# ETH-Tight Algorithms for Finding Surfaces in Simplicial Complexes of Bounded Treewidth

**Mitchell Black**
School of Electrical Engineering and Computer Science,
Oregon State University, Corvallis, OR, USA

**Nello Blaser**
Department of Informatics, University of Bergen, Norway

**Amir Nayyeri**
School of Electrical Engineering and Computer Science,
Oregon State University, Corvallis, OR, USA

**Erlend Raa Vågset**
Department of Informatics, University of Bergen, Norway

## Abstract

Given a simplicial complex with $n$ simplices, we consider the Connected Subsurface Recognition (c-SR) problem of finding a subcomplex that is homeomorphic to a given connected surface with a fixed boundary. We also study the related Sum-of-Genus Subsurface Recognition (SoG) problem, where we instead search for a surface whose boundary, number of connected components, and total genus are given. For both of these problems, we give parameterized algorithms with respect to the treewidth $k$ of the Hasse diagram that run in $2^{O(k \log k)} n^{O(1)}$ time. For the SoG problem, we also prove that our algorithm is optimal assuming the exponential-time hypothesis. In fact, we prove the stronger result that our algorithm is ETH-tight even without restriction on the total genus.

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Mathematics of computing → Algebraic topology; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** Computational Geometry, Surface Recognition, Treewidth, Hasse Diagram, Simplicial Complexes, Low-Dimensional Topology, Parameterized Complexity, Computational Complexity

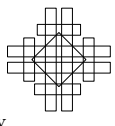
**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2022.17

**Related Version** *Full Version*: `https://arxiv.org/abs/2203.07566` [5]

**Funding** *Mitchell Black*: This author was supported in part by NSF grants CCF-1941086 and CCF-1816442.
*Amir Nayyeri*: This author was supported in part by NSF grants CCF-1941086 and CCF-1816442.
*Erlend Raa Vågset*: This author was supported in part by the Research Council of Norway grant "Parameterized Complexity for Practical Computing (PCPC)" (NFR, no. 274526).

## 1 Introduction

Simplicial complexes are a generalization of graphs that give a discrete representation of higher-dimensional spaces. A natural and interesting class of such spaces are manifolds. A $d$-manifold is a space that is "locally $d$-dimensional", meaning each point has a neighborhood homeomorphic to $\mathbb{R}^d$. Circles and spheres are prototypical examples of 1- and 2-manifolds respectively. Manifolds are important in both mathematics and computer science. For example, triangular meshes in computer graphics are typically 2-manifolds, and the manifold hypothesis in machine learning is the assumption that real-world data often lie on low-dimensional submanifolds of high-dimensional spaces.


Since manifolds are so important, it is natural to ask if a given simplicial complex is a manifold, or whether two manifolds are homeomorphic. There are fascinating complexity results on these problems. While both recognizing and classifying a 2-manifold have polynomial algorithms, this problem becomes much harder for arbitrary $d$-manifolds. Deciding whether two manifolds are homeomorphic is undecidable for $d \geq 4$ [20]. Deciding whether or not a simplicial complex is homeomorphic to the $d$-sphere is undecidable for $d \geq 5$ (see [13]), which implies deciding whether or not a simplicial complex is an $n$-manifold is undecidable for $d \geq 6$.

We consider several variants of the problem of finding subcomplexes homeomorphic to 2-manifolds, or *surfaces*, in simplicial complexes. While there are polynomial time algorithms for deciding if a simplicial complex is homeomorphic to a surface or deciding the homeomorphism class of a surface, it is a hard problem deciding whether or not a simplicial complex contains a surface as a subcomplex. In particular, Ivanov proved that it is NP-Hard to decide if a simplicial complex contains a 2-sphere [19], and Burton et al. proved that finding a 2-sphere is W[1]-hard when parameterized by solution size [9]. The complexity of this problem is analogous to the graph isomorphism problem. While there is a quasipolynomial algorithm to determine if two graphs are isomorphic [3], it is NP-Hard to determine if one graph contains a subgraph isomorphic to another graph [14].

As this problem is NP-Hard, it is natural to ask whether there are any class of simplicial complexes for which polynomial time algorithms exist. In this paper, we consider the parameterized complexity of this problem and related problems with respect to the treewidth of the Hasse diagram. A tree decomposition of the Hasse diagram defines a recursive series of subcomplexes of $K$ that we can use to incrementally build our surfaces. We also give tight lower bounds for a subset of our algorithms based on the Exponential Time Hypothesis.

