.

[27] D. Lokshtanov, D. Marx, and S. Saurabh. Slightly Superexponential Parameterized Problems. In *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, page to appear, 2011.

[28] B. Mohar. A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999.

[29] B. Mohar and C. Thomassen. *Graphs on surfaces*. John Hopkins University Press, 2001.

[30] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[31] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[32] N. Robertson and P. Seymour. Graph minors. XII. Distance on a surface. *J. Combin. Theory Series B*, 64:240–272, 1995.

[33] J. Rué, I. Sau, and D. M. Thilikos. Asymptotic enumeration of non-crossing partitions on surfaces. Manuscript submitted for publication, temporarily available at `www.lirmm.fr/~sau/Pubs/RST11NCP.pdf`.

[34] J. Rué, I. Sau, and D. M. Thilikos. Dynamic programming for minor-free graphs. Manuscript in preparation, 2011.

[35] 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.

[36] P. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[37] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.

[38] C. Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989.

Appendix

Two examples of dynamic programming algorithms

In this Appendix we present two examples of typical dynamic programming algorithms on graphs of bounded branchwidth. The first algorithm solves the Vertex Cover problem, which is a problem whose solutions can be simply encoded by a subset of vertices. The second algorithm solves the Connected Vertex Cover problem, which is a packing-encodable problem, but cannot be encoded by neither a subset nor a pairing of vertices.

Dynamic programming for Vertex Cover**.** Given a graph G and a non-negative integer ℓ, we have to decide whether G contains a set $S \subseteq V(G), |S| \leq \ell$, meeting all edges of G.

Let G be a graph and $X, X' \subseteq V(G)$ where $X \cap X' = \emptyset$. We say that $\mathbf{vc}(G, X, X') \leq \ell$ if G contains a vertex cover S where $|S| \leq \ell$ and $X \subseteq S \subseteq V(G) \setminus X'$. Let $\mathcal{R}_e = \{(X, \ell) \mid X \subseteq \mathbf{mid}(e)$ and $\mathbf{vc}(G_e, X, \mathbf{mid}(e) \setminus X) \leq \ell\}$ and observe that $\mathbf{vc}(G) \leq \ell$ if and only if $(\emptyset, \ell) \in \mathcal{R}_{e_r}$. For each $e \in E(T)$ we can compute $\mathcal{R}_e$ by using the following dynamic programming formula:

$$\mathcal{R}_e = \begin{cases} \{(X, \ell) \mid X \subseteq e \text{ and } X \neq \emptyset \wedge \ell \geq |X|\} & \text{if } e \in L(T) \\ \{(X, \ell) \mid \exists (X_1, \ell_1) \in \mathcal{R}_{e_1}, \exists (X_2, \ell_2) \in \mathcal{R}_{e_2} : \\ (X_1 \cup X_2) \cap \mathbf{mid}(e) = X \wedge \ell_1 + \ell_2 - |X_1 \cap X_2| \leq \ell\} & \text{if } e \notin L(T) \end{cases}$$

Note that for each $e \in E(T)$, $|\mathcal{R}_e| \leq 2^{|\mathbf{mid}(e)|} \cdot \ell$. Therefore, the above algorithm can check whether $\mathbf{vc}(G) \leq \ell$ in $O(4^{\mathbf{bw}(G)} \cdot \ell^2 \cdot |V(T)|)$ steps. Clearly, this simple algorithm is single-exponential in $\mathbf{bw}(G)$. Moreover the above dynamic programming machinery can be adapted to many other combinatorial problems where the certificate of the solution is a (non-restricted) subset of vertices (e.g. Dominating Set, 3-Coloring, Independent Set, among others).

Dynamic programming for Connected Vertex Cover**.** Suppose now that we are looking for a *connected* vertex cover of size $\leq \ell$. Clearly, the above dynamic programming formula does not work for this variant as we should keep track of more information on X towards encoding the connectivity demand.

