An FPT Algorithm for the Embeddability of Graphs into Two-Dimensional Simplicial Complexes

Éric Colin de Verdière, Thomas Magnard

An FPT Algorithm for the Embeddability of Graphs into Two-Dimensional Simplicial Complexes

Éric Colin de Verdière
LIGM, CNRS, Univ Gustave Eiffel, F-77454 Marne-la-Vallée, France

Thomas Magnard
LIGM, CNRS, Univ Gustave Eiffel, F-77454 Marne-la-Vallée, France

Abstract

We consider the embeddability problem of a graph G into a two-dimensional simplicial complex C: Given G and C, decide whether G admits a topological embedding into C. The problem is NP-hard, even in the restricted case where C is homeomorphic to a surface.

It is known that the problem admits an algorithm with running time $f(c)n^{O(c)}$, where n is the size of the graph G and c is the size of the two-dimensional complex C. In other words, that algorithm is polynomial when C is fixed, but the degree of the polynomial depends on C. We prove that the problem is fixed-parameter tractable in the size of the two-dimensional complex, by providing a deterministic $f(c)n^3$-time algorithm. We also provide a randomized algorithm with expected running time $2^{c^{O(1)}}n^{O(1)}$.

Our approach is to reduce to the case where G has bounded branchwidth via an irrelevant vertex method, and to apply dynamic programming. We do not rely on any component of the existing linear-time algorithms for embedding graphs on a fixed surface; the only elaborated tool that we use is an algorithm to compute grid minors.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph algorithms; Mathematics of computing → Graphs and surfaces; Mathematics of computing → Topology

Keywords and phrases computational topology, embedding, simplicial complex, graph, surface, fixed-parameter tractability

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.32

Related Version *Full Version*: `https://arxiv.org/abs/2107.06236`

Funding Partially supported by the ANR projects Min-Max (ANR-19-CE40-0014) and SoS (ANR-17-CE40-0033).

Acknowledgements We would like to thank Arnaud de Mesmay for useful discussions.

1 Introduction

An embedding of a graph G into a host topological space X is a crossing-free topological drawing of G into X. The use and computation of graph embeddings is central in the communities of computational topology, topological graph theory, and graph drawing. A landmark result is the algorithm of Hopcroft and Tarjan [12], which allows to decide whether a given graph is planar (has an embedding into the plane) in linear time. Related results include more planarity testing algorithms [21], algorithms for embedding graphs on surfaces [18, 13] and for computing book embeddings [17], Hanani-Tutte theorems [24], and the theory of crossing numbers and planarization [2].

In this paper, we describe algorithms for deciding the embeddability of graphs into topological spaces that are, in a sense, as general as possible: two-dimensional simplicial complexes (or 2-complexes for brevity), which are made from vertices, edges, and triangles glued together. (We remark that every graph is embeddable in $\mathbb{R}^3$, so considering higher-dimensional simplicial complexes is irrelevant.) In a previous article, jointly written with

Mohar [6], we proved that, given a graph G and a 2-complex $\mathscr{C}$, one can decide whether G embeds into $\mathscr{C}$ in polynomial time for fixed $\mathscr{C}$; but the algorithm has running time $f(c) \cdot n^{O(c)}$, where n and c are the respective sizes of G and $\mathscr{C}$. Using a very different strategy, we describe algorithms for this problem, proving that it is fixed-parameter tractable in the complexity of the input complex:

▶ **Theorem 1.1.** *One can solve the embeddability problem of graphs into 2-dimensional simplicial complexes in deterministic $f(c)n^3$ time or in expected time $2^{c^{O(1)}}n^{O(1)}$, where c is the number of simplices of the input 2-complex, n is the number of vertices and edges of the input graph, and f is some computable function of c.*

2-complexes are much more general than surfaces, and tools that are suitable for studying embeddability of graphs on surfaces do not generalize. For example, the set of graphs embeddable on a given 2-complex is not closed under minor, which makes many tools for dealing with graphs on surfaces unsuitable for 2-complexes. Moreover, the complexity of some topological problems increase drastically when we consider 2-complexes instead of surfaces, e.g., deciding homeomorphism [20], or deciding the contractibility of curves [8, 16, 10]. Some other topological problems, such as the existence of a drawing a graph with at most k crossings in the plane or in a surface [14], can be recast as deciding whether the graph embeds on a certain 2-complex. For more detailed motivations, see [6, Introduction].

Comparison with previous works on surfaces

Since every surface is homeomorphic to a 2-complex, our problem has been largely considered in the special case where the input 2-complex is (homeomorphic to) a surface. That restricted problem is NP-hard [26], but several algorithms that are fixed-parameter tractable in the genus have been given, which we review now.

Mohar [18] has given an algorithm for this purpose that takes linear time in the input graph, for every fixed surface. This algorithm is very technical and relies on several other articles. The dependence on the genus is not made explicit, but seems to be doubly exponential [13].

Kawarabayashi et al., in an extended abstract [13], have given a simpler linear-time algorithm for this problem, but not all details are presented, which makes the approach hard to check [15, p. 3657, footnote].

General graph minor theory provides an algorithm for the same purpose. The graph minor theorem by Robertson and Seymour [23] implies that, for every fixed surface $\mathscr{S}$, there is a finite list of graphs $\mathcal{O}_{\mathscr{S}}$ such that a graph G can be embedded on $\mathscr{S}$ if and only if G does not contain any graph in $\mathcal{O}_{\mathscr{S}}$ as a minor. Moreover, there is an algorithm that given any surface $\mathscr{S}$ (specified by its genus and orientability) outputs the list $\mathcal{O}_{\mathscr{S}}$ [1], and there is an algorithm to decide whether a graph M is a minor of another graph G running, for fixed M, in time cubic in the size of G [22][7, Theorem 6.12]. These considerations thus lead to an algorithm to decide embeddability of a graph on a surface that runs, if the input surface is fixed, in cubic time in the size of the input graph.

Finally, in the same vein, Kociumaka and Pilipczuk [15] have studied the following more general problem than the embeddability problem of graphs on surfaces: Given a surface $\mathscr{S}$, a graph G, and an integer $k \geq 0$, is it possible to remove a set U of at most k vertices from G so that $G - U$ is embeddable on $\mathscr{S}$? They provide an algorithm that is fixed-parameter tractable in k and the genus of $\mathscr{S}$, where the dependence on the genus is unspecified. In particular, as a special case, they decide the embeddability of a graph on a surface; however, they use one of the previous algorithms [18, 13] as a subroutine. The problem that we study, the embeddability of graphs on 2-complexes, is independent from the problem studied by Kociumaka and Pilipczuk, in the sense that there is, a priori, no obvious reduction from one problem to the other. However, we will reuse some ingredients from that paper.

Our algorithms, restricted to the case where we want to embed graphs on surfaces, are not as efficient as the existing algorithms mentioned above. Indeed, the deterministic one runs in cubic time in the size of the input graph (for a fixed complex); the dependence on the size of the complex is not made explicit, because the algorithm uses, as a subroutine, an algorithm to compute grid minors [22]. The second algorithm is randomized, because it uses an algorithmic version of the excluded grid theorem [3] that uses randomness; for every fixed surface, it runs in expected time that is a polynomial of fixed (but large) degree in the size of the input graph. However, our algorithms are independent from the existing algorithms for embedding graphs on surfaces; the only elaborated tool that we use is an algorithm to compute grid minors [3, 22].