## 1.1 Subsurface recognition problems

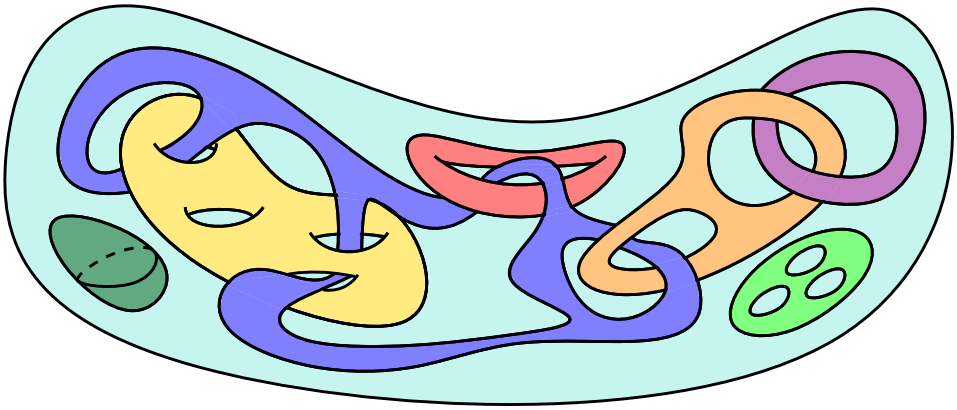


**Figure 1** A solution to an instance of the Subsurface Recognition problem where we have found an orientable surface consisting of seven connected components with genus **0**, **1**, **1**, **2**, **3**, **3** and 4 respectively.

We consider several variants of the following generic problem: given a 2-dimensional simplicial complex $K$ and a 1-dimensional subcomplex $B \subset K$, does $K$ contain a subcomplex homeomorphic to a surface with boundary $B$? Note that this includes finding surfaces without boundary, as we can set $B = \emptyset$.

The Subsurface Recognition (SR) problem places the most restrictions on the manifold we are looking for. In this problem, we are asked to find a subcomplex of $K$ homeomorphic to a given (possibly disconnected) surface $X$. Figure 1 shows an example of SR.

▶ **Problem 1.** *The Subsurface Recognition (SR) problem:*
*Input: A simplicial complex $K$, a subcomplex $B \subset K$, and a surface $X$.*
*Question: Does $K$ contain a subcomplex homeomorphic to $X$ with boundary $B$?*

Although there is no known FPT algorithm for SR, several variants of SR with looser requirements admit FPT algorithms. One special case of SR requires the surface $X$ to be connected. This variant is called the Connected Subsurface Recognition (c-SR) problem. The extra requirement of connectivity allows us to find an FPT algorithm.

▶ **Problem 2.** *The Connected Subsurface Recognition (c-SR) problem:*
*Input: A simplicial complex $K$, a subcomplex $B \subset K$, and a connected surface $X$.*
*Question: Does $K$ contain a subcomplex homeomorphic to $X$ with boundary $B$?*

We can also ask for a surface of a certain genus and orientability in $K$, which is a slightly weaker criterion than finding a surface up to homeomorphism. For a disconnected surface, we define its **total genus** to be the sum of the genus of each of its connected components[1]. While a connected surface is characterized up to homeomorphism by its genus and orientability, this is not true for disconnected surfaces. As an example, consider a surface $X$ that is a genus 2 surface and a surface $Y$ that is the disjoint union of two tori. The two surfaces both have total genus 2, but they are not homeomorphic.

▶ **Problem 3.** *The Sum-of-Genus Subsurface Recognition (SoG) problem:*
*Input: A simplicial complex $K$, a subcomplex $B \subset K$, and integers $g$ and $c$.*
*Question: Does $K$ contain a surface $X$ of total genus $g$ with $c$ connected components and with boundary $B$?*

The Subsurface Packing problem asks to find *any* set of $c$ disjoint surfaces. In particular, no restriction is placed on the genus or orientability of these surfaces.

▶ **Problem 4.** *The Subsurface Packing (SP) problem:*
*Input: A simplicial complex $K$, a subcomplex $B$, and an integer $c$.*
*Question: Does $K$ contain a surface $X$ with $c$ connected components and boundary $B$?*

## 1.2 Our results

**Table 1** Upper and ETH lower bounds for times to solve the different problems considered in this manuscript. Here $n$ is the number of simplices and $k$ is the treewidth of the Hasse diagram. The results of this paper are highlighted.

| Problem | SR | c-SR | SoG | SP |
|---|---|---|---|---|
| Upper | $2^{O(n)}$ | $\mathbf{2^{O(k \log k)} n^{O(1)}}$ | $\mathbf{2^{O(k \log k)} n^{O(1)}}$ | $\mathbf{2^{O(k \log k)} n^{O(1)}}$ |
| Lower | $\mathbf{2^{o(k \log k)} n^{O(1)}}$ | NP-Hard [19] | $\mathbf{2^{o(k \log k)} n^{O(1)}}$ | $\mathbf{2^{o(k \log k)} n^{O(1)}}$ |

We consider the parameterized complexity of the above problems with respect to the treewidth $k$ of the Hasse diagram. Table 1 summarizes the known upper and lower bounds. The results of this paper are highlighted. We give FPT algorithms for c-SR, SoG, and SP, and ETH-based lower bounds for SR, SP, and SoG. In fact, we show that these lower bounds are true even when $k$ is the pathwidth of the Hasse diagram. The algorithms for SoG and SP are ETH-tight.

[1] If any connected component of a surface is non-orientable, we will add twice the genus of any orientable components.