Let G be a graph, $X \subseteq V(G)$ and $\mathcal{H}$ be a (possibly empty) hypergraph whose vertex set is a subset of X, whose hyperedges are non-empty, pairwise non-intersecting, and such that each vertex of $\mathcal{H}$ belongs to some of its hyperedges (we call such a hypergraph *partial packing* of X). Suppose that $\mathcal{H}$ is a partial packing on $\mathbf{mid}(e)$. We say that $\mathbf{cvc}(G, \mathcal{H}) \leq \ell$ if G contains a vertex cover S where $|S| \leq \ell$ and such that if $\mathcal{C}$ is the collection of the connected components of $G_e[S]$, then either $|E(\mathcal{H})| = |\mathcal{C}|$ and $(X, \{X \cap V(C) \mid C \in \mathcal{C}\}) = \mathcal{H}$ or $E(\mathcal{H}) = \emptyset$ and $|\mathcal{C}| = 1$.

As before, let $\mathcal{Q}_e = \{(\mathcal{H}, \ell) \mid \mathbf{cvc}(G, \mathcal{H}) \leq \ell\}$ and observe that $\mathbf{cvc}(G) \leq \ell$ if and only if $((\emptyset, \emptyset), \ell) \in \mathcal{Q}_{e_r}$. The dynamic programming formula for computing $\mathcal{Q}_e$ for each $e \in E(T)$

is the following.

$$\mathcal{Q}_e = \begin{cases} \{(\mathcal{H},\ell) \mid \min\{\ell, |E(\mathcal{H})|+1\} \geq |V(\mathcal{H})| \geq 1 & \text{if } e \in L(T) \\ \{(\mathcal{H},\ell) \mid \exists (\mathcal{H}_1,\ell_1) \in \mathcal{Q}_{e_1}, \exists (\mathcal{H}_2,\ell_2) \in \mathcal{Q}_{e_2} : \\ V(\mathcal{H}_1) \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)) = V(\mathcal{H}_2) \cap (\mathbf{mid}(e_1) \cap \mathbf{mid}(e_2)), \\ (\mathcal{H}_1 \oplus \mathcal{H}_2)[\mathbf{mid}(e)] = \mathcal{H}, \ell_1 + \ell_2 - |V(\mathcal{H}_1) \cap V(\mathcal{H}_2)| \leq \ell\}, \\ \text{ if } E(\mathcal{H}) = \emptyset \text{ then } |E(\mathcal{H}_1 \oplus \mathcal{H}_2)| = 1, \text{and} \\ \text{if } E(\mathcal{H}) \neq \emptyset \text{ then } |E(H_1 \oplus H_2)| = |E(\mathcal{H})| & \text{if } e \notin L(T). \end{cases}$$

In the above formula, $\mathcal{H}_1 \oplus \mathcal{H}_2$ is the hypergraph with vertex set $V(\mathcal{H}_1) \cup V(\mathcal{H}_2)$ where each of its hyperedges contains the vertices of each of the connected components of $\mathcal{H}_1 \cup \mathcal{H}_2$.

Clearly, each $\mathcal{H}$ corresponds to a collection of subsets of X and the number of such collections for a given set $\mathbf{mid}(e)$ of r elements is given by the r-th Bell number of r, denoted by B_r. By taking the straightforward upper bound $|B_r| = 2^{O(r \log r)}$, we have that one can check whether an input graph G has a connected vertex cover of size at most ℓ in $2^{O(\mathbf{bw}(G) \cdot \log \mathbf{bw}(G))} \cdot \ell \cdot |V(T)|$ steps.

As the growth of B_r is not single-exponential, we cannot hope for a single-exponential (in $\mathbf{bw}(G)$) running time for the above dynamic programming procedure, and no deterministic algorithm is known for this problem running in time single-exponential in $\mathbf{bw}(G)$. The same problem appears for numerous other problems where further restrictions apply to their solution certificates. Such problems can be connected variants of problems encodable by a subset of vertices, and others such as Maximum Induced Forest, Maximum d-Degree-Bounded Connected Subgraph, Metric TSP, Maximum d-Degree-Bounded Connected Induced Subgraph and all the variants studied in [35], Connected Dominating Set, Connected r-Domination, Feedback Vertex Set, Connected Feedback Vertex Set, Maximum Leaf Spanning Tree, Maximum Full-Degree Spanning Tree, Steiner Tree, or Maximum Leaf Tree.