Overview and structure of the paper

We use a standard strategy in graph algorithms and parameterized complexity (see, e.g., the book by Cygan et al. [7, Chapter 7]): we show by dynamic programming that the problem can be solved efficiently for graphs of bounded branchwidth, and then, using an irrelevant vertex method, we prove that one can assume without loss of generality that the input graph G has branchwidth bounded by a polynomial in the size of the input 2-complex. In the context of surface-embedded graphs, this paradigm has been used in the extended abstract by Kawarabayashi et al. [13] and in the article by Kociumaka and Pilipczuk [15]; our algorithm takes inspiration from the former, for the idea of the dynamic programming algorithm, and from the latter, for some arguments in the irrelevant vertex method. However, handling 2-complexes requires significantly more effort. More precisely, Theorem 1.1 follows immediately from the following two theorems.

▶ **Theorem 1.2** (algorithm for bounded branchwidth)**.** *One can solve the embeddability problem of graphs into two-dimensional simplicial complexes in time $(c+w)^{O(c+w)}n$, where c is the number of simplices of the input 2-complex, and where n and w are the number of vertices and edges and the branchwidth of the input graph, respectively.*

▶ **Theorem 1.3** (algorithm to reduce branchwidth)**.** *Let $\mathscr{C}$ be a 2-complex with c simplices, and G a graph with n vertices and edges. We can correctly report that G is embeddable on $\mathscr{C}$, or correctly report that G is not embeddable on $\mathscr{C}$, or compute a subgraph H of G, of branchwidth polynomial in the number of simplices of $\mathscr{C}$, such that G embeds on $\mathscr{C}$ if and only if H does:*

- *in deterministic time $f(c)\cdot n^3$ for some computable function f,*
- *or in expected polynomial time.*

We now present the structure of the paper, indicating which techniques are used. We also emphasize which components would be simpler if we were just aiming for an algorithm for embedding graphs on surfaces.

We introduce some standard notions in Section 2.

Then, in Section 3, we show that we can make some simple assumptions on the input, and present data structures for representing 2-complexes and graphs embedded on them. If we restrict ourselves to the case where the input 2-complex is homeomorphic to a surface, we essentially consider combinatorial maps of graphs on surfaces, except that the graphs need not be cellularly embedded. The case of 2-complexes is largely more involved.

In Section 4, we show that if our input graph G has an embedding into our input 2-complex $\mathscr{C}$, then there exists an embedding of G on $\mathscr{C}$ that is *sparse* with respect to a branch decomposition of G. This means that each subgraph of G induced by the leaves

of any subtree of the branch decomposition can be separated from the rest of G using a graph embedded on $\mathscr{C}$, called *partitioning graph*, of small complexity. We find that this new structural result, even in the surface case, is interesting and can prove useful in other contexts. If the target space were a surface, we could assume that G is 3-connected and has no loop or multiple edges, which would imply (still with some work) that *any* embedding of G would be sparse, but again the fact that we consider 2-complexes requires additional work.

In Section 5, we present the dynamic programming algorithm, which either determines the existence of an embedding of G on $\mathscr{C}$, or shows that no sparse embedding of G on $\mathscr{C}$ exists (and thus no embedding at all, by the previous paragraph). The idea is to use bottom-up dynamic programming and to consider all regions of the 2-complex in which the subgraph of G (induced by a subtree of the branch decomposition) can be embedded. The complexity depends exponentially on the branchwidth of G.

The previous arguments, most notably in Section 4, implicitly assumed that, if G has an embedding into $\mathscr{C}$, it has a *proper and cellular embedding*, in particular, in which the faces are homeomorphic to disks. In Section 6, we show that we can assume this property. Essentially, we build all 2-complexes "smaller" than $\mathscr{C}$, such that G embeds on $\mathscr{C}$ if and only if it embeds into (at least) one of these 2-complexes, and moreover if it is the case, it has an embedding into (at least) one of these 2-complexes that is proper and cellular. If $\mathscr{C}$ were an orientable surface, we would just consider the surfaces of lower genus; but here a more sophisticated approach is needed.

The above ingredients allow to prove Theorem 1.2 (Section 7).

In Section 8, we show, using an irrelevant vertex method, that we can assume that G has branchwidth polynomial in the size of $\mathscr{C}$ (Theorem 1.3).

2 Preliminaries

2.1 Graphs and branch decompositions

In this paper, graphs may have loops and multiple edges unless noted otherwise. Let G be a graph; as usual, we denote by $V(G)$ and $E(G)$ the sets of vertices and edges of G.

A **(rooted) branch decomposition** of G is a rooted tree $\mathscr{B}$ in which:

- every node has degree either one (it is a **leaf**) or three (it is an **internal node**),
- the root is a leaf,
- each non-root leaf is labelled with an edge of G, and this labelling induces a bijection.

The vertices and edges of $\mathscr{B}$ are called **nodes** and **arcs**, respectively. Each arc α of $\mathscr{B}$ splits the tree $\mathscr{B}$ into two subtrees $\mathscr{B}_1$ and $\mathscr{B}_2$; if, for $i = 1, 2$, we denote by E_i the set of labels appearing in T_i, we see that α naturally induces a partition (E_1, E_2) of the set of edges of G (if α is the arc incident to the root, then one part of the partition is empty). The **middle set** associated to α is the set of vertices of G which are the endpoints of at least one edge in E_1 and at least one edge in E_2. The **width** of $\mathscr{B}$ is the maximum size of a middle set associated to an arc of $\mathscr{B}$. The **branchwidth** of G is the minimum width of its (rooted) branch decompositions.

The usual definition of a branch decomposition is identical, except that the tree is unrooted, and thus the leaves are in bijection with the edges of G. Our definition turns out to be more convenient to use in the dynamic program. The difference is cosmetic, as one can transform one kind of branch decomposition into the other easily while preserving the width.

2.2 Surfaces

We will assume some familiarity with surface topology; see, e.g., [19, 25, 4] for suitable introductions under various viewpoints. We recall some basic definitions and properties. A **surface** (without boundary) $\mathscr{S}$ is a compact, connected Hausdorff topological space in which every point has an open neighborhood homeomorphic to the open disk. Up to homeomorphism, every surface $\mathscr{S}$ is obtained from a sphere by:

- either removing $g/2$ open disks and attaching a handle (a torus with an open disk removed) to each resulting boundary component, where g is an even, nonnegative integer called the *(Euler)* **genus** of $\mathscr{S}$; in this case, $\mathscr{S}$ is **orientable**;
- or removing g open disks and attaching a Möbius band to each resulting boundary component, for a positive number g called the (non-orientable) **genus** of $\mathscr{S}$; in this case, $\mathscr{S}$ is **non-orientable**.

A **possibly disconnected surface** is a disjoint union of surfaces.

A **surface with boundary** is obtained from a surface (without boundary) by removing a finite set of interiors of disjoint closed disks. The boundary of each disk forms a **boundary component** of $\mathscr{S}$. The **genus** of $\mathscr{S}$ is defined as the genus of the original surface without boundary. Equivalently, a surface with boundary is a compact, connected Hausdorff topological space in which every point has an open neighborhood homeomorphic to the open disk or the closed half disk $\{(x, y) \in \mathbb{R}^2 \mid y \geq 0, x^2 + y^2 < 1\}$.

2.3 2-complexes

A **2-complex** (or two-dimensional simplicial complex) is an abstract simplicial complex of dimension at most two: a finite set of 0-simplices called **vertices**, 1-simplices called **edges**, and 2-simplices called **triangles**. Each edge is a pair of vertices, and each triangle is a triple of vertices; moreover, each subset of size two in a triangle must be an edge.

Each 2-complex $\mathscr{C}$ corresponds naturally to a topological space, obtained as follows: Start with one point per vertex in $\mathscr{C}$; connect them by segments as indicated by the edges in $\mathscr{C}$; similarly, for every triangle in $\mathscr{C}$, create a triangle whose boundary is made of the three edges contained in that triangle. By abuse of language, we identify $\mathscr{C}$ with that topological space.

2.4 Graph embeddings

Each graph has a natural associated topological space (for graphs without loops or multiple edges, this is a specialization of the definition for 2-complexes). An **embedding** Γ of a graph G into a 2-complex $\mathscr{C}$ is an injective continuous map from (the topological space associated to) G to (the topological space associated to) $\mathscr{C}$. A **face** of Γ is a connected component of the complement of the image of Γ in $\mathscr{C}$.

3 2-complexes and their data structures

3.1 Some preprocessing

A **3-book** is a topological space obtained from three triangles by considering one side per triangle and identifying these three sides together into a single edge. We say that a 2-complex $\mathscr{C}$ **contains a 3-book** if $\mathscr{C}$ contains three distinct triangles that share a common edge. Because every graph can be embedded into a 3-book [6, Proposition 3.1], we have the following proposition:

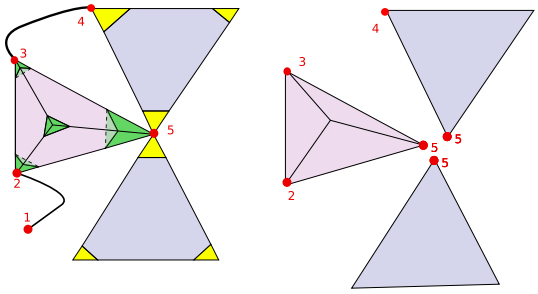

Figure 1 On the left: A 2-complex with 5 singular points, numbered from 1 to 5, and 2 isolated edges (one between 3 and 4 and one between 1 and 2) where, at singular points, the cones are in green and the corners in yellow. On the right: the corresponding detached surface.

▶ **Proposition 3.1.** *To decide the embeddability of a graph G on a 2-complex $\mathscr{C}$, we can without loss of generality, after a linear-time preprocessing, assume the following properties on the input:*

- *$\mathscr{C}$ has no 3-book and no connected component that is reduced to a single vertex;*
- *G has no connected component reduced to a single vertex, and at most one connected component homeomorphic to a segment.*

In the rest of this article, without loss of generality, we implicitly assume that $\mathscr{C}$ and G satisfy the properties stated in Proposition 3.1.

3.2 Structure of 2-complexes without 3-book or isolated vertex

Let $\mathscr{C}$ be a 2-complex without 3-book or isolated vertex, and let p be a vertex of $\mathscr{C}$. Following [6, Section 2.2], we describe the possible neighborhoods of p in $\mathscr{C}$. A **cone at** p is a cyclic sequence of triangles $t_1, \dots, t_k, t_1$ ($k \geq 3$), all incident to p, such that, for each $i = 1, \dots, k$, the triangles t_i and t_{i+1} (where $t_{k+1} = t_1$) share an edge incident with p, and any other pair of triangles have only p in common. A **corner at** p is a sequence of distinct triangles $t_1, \dots, t_k$, all incident to p, such that, for each $i = 1, \dots, k-1$, the triangles t_i and t_{i+1} share an edge incident with p, any other pair of these triangles have only p in common, and no other triangle in $\mathscr{C}$ shares an edge incident with p and belonging to one of $t_1, \dots, t_k$. An **isolated segment at** p is an edge incident to p but not incident to any triangle. The cones, corners, and isolated segments at p form the **link components** at p.

The set of edges and triangles incident with a given vertex p of $\mathscr{C}$ are uniquely partitioned into cones, corners, and isolated segments. We say that p is a **regular point** if all the edges and triangles incident to p form a single cone or corner; in that case, p has an open neighborhood homeomorphic to a disk or a closed half-disk. Otherwise, p is a **singular point**. See Figure 1, left, for an illustration.

Detaching a singular point p in $\mathscr{C}$ consists of the following operation: replace p with new vertices, one for each cone, corner, and isolated segment at p. Detaching all singular points of a 2-complex (without 3-book) yields a disjoint union of (1) isolated segments and (2) a surface, possibly disconnected, possibly with boundary, called the **detached surface** (see Figure 1, right). The trace of the singular points on the detached surface are the **marked points**. Conversely, $\mathscr{C}$ can be obtained from a surface (possibly disconnected, possibly with boundary) and a finite set of segments by choosing finitely many subsets of points and identifying the points in each subset together.

The **boundary** of $\mathscr{C}$ is the closure of the set of points of $\mathscr{C}$ that have an open neighborhood homeomorphic to a closed half-plane. Equivalently, it is the union of the edges of $\mathscr{C}$ incident with a single triangle.

3.3 Topological data structure for 2-complexes

Any 2-complex $\mathscr{C}$ without 3-book or isolated vertex is obtained from the detached surface and a set of disjoints segments, by identifying finitely many finite subsets of points. It is thus easy to describe a **topological data structure** for storing 2-complexes, by storing the topology of the detached surface and the segments, and recording additionally the identification of points to obtain the complex. The **size** of $\mathscr{C}$ is the sum of the number of isolated segments, the number of connected components of the detached surface, the total genus of the detached surface, the total number of boundary components of the detached surface, and the total number of marked points of the detached surface (the occurrences of the singular points). This is, up to a constant factor, the size of the topological data structure indicated above, if the genus is stored in unary. Any 2-complex described in the usual combinatorial way can be converted in polynomial time into this representation. Thus, **in the rest of this article, without loss of generality, we implicitly assume that $\mathscr{C}$ is given in the form of the above topological data structure.**

Moreover, given two 2-complexes in the form above, deciding whether they are homeomorphic essentially amounts to testing whether the corresponding data structures are isomorphic, which leads to the following lemma (the running time might be improvable, but this suffices for our purposes):

▶ **Lemma 3.2.** *Given two 2-complexes $\mathscr{C}$ and $\mathscr{C}'$, given in the topological representation above, of sizes c and c' respectively, we can decide whether $\mathscr{C}$ and $\mathscr{C}'$ are homeomorphic in time $(c+c')^{O(c+c')}$.*

3.4 Proper and cellular graph embeddings on 2-complexes

Let $\mathscr{C}$ be a 2-complex with size c, G a graph, and Γ an embedding of G on $\mathscr{C}$. The embedding Γ is **proper** if:

- the image of Γ meets the boundary of $\mathscr{C}$ only on singular points;
- the vertices of Γ cover the singular points of $\mathscr{C}$.

The embedding Γ is **cellular** if each face of Γ is an open disk plus possibly some part of the boundary of $\mathscr{C}$. We emphasize that this definition slightly departs from the standard one. Moreover, we will only consider cellular embeddings that are proper.

We will use a data structure to store possibly non-cellular embeddings of graphs on surfaces [5, Section 2.2]. Such a data structure is based on the gem representation of cellular graph embeddings [9, Section 2], but store additional information about the topology of the faces. It is important to remark that this data structure also allows to recover the topology of the underlying surface.

Let Γ be a proper graph embedding of a graph G on a 2-complex $\mathscr{C}$ (under the assumptions of Proposition 3.1). Let $\mathscr{S}$ be the detached surface of $\mathscr{C}$. Because Γ is proper, it naturally induces an embedding Γ', of another graph G', on $\mathscr{S}$; some vertices of G located on singular points of $\mathscr{C}$ are duplicated in G', the vertices of G located in the relative interior of isolated segments are absent from G', and the edges of G not in G' are edges on the isolated segments of $\mathscr{C}$. Our data structure, called **combinatorial map**, for storing the graph embedding Γ and the 2-complex $\mathscr{C}$ consists of storing (1) the graph embedding Γ' on $\mathscr{S}$, as indicated in the previous paragraph, (2) the isolated segments of $\mathscr{C}$, together with, for each such isolated segment, an ordered list alternating vertices and edges of Γ (or, instead of an edge, a mark indicating the absence of such an edge in the region of the isolated segment between the incident vertices), (3) the identifications of vertices of Γ' that are needed to recover Γ (and thus implicitly $\mathscr{C}$).

Isomorphisms between combinatorial maps are defined in the obvious way, similar to the concept of isomorphism between topological data structures: Two combinatorial maps are isomorphic if there is an isomorphism between the combinatorial maps restricted to the detached surfaces, isomorphisms between the maps on each isolated segments, and such that incidences are preserved on the singular points. We can easily test isomorphism between two combinatorial maps of size k and k', respectively, in $(k+k')^{O(k+k')}$ time.

We will need an algorithm to enumerate all proper embeddings of small graphs on a given 2-complex. This is achieved by a brute-force algorithm:

▶ **Lemma 3.3.** *Let $\mathscr{C}$ be a 2-complex of size c and k an integer. We can enumerate the $(c+k)^{O(c+k)}$ combinatorial maps of graphs with at most k vertices and at most k edges properly embedded on $\mathscr{C}$ in $(c+k)^{O(c+k)}$ time.*

4 Partitioning graphs

Let $\mathscr{C}$ be a 2-complex and G a graph, which satisfy the properties of Proposition 3.1. In this section, we lay the structural foundations of the dynamic programming algorithm, described in the next section (Proposition 5.1). The goal, in this section and the following one, is to obtain an algorithm that takes as input $\mathscr{C}$ and G, and, in time FPT in the size of $\mathscr{C}$ and the branchwidth of G, reports correctly one of the following two statements:

- G has no proper cellular embedding on $\mathscr{C}$,
- G has an embedding on $\mathscr{C}$.

This algorithm uses dynamic programming on a rooted branch decomposition of G. When processing a node of the rooted branch decomposition, it considers embeddings of the subgraph of G induced by the edges in the leaves of the subtree rooted at that node in a region of $\mathscr{C}$. This region will be delimited by a *partitioning graph* embedded on $\mathscr{C}$. Our dynamic program will roughly guess the partitioning graph at each node of the rooted branch decomposition. For this purpose, we need that, if G has a proper cellular embedding on $\mathscr{C}$, it has such an embedding that is *sparse*: at each node of the rooted branch decomposition of G, the partitioning graph corresponding to the embedding of the induced subgraph is small (its size is upper-bounded by a function of the branchwidth of G and of the size of $\mathscr{C}$). The goal of this section is to prove that this is indeed the case.

Let $(E_1, \dots, E_k)$ be an (ordered) partition of the edge set $E(G)$ of G. (We will only use the cases $k = 2$ or $k = 3$.) The **middle set** of $(E_1, \dots, E_k)$ is the set of vertices of G whose incident edges belong to at least two sets E_i.

Let Γ be a proper cellular embedding of G on $\mathscr{C}$. Since Γ is cellular, every boundary of $\mathscr{C}$ is incident to at least one vertex of Γ. Let $\hat{\Gamma}$ be obtained from Γ by adding edges as follows: for any pair of vertices u and v of Γ consecutive along a given boundary component of $\mathscr{C}$, we connect u and v via a new edge that runs along the boundary component. For each (ordered) partition $(E_1, \dots, E_k)$ of the edge set of G, we let $\hat{E}_1$ be the union of E_1 and of these new edges, and $\hat{E}_i = E_i$ for each $i \neq 1$; thus, $(\hat{E}_1, \dots, \hat{E}_k)$ is a partition of the set of edges of $\hat{\Gamma}$.

The **partitioning graph** $\Pi(\Gamma, E_1, \dots, E_k)$ (or more concisely Π) associated to Γ and $(E_1, \dots, E_k)$ is a graph properly embedded on $\mathscr{C}$ (but possibly non-cellularly), with labels on its faces, defined as follows:

- The vertex set of Π is the union of the singular points of $\mathscr{C}$ and of (the images under Γ of) the middle set of $E_1, \dots, E_k$.
- The relative interiors of the edges of Π are disjoint from the edges of $\hat{\Gamma}$ and from the isolated segments of $\mathscr{C}$. Let f be a face of $\hat{\Gamma}$ (which is homeomorphic to an open disk plus possibly some points of the boundary of $\mathscr{C}$). Let us describe the edges of Π inside f.

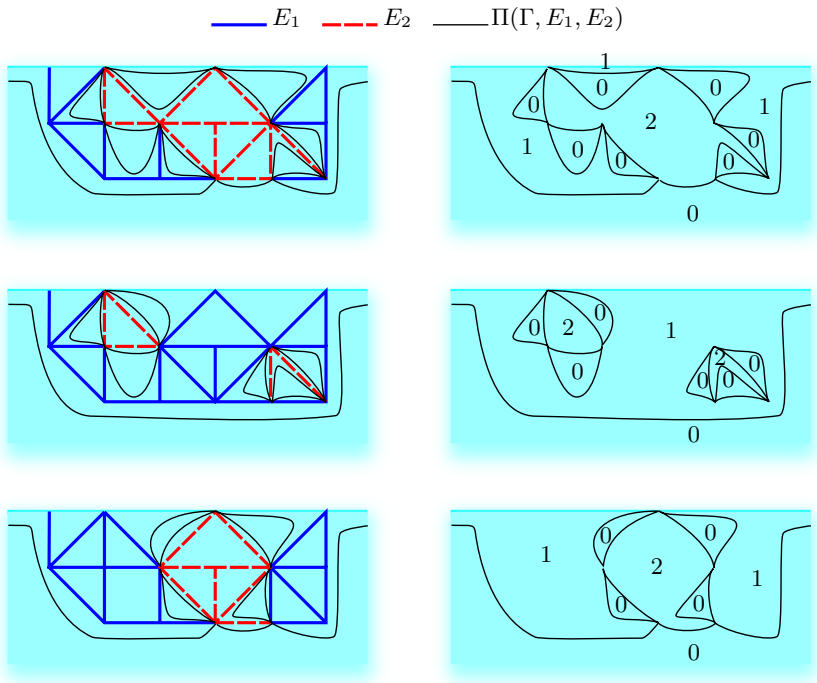

Figure 2 Construction of the partitioning graph $\Pi = \Pi(\Gamma, E_1, E_2)$, for three choices of the partition (E_1, E_2) of the same embedding Γ. Only a part of the 2-complex $\mathscr{C}$ is shown, with a boundary at the upper part, and without singular point. Left: The graph embeddings Γ (in thick lines) and Π (in thin lines). Right: The sole graph Π, together with the labelling of its faces.

If, for some $i \in \{1, \dots, k\}$, the boundary of f is comprised only of edges of $\hat{\Gamma}$ that lie in a single set $\hat{E}_i$, then Π contains no edge inside f. Otherwise, the boundary of f is a succession of edges of $\hat{E}_1$, $\hat{E}_2$, …, $\hat{E}_k$. The edges of Π inside f run along the boundary of f; for each $i \in \{1, \dots, k\}$, for each (maximal) group of consecutive edges in $\hat{E}_i$ along the boundary of f, we create an edge of Π that runs along this group, with endpoints the corresponding vertices on the boundary of f (see Figures 2 and 3). These vertices either are in the middle set of $(E_1, \dots, E_k)$, or lie on the boundary of $\mathscr{C}$ (and thus on singular points of $\mathscr{C}$).

It follows from the construction that $\hat{\Gamma}$ and Π intersect only at common vertices.

- Each face of Π is labelled by an integer in $\{0, \dots, k\}$ as follows: faces of Π containing edges in $\hat{E}_i$ are labelled i, and the other ones are labelled 0. By construction of the graph Π, each face of Π contains edges from at most one set $\hat{E}_i$, so this labelling is well defined.

Henceforth, we fix a rooted branch decomposition $\mathscr{B}$ of G, the root of which is denoted by ρ. Every arc α of $\mathscr{B}$ naturally partitions $E(G)$ into two parts E_1 and E_2, in which E_1 is the part on the same side as ρ; this (ordered) partition is the **edge partition associated to** α. Recall that Γ is a proper and cellular embedding of G on $\mathscr{C}$; we let $\Pi(\Gamma, \alpha)$ be $\Pi(\Gamma, E_1, E_2)$. Similarly, every node ν of $\mathscr{B}$ naturally partitions $E(G)$ into three parts E_1, E_2, and E_3, in which E_1 is the part on the same side as ρ; this partition is the **edge partition associated to** ν; we let $\Pi(\Gamma, \nu)$ be $\Pi(\Gamma, E_1, E_2, E_3)$.

We say that Γ is **sparse** (with respect to $\mathscr{B}$) if the following conditions hold, letting c be the size of $\mathscr{C}$ and w the width of $\mathscr{B}$:

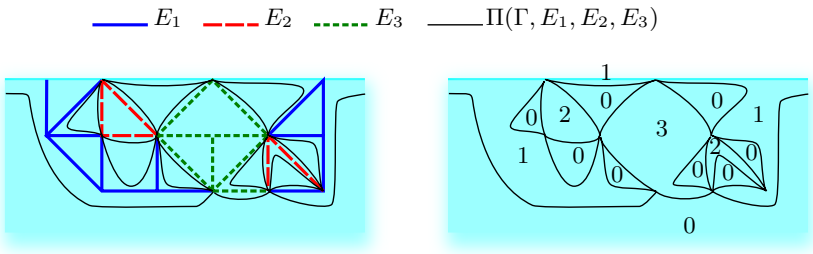

Figure 3 The partitioning graph $\Pi = \Pi(\Gamma, E_1, E_2, E_3)$. Left: The graph embeddings Γ (in thick lines) and Π (in thin lines). Right: The sole graph Π, together with the labelling of its faces.

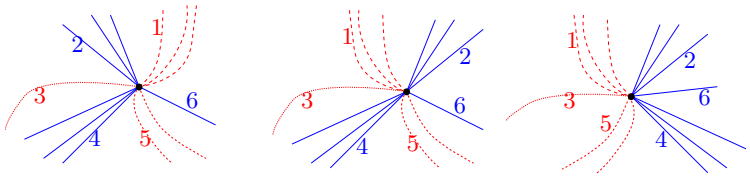

Figure 4 Left: A vertex with 6 intervals, numbered from 1 to 6. Middle: The cyclic order obtained by applying the first type of simplification operation on intervals 1 and 2. After the simplification, the intervals 1 and 3 are merged into a single one, and similarly for the intervals 2 and 6. Right: The cyclic order obtained by applying the second type of simplification to the configuration on the left, on pairs of intervals $\{1, 2\}$ and $\{4, 5\}$. After the simplification, the intervals 1, 3, and 5 are merged, and similarly for the intervals 2, 6, and 4.

- For each arc α of $\mathscr{B}$, the graph $\Pi(\Gamma, \alpha)$ has at most $74c + 26w$ edges;
- similarly, for each internal node ν of $\mathscr{B}$, the graph $\Pi(\Gamma, \nu)$ has at most $3(74c + 26w)$ edges.

The result of this section is the following.

▶ **Proposition 4.1.** *Let $\mathscr{C}$ be a 2-complex and G a graph, which satisfy the properties of Proposition 3.1. Let $\mathscr{B}$ be a rooted branch decomposition of G. Assume that G has a proper cellular embedding on $\mathscr{C}$. Then it has a proper cellular embedding Γ on $\mathscr{C}$ that is sparse (with respect to $\mathscr{B}$).*

4.1 Monogons and bigons

A **monogon** of a graph Π embedded on a 2-complex $\mathscr{C}$ is a face of Π that is an open disk whose boundary is a single edge of Π (a loop). Similarly, a **bigon** of Π is a face of Π that is an open disk whose boundary is the concatenation of two edges of Π (possibly the same edge appearing twice). The following general lemma on graphs embedded on surfaces without monogons or bigons will be used.

▶ **Lemma 4.2.** *Let $\mathscr{S}$ be a surface of genus g without boundary. Let Π be a graph embedded on $\mathscr{S}$, not necessarily cellularly. Assume that Π has no monogon or bigon. Then $|E(\Pi)| \leq \max\{0, 3g + 3|V(\Pi)| - 6\}$.*

4.2 Vertex simplifications

The proof of Proposition 4.1 starts with any proper cellular embedding of Γ, and iteratively changes the cyclic ordering of edges around vertices in a specific way. Let (E_1, E_2) be an (ordered) partition of $E(G)$, let v be a vertex of G, and let C be a link component at v (if

the image of v under Γ is a singular point, there may be several such link components). We restrict our attention to the edges of $\hat{\Gamma}$ incident to v and belonging to C, in cyclic order around v. For $i = 1, 2$, an **interval** (at v, relatively to $(\hat{E}_1, \hat{E}_2)$) is a maximal contiguous subsequence of edges in this cyclic ordering that all belong to $\hat{E}_i$; the interval is labelled i. **Simplifying** v (with respect to (E_1, E_2)) means changing the cyclic ordering of the edges of $\hat{\Gamma}$ incident to v in C by one of the two following operations (Figure 4):

1. either exchanging two consecutive intervals in that ordering, in a way that the ordering of the edges in each interval is preserved; this operation is allowed only if v is incident to at least four intervals;
2. or performing the previous operation twice, on two disjoint pairs of consecutive intervals in that ordering; this is allowed only if v is incident to at least six intervals.

We will rely on the following lemma.

▶ **Lemma 4.3.** *Let Γ be a proper cellular embedding of G on $\mathscr{C}$, and let (E_1, E_2) be an (ordered) partition of $E(G)$. Let Γ' be another proper cellular embedding of G, obtained from Γ by simplifying one or two vertices with respect to (E_1, E_2), while keeping the other cyclic orderings unchanged. Then:*

1. *$|E(\Pi(\Gamma', E_1, E_2))| < |E(\Pi(\Gamma, E_1, E_2))|$;*
2. *for each (ordered) partition $(\tilde{E}_1, \tilde{E}_2)$ of $E(G)$ such that $\hat{\tilde{E}}_i \subseteq \hat{E}_j$ for some $i, j \in \{1, 2\}$, we have $|E(\Pi(\Gamma', \tilde{E}_1, \tilde{E}_2))| \leq |E(\Pi(\Gamma, \tilde{E}_1, \tilde{E}_2))|$.*

Proof. The proof is based on the following easy but key observations (the second one will be reused later):

- A simplification of v strictly decreases the number of intervals at v;
- the number of half-edges of $\Pi(\Gamma, E_1, E_2)$ at v in the link component C equals twice the number of intervals associated to $(\hat{E}_1, \hat{E}_2)$ at v in C.

The first point of the lemma immediately follows. For the second point, let us consider, in the cyclic ordering around v in C, a maximal contiguous sequence of edges in $\hat{\tilde{E}}_i$. Since $\hat{\tilde{E}}_i \subseteq \hat{E}_j$, when simplifying with respect to (E_1, E_2), this sequence is still contiguous in the new embedding Γ'. It follows that the number of intervals associated to $(\hat{\tilde{E}}_1, \hat{\tilde{E}}_2)$ does not increase when replacing Γ with Γ'. ◀

4.3 Rearranging Γ with respect to an edge partition

We can now prove the following lemma:

▶ **Lemma 4.4.** *Let Γ be a proper cellular embedding of G on $\mathscr{C}$, and let (E_1, E_2) be an (ordered) partition of $E(G)$. There exists a proper cellular embedding Γ' of G such that:*

- *$|E(\Pi(\Gamma', E_1, E_2))| \leq 74c + 26w$, where w is the size of the middle set of (E_1, E_2);*
- *for each (ordered) partition $(\tilde{E}_1, \tilde{E}_2)$ of $E(G)$ such that $\hat{\tilde{E}}_i \subseteq \hat{E}_j$ for some $i, j \in \{1, 2\}$, we have $|E(\Pi(\Gamma', \tilde{E}_1, \tilde{E}_2))| \leq |E(\Pi(\Gamma, \tilde{E}_1, \tilde{E}_2))|$.*

Sketch of proof. Let $\Pi := \Pi(\Gamma, E_1, E_2)$. We assume that Π has "many monogons or bigons" and show that there is another cellular embedding Γ' of G such that:

- $|E(\Pi(\Gamma', E_1, E_2))| < |E(\Pi(\Gamma, E_1, E_2))|$;
- for each (ordered) partition $(\tilde{E}_1, \tilde{E}_2)$ of $E(G)$ such that $\hat{\tilde{E}}_i \subseteq \hat{E}_j$ for some $i, j \in \{1, 2\}$, we have $|E(\Pi(\Gamma', \tilde{E}_1, \tilde{E}_2))| \leq |E(\Pi(\Gamma, \tilde{E}_1, \tilde{E}_2))|$.

Intuitively, if Π has many monogons attached to a single vertex, or many bigons glued together, we can change the embedding Γ in a way that leads to vertex simplifications. By repeatedly iterating this argument, and up to replacing Γ with Γ', this implies that we can assume without loss of generality that Π has "not too many monogons or bigons". Lemma 4.2 then implies that Π has at most $74c + 26w$ edges, which concludes. ◀

4.4 Proof of Proposition 4.1

Proof of Proposition 4.1. Let $\mathscr{B}$ be a rooted branch decomposition of G, and let Γ be a proper cellular embedding of G on $\mathscr{C}$. We consider each arc α of the rooted branch decomposition in turn, in an arbitrary order. For each such arc, we modify Γ by applying Lemma 4.4. We first claim that after these iterations, for each arc α of $\mathscr{B}$, we have $|E(\Pi(\Gamma, \alpha))| \leq 74c + 26w$.

Immediately after applying the above procedure to an arc $\tilde{\alpha}$ of $\mathscr{B}$, corresponding to the (ordered) partition $(\tilde{E}_1, \tilde{E}_2)$ of $E(G)$, we have $|E(\Pi(\Gamma, \tilde{E}_1, \tilde{E}_2))| \leq 74c + 26w$. We now prove that subsequent applications of Lemma 4.4 to other arcs of the rooted branch decomposition do not increase this number of edges. Indeed, let α be another arc, corresponding to the (ordered) partition (E_1, E_2) of $E(G)$, to which we apply Lemma 4.4. The arc α partitions the nodes of the tree $\mathscr{B}$ into two sets N_1 and N_2, and similarly $\tilde{\alpha}$ partitions the nodes of the tree $\mathscr{B}$ into two sets $\tilde{N}_1$ and $\tilde{N}_2$. Because $\mathscr{B}$ is a tree, we have $\tilde{N}_i \subseteq N_j$ for some $i, j \in \{1, 2\}$. This implies that $\hat{\tilde{E}}_i \subseteq \hat{E}_j$ for some $i, j \in \{1, 2\}$; so the second item of Lemma 4.4 implies that the number of edges of $\Pi(\Gamma, \tilde{E}_1, \tilde{E}_2)$ does not increase when processing arc α. This proves the claim.

Finally, there remains to prove that, for each internal node ν of $\mathscr{B}$, the graph $\Pi(\Gamma, \nu)$ has at most $3(74c + 26w)$ edges. This follows relatively easily from the above claim. ◀

5 Dynamic programming algorithm

The result of this section is the following proposition.

▶ **Proposition 5.1.** *Let $\mathscr{C}$ be a 2-complex and G a graph, which satisfy the properties of Proposition 3.1. Let c be the size of $\mathscr{C}$ and n the number of vertices and edges of G. Let $\mathscr{B}$ be a rooted branch decomposition of G of width w. In $(c + w)^{O(c+w)}n$ time, one can report one of the following statements, which is true:*

- *G has no sparse proper cellular embedding into $\mathscr{C}$;*
- *G has an embedding into $\mathscr{C}$.*

(Proposition 4.1 implies that we can remove the adjective "sparse" in the above proposition.)

5.1 Bounding graphs

Let $\mathscr{B}$ be a rooted branch decomposition of G of width w. Recall (see Section 2.1) that the root ρ of $\mathscr{B}$ is a leaf associated to no edge of G. Our algorithm will use dynamic programming in the rooted branch decomposition. For each arc α of $\mathscr{B}$, let G_α be the subgraph of G induced by the edges of G corresponding to the leaves of the subtree of $\mathscr{B}$ rooted at α. The general idea is that we compute all possible relevant embeddings of G_α in subregions of $\mathscr{C}$. Such subregions will be delimited by a graph embedded on $\mathscr{C}$ of small complexity. For the dynamic program to work, we also need to keep track of the location of the vertices in the middle set of α. More precisely, a **bounding graph** for G_α is a proper labelled graph embedding Π on $\mathscr{C}$ (but possibly non-cellular), such that:

- some vertices of Π are labelled; these labels are exactly the vertices of the middle set associated with α, and each label appears exactly once;
- each unlabelled vertex of Π is mapped to a singular point of $\mathscr{C}$;
- each face of Π is labelled 0, 1, or 2;

- G_α has an embedding Γ_α that **respects** Π: each vertex of Π labelled v is mapped, under Π, to the image of v in Γ_α; moreover, the relative interior of each edge of Γ_α lies in the interior of a face of Π labelled 2.

A bounding graph for G_α is **sparse** if it has at most $74c + 26w$ edges. Remark that, if Γ is a sparse proper cellular embedding of G on $\mathscr{C}$ (as defined in Section 4), then $\Pi(\Gamma, \alpha)$ is a sparse bounding graph for the restriction of Γ to G_α.

Henceforth, we regard two (labelled) properly embedded graphs as equal if and only if their (labelled) combinatorial maps are isomorphic. A list $\mathscr{L}_\alpha$ of sparse bounding graphs for G_α is **exhaustive** if the following condition holds: If G has a sparse proper cellular embedding on $\mathscr{C}$, then for each such embedding Γ, the (combinatorial map of the) graph $\Pi(\Gamma, \alpha)$ is in $\mathscr{L}_\alpha$.

The induction step for the dynamic programming algorithm is the following.

▶ **Proposition 5.2.** *Let ν be a non-root node of $\mathscr{B}$ and α be the arc of $\mathscr{B}$ incident to ν that is the closest to the root ρ. Assume that, for each arc $\beta \neq \alpha$ of $\mathscr{B}$ incident to ν, we are given an exhaustive list of sparse bounding graphs for G_β. Then we can, in $(c+w)^{O(c+w)}$ time, compute an exhaustive list of $(c+w)^{O(c+w)}$ sparse bounding graphs for G_α.*

Assuming Proposition 5.2, the proof of which is deferred to the next subsection, it is easy to prove Proposition 5.1:

Proof of Proposition 5.1, assuming Proposition 5.2. We apply the algorithm of Proposition 5.2 in a bottom-up manner in the rooted branch decomposition $\mathscr{B}$. Let α be the arc of $\mathscr{B}$ incident with the root node ρ. We end up with an exhaustive list of sparse bounding graphs for $G_\alpha = G$. By definition of a bounding graph, if this list is non-empty, then G has an embedding on $\mathscr{C}$. On the other hand, by definition of an exhaustive list, if this list is empty, then G has no sparse proper cellular embedding on $\mathscr{C}$.

There are $O(n)$ recursive calls, each of which takes $(c+w)^{O(c+w)}$ time. ◀

5.2 The induction step: Proof of Proposition 5.2

Proof of Proposition 5.2. *First case.* Let us first assume that ν is a (non-root) leaf of $\mathscr{B}$; thus, G_α is a single edge uv. Using Lemma 3.3, we compute *all* the labelled combinatorial maps of sparse bounding graphs for G_α. This is clearly an exhaustive list. Indeed, assume that G has a sparse proper cellular embedding Γ on $\mathscr{C}$; by sparsity, $\Pi(\Gamma, \alpha)$ has at most $74c + 26w$ edges; thus, one of the labelled combinatorial maps computed will be equal to that of $\Pi(\Gamma, \alpha)$.

Second case. Let us now assume that ν is an internal node of $\mathscr{B}$. As above, let α be the arc of $\mathscr{B}$ incident to ν that is the closest to the root ρ. Let β and γ be the arcs different from α incident to ν. Let $\mathscr{L}_\beta$ and $\mathscr{L}_\gamma$ be exhaustive lists of bounding graphs for G_β and G_γ, respectively. Intuitively, every pair of bounding graphs in $\mathscr{L}_\beta$ and $\mathscr{L}_\gamma$ that are compatible, in the sense that the regions labelled 2 in each of these two graphs are disjoint, will lead to a bounding graph in $\mathscr{L}_\alpha$. This is the motivating idea to our approach. More precisely, we will enumerate labelled combinatorial maps Π, each of which can be "restricted" to two compatible graphs, which are possible bounding graphs for G_β and G_γ. If these two restrictions lie in $\mathscr{L}_\alpha$ and $\mathscr{L}_\beta$, this leads to a graph that is added to $\mathscr{L}_\alpha$.

We first introduce some terminology. Let Π be a graph properly embedded on $\mathscr{C}$ (possibly non-cellularly), with faces labelled 0, 1, 2, or 3, and with labels on some vertices. Let i, j, k be integers such that $\{i, j, k\} = \{1, 2, 3\}$. We will define a graph embedding $\Pi_{i,j}$ obtained

from Π by somehow "merging" faces i and j. First, for an illustration, refer back to Figures 2 and 3: If Π is the graph embedding depicted on the right of Figure 3, then the configurations shown on the right of Figure 2 correspond, from top to bottom, to $\Pi_{2,3}$, $\Pi_{1,3}$, and $(\Pi_{1,2})^-$ (the latter being the graph $\Pi_{1,2}$ in which each face label 3 is replaced by a 2).

Formally, $\Pi_{i,j}$ is defined as follows. First, let us replace all face labels j by i. Now, for each face f of Π that is homeomorphic to a disk and labelled 0, we do the following. The boundary of f is made of edges of Π; for the sake of the discussion, let us temporarily label each such edge by the label of the face on the other side of f. If all edges on the boundary of f all labelled i, then we remove all these edges, and f becomes part of a larger face labelled i. Otherwise, for each maximal subsequence $e_1, \ldots, e_\ell$ of edges along the boundary of f that are all labelled i, we remove each of $e_1, \ldots, e_\ell$, and replace them with an edge inside f from the source of e_1 to the target of e_ℓ. Finally, we remove all isolated vertices that do not coincide with a singular point of $\mathscr{C}$, and all vertices in the relative interior of an isolated segment that are incident to two faces with the same label.

For any labelled combinatorial map Π, we denote by Π^- the same map where each label 3 on a face is replaced by a 2. The easy but key properties of this construction are the following:

(i) Assume that $\Pi_{1,3}$ is a bounding graph for G_β and $(\Pi_{1,2})^-$ is a bounding graph for G_γ. Then $\Pi_{2,3}$ is a bounding graph for G_α.

(ii) The node ν naturally partitions the edge set of G into three parts, which we denote by E_1 (on the side of α), E_2 (on the side of β), and E_3 (on the side of γ). Assume that G has a sparse proper cellular embedding Γ on $\mathscr{C}$ and that $\Pi = \Pi(\Gamma, E_1, E_2, E_3)$. Then:

- $\Pi(\Gamma, \alpha) = \Pi(\Gamma, E_1, E_2 \cup E_3) = \Pi_{2,3}$;
- $\Pi(\Gamma, \beta) = \Pi(\Gamma, E_1 \cup E_3, E_2) = \Pi_{1,3}$;
- $\Pi(\Gamma, \gamma) = \Pi(\Gamma, E_1 \cup E_2, E_3) = (\Pi_{1,2})^-$.

Property (ii) is, again, illustrated by Figures 2 and 3: If (E_1, E_2, E_3) is the edge partition depicted on Figure 3, then the edge partitions depicted on Figure 2, left, are, respectively, $(E_1, E_2 \cup E_3)$, $(E_1 \cup E_3, E_2)$, and $(E_1 \cup E_2, E_3)$. As shown above, the corresponding partitioning graphs are respectively $\Pi_{2,3}$, $\Pi_{1,3}$, and $\Pi^-_{1,2}$.

If Π is the graph embedding depicted on the right of Figure 3, then the configurations shown on the right of Figure 2 correspond, from top to bottom, to $\Pi_{2,3}$, $\Pi_{1,3}$, and $(\Pi_{1,2})^-$ (the latter being the graph $\Pi_{1,2}$ in which each face label 2 is replaced by a 2).

We compute our exhaustive list $\mathscr{L}_\alpha$ of sparse bounding graphs for G_α as follows. Initially, let this list be empty. Using Lemma 3.3, we enumerate all combinatorial maps Π of graphs with at most $c + 3w$ vertices and $3(74c + 26w)$ edges properly embedded on $\mathscr{C}$ (possibly non-cellularly), with faces labelled 0, 1, 2, or 3, and such that the labels appearing on the vertices are exactly the vertices of the middle set of α, β, or γ (and each label appears exactly once). This takes $(c + w)^{O(c+w)}$ time. Whenever $\Pi_{1,3} \in \mathscr{L}_\beta$ and $(\Pi_{1,2})^- \in \mathscr{L}_\gamma$, we add $\Pi_{2,3}$ to $\mathscr{L}_\alpha$. Finally, we eliminate duplicates by testing pairwise isomorphism between the labelled combinatorial maps in $\mathscr{L}_\alpha$, and remove the graphs that are not sparse or contain vertices that bear a label not in the middle set of α.

$\mathscr{L}_\alpha$ contains only sparse bounding graphs for G_α, by (i) above. Moreover, let Γ be a sparse proper cellular graph embedding of G on $\mathscr{C}$. By sparsity, one of the graphs Π enumerated in the previous paragraph is $\Pi(\Gamma, \nu)$. By definition of $\mathscr{L}_\beta$ and $\mathscr{L}_\gamma$, we have that $\Pi(\Gamma, \beta) \in \mathscr{L}_\beta$ and $\Pi(\Gamma, \gamma) \in \mathscr{L}_\gamma$, so by (ii) above, $\Pi(\Gamma, \alpha) \in \mathscr{L}_\alpha$, which implies that $\mathscr{L}_\alpha$ is exhaustive. ◀

6 Reduction to proper cellular embeddings

▶ **Proposition 6.1.** *Let $\mathscr{C}$ be a 2-complex with at most c simplices, and G a graph with at most n vertices and edges and branchwidth at most w. Assume that G and $\mathscr{C}$ satisfy the properties of Proposition 3.1. In $c^{O(c)} + O(cn)$ time, one can compute a graph G', and $c^{O(c)}$ 2-complexes $\mathscr{C}_i$, such that:*

1. *each $\mathscr{C}_i$ and G' satisfy the properties of Proposition 3.1;*
2. *G' has at most $5cn$ vertices and $5cn$ edges, and branchwidth at most w;*
3. *each $\mathscr{C}_i$ has size at most c;*
4. *if, for some i, G' embeds into $\mathscr{C}_i$, then G embeds into $\mathscr{C}$;*
5. *if G embeds into $\mathscr{C}$, then for some i, G' has a proper cellular embedding into $\mathscr{C}_i$.*

Sketch of proof. Roughly but not exactly, G' is obtained from G by dissolving every degree-two vertex of G and then subdividing edges $\Theta(c)$ times, and the complexes $\mathscr{C}_i$ are all the 2-complexes "smaller" than $\mathscr{C}$, obtained by detaching the singular points of $\mathscr{C}$ in all possible ways, removing parts of the isolated segments of $\mathscr{C}$ in all possible ways, and simplifying the topology of the detached surface in all possible ways. It is clear that, if G' embeds into one of these complexes, then G embeds into $\mathscr{C}$. Conversely, if G embeds into $\mathscr{C}$, then, by the subdivision process above, G' has a proper embedding into $\mathscr{C}$, and the only problem is that the faces of G' may fail to be disks. However, by modifying $\mathscr{C}$ using the appropriate choice of operations above, these faces are transformed into disks. ◀

7 Algorithm for bounded branchwidth: Proof of Theorem 1.2

Proof of Theorem 1.2. The proof of this theorem follows directly from the previous propositions. After the preprocessing step (Proposition 3.1), combining Propositions 4.1 and 5.1 give an algorithm that takes as input G and $\mathscr{C}$ and reports one of the following true statements: (i) G has no proper cellular embedding on $\mathscr{C}$; (ii) G has an embedding on $\mathscr{C}$. So, we apply this algorithm to the graph G' and each 2-complex $\mathscr{C}_i$ obtained from Proposition 6.1. If for at least one of these instances, the outcome is (ii), then G has an embedding into $\mathscr{C}$. Otherwise, G has no embedding into $\mathscr{C}$. ◀

8 Reduction to bounded branchwidth: Proof of Theorem 1.3

Sketch of proof of Theorem 1.3. This is based on an irrelevant method. We assume that G has large branchwidth. An important fact that we will use is that, if G is embeddable on $\mathscr{C}$, it has genus $O(c)$.

A **wall** of size $k \times k$ is a subgraph of the $(k \times k)$-grid obtained by removing alternately the vertical edges of even (resp. odd) x-coordinate in each even (resp. odd) line, and then the degree-one vertices.

We first use any algorithm to approximate the treewidth of G, e.g., Fomin et al. [11, Theorem 1.1]: In polynomial time, we either compute a branch decomposition of small width of G, in which case we are done, or correctly report that the treewidth is large. In the latter case, the result by Chekuri and Chuzhoy [3] implies that there is a large grid minor, which we can compute in randomized polynomial time using the same article, or in deterministic $f(k) \cdot n^2$ time, where f is some computable function and k is the size of the grid [22, Algorithm 4.4]. We have thus computed a subdivision of a large wall.

We then partition this wall into $\Omega(c)$ smaller subwalls $W_1, \dots, W_m$, where $m = \Omega(c)$, each bounded by a cycle γ_i. We then show that we can assume that there are (intuitively) not too many connections, in G, between two different walls W_i, W_j that avoid the cycles γ_i

and γ_j; otherwise, by computing these connections, we exhibit a subgraph of G of genus $\Omega(c)$, and deduce that G is not embeddable on $\mathscr{C}$. Finally, we compute a cycle γ of G, such that one connected component of $G - \gamma$ is planar and contains a subdivision of a smaller, but still large, wall.

We then show, borrowing some ingredients to Kociumaka and Pilipczuk [15, Section 5], that the central vertex of this wall is irrelevant, in the sense that its removal does not affect the embeddability or non-embeddability of the graph into $\mathscr{C}$. Intuitively, v is surrounded by $\Omega(c)$ concentric cycles in the wall; if $G - v$ is embedded in $\mathscr{C}$, then two concentric cycles must bound an annulus. The planar part inside the inner cycle can be embedded close to the boundary of the annulus that corresponds to this inner cycle.

Iterating the whole procedure, we obtain a graph of branchwidth polynomial in the size of $\mathscr{C}$. ◀

References

1 Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 641–650, 2008.

2 Christoph Buchheim, Markus Chimani, Carsten Gutwenger, Michael Jünger, and Petra Mutzel. Crossings and planarization. In Roberto Tamassia, editor, *Handbook of graph drawing and visualization*. Chapman and Hall, 2006.

3 Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *Journal of the ACM*, 63(5):Article 40, 2016.

4 Éric Colin de Verdière. Computational topology of graphs on surfaces. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 23. CRC Press LLC, third edition, 2018.

5 Éric Colin de Verdière and Arnaud de Mesmay. Testing graph isotopy on surfaces. *Discrete & Computational Geometry*, 51(1):171–206, 2014.

6 Éric Colin de Verdière, Thomas Magnard, and Bojan Mohar. Embedding graphs into two-dimensional simplicial complexes. In *Proceedings of the 34th International Symposium on Computational Geometry (SOCG)*, pages 27:1–27:14, 2018.

7 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer-Verlag, 2015.

8 Tamal K. Dey and Sumanta Guha. Transforming curves on surfaces. *Journal of Computer and System Sciences*, 58:297–325, 1999.

9 David Eppstein. Dynamic generators of topologically embedded graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 599–608, 2003.

10 Jeff Erickson and Kim Whittlesey. Transforming curves on surfaces redux. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1646–1655, 2013.

11 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, Michał Pilipczuk, and Marcin Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms*, 14(3):Article 34, 2018.

12 John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM*, 21(4):549–568, 1974.

13 Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 771–780, 2008.

14 Ken-ichi Kawarabayashi and Bruce Reed. Computing crossing number in linear time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 382–390, 2007.

15 Tomasz Kociumaka and Marcin Pilipczuk. Deleting vertices to graphs of bounded genus. *Algorithmica*, 81:3655–3691, 2019.

16 Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 440–449, 2012.

17 Seth M. Malitz. Genus g graphs have pagenumber $O(\sqrt{g})$. *Journal of Algorithms*, 17:85–109, 1994.

18 Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999.

19 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001.

20 Colm Ó Dúnlaing, Colum Watt, and David Wilkins. Homeomorphism of 2-complexes is equivalent to graph isomorphism. *International Journal of Computational Geometry & Applications*, 10:453–476, 2000.

21 Maurizio Patrignani. Planarity testing and embedding. In Roberto Tamassia, editor, *Handbook of graph drawing and visualization*. Chapman and Hall, 2006.

22 Neil Robertson and Paul D Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995.

23 Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92:325–357, 2004.

24 Marcus Schaefer. Toward a theory of planarity: Hanani–Tutte and planarity variants. *Journal of Graph Algorithms and Applications*, 17(4):367–440, 2013.

25 John Stillwell. *Classical topology and combinatorial group theory*. Springer-Verlag, New York, second edition, 1993.

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