## 1.3 Related work

### Tree decompositions and simplicial complexes

Tree decompositions have seen much success as an algorithmic tool on graphs. Often, graphs having tree decompositions of bounded-width admit polynomial-time solutions to otherwise hard problems. A highlight of the algorithmic application of tree decompositions is Courcelle's Theorem [15], which states that any problem that can be stated in monadic second order logic can be solved in linear time on graphs with bounded treewidth. We recommend [16, Chapter 7] for an introduction to the algorithmic use of tree decompositions.

While tree decompositions have long been successful for algorithms on graphs, they have only recently seen attention for algorithms on simplicial complexes. Existing algorithms use tree decompositions of a variety of graphs associated with a simplicial complex. The most commonly used graph is the dual graph of combinatorial $d$-manifolds [4, 10, 11, 12]. Other graphs that have been used are level $d$ of the Hasse diagram [11, 7, 6], the adjacency graph of the $d$-simplices [7], and the 1-skeleton [4]. Our algorithm uses a tree decomposition of the entire Hasse diagram. As far as we know, we are the first to consider tree decompositions of the full Hasse diagram. The condition on vertex links that makes a simplicial complex a surface is dependent on the incidence of vertices and triangles (see Section 2.2), so considering only one level of the Hasse diagram would likely not be sufficient for our problem.

### Normal surface theory

Normal surface theory is the study of which surfaces exist as submanifolds of a given 3-manifold. Many algorithms on 3-manifolds, like those for unknot recognition [18] and 3-sphere recognition [21, 22], use normal surface theory. While normal surface theory appears to be similar to our problems, the distinction is that the surfaces in normal surface theory are not subcomplexes of the 3-manifold and can instead intersect 3-simplices in the manifold. Accordingly, the techniques in normal surface theory are quite different from the algorithms we present in this paper.

# 2 Background

## 2.1 Simplicial complexes and directed graphs

A **simplicial complex** is a set $K$ such that (1) each element $\sigma \in K$ is a finite set and (2) for each $\sigma \in K$, if $\tau \subset \sigma$, then $\tau \in K$. An element $\sigma \in K$ is a **simplex**. A simplex $\sigma$ is a **face** of a simplex $\tau$ if $\sigma \subset \tau$. Likewise, $\tau$ is a **coface** of $\sigma$. The simplices $\sigma$ and $\tau$ are **incident**. Two simplices $\sigma_1$ and $\sigma_2$ are **adjacent** if they are both the face or coface of a simplex $\tau$.

A simplex $\sigma$ with $|\sigma| = d + 1$ is a **d-simplex**. The set of all $d$-simplices in $K$ is denoted $K_d$. The **dimension** of a simplicial complex is the largest integer $d$ such that $K$ contains a $d$-simplex. A $d$-dimensional simplicial complex $K$ is **pure** if each simplex in $K$ is a face of $d$-simplex. We call a 0-simplex a **vertex**, a 1-simplex an **edge**, and a 2-simplex a **triangle**.

The **Hasse diagram** of $K$ is a graph $H$ with vertex set $K$ and edges between each $d$-simplex $\sigma \in K$ and each $(d-1)$-dimensional face of $\sigma$ for all $d > 0$.

Let $\Sigma \subset K$. The **closure** of $\Sigma$ is $\operatorname{cl}\Sigma := \{\tau \subset \sigma \mid \sigma \in \Sigma\}$. Note that the closure of $\Sigma$ is a simplicial complex, even if $\Sigma$ is not. Note also that the closure $\operatorname{cl}\Sigma$ is defined only by the set $\Sigma$ and not the complex $K$. The **star** of $\Sigma$ is $\operatorname{st}_K \Sigma := \{\sigma \in K \mid \exists \tau \in \Sigma \text{ such that } \tau \subset \sigma\}$.

The **link** of a simplex $\sigma$ is $\operatorname{lk}_K \sigma = \operatorname{cl}\operatorname{st}_K \sigma - \operatorname{st}_K \operatorname{cl}\sigma$. Alternatively, the link $\operatorname{lk}_K \sigma$ is all simplices in $\operatorname{cl}\operatorname{st}_K \sigma$ that do not intersect $\sigma$. Note that for any simplex $\tau \in \operatorname{lk}_K \sigma$ that $\sigma$ and $\tau$ are incident to a common coface in $\operatorname{st}_K \sigma$.

A **simple path** is a 1-dimensional simplicial complex $P = \{\{v_1\}, \{v_1, v_2\}, \{v_2\}, \ldots, \{v_l\}\}$ such that the vertices $\{v_i\}$ are distinct. The vertices $\{v_1\}, \{v_l\}$ are the **endpoints** of $P$. We will denote a simple cycle as a tuple $P = (v_1, \ldots, v_l)$ as the edges are implied by the vertices. A **simple cycle** is a simple path, with the exception that the endpoints $v_1 = v_l$. We denote a simple cycle with an overline, e.g. $(\overline{v_1, \ldots, v_l})$.

A directed graph $D$ consists of a set of vertices and a set of directed edges, i.e. ordered pairs of vertices $(u, v) := uv$ so that $uv \neq vu$. A **directed simple cycle** $C$ in $D$ (not to be confused with a simple cycle) is a sequence of directed edges $(v_1v_2, v_2v_3, \ldots, v_lv_1)$ where all the vertices $v_i$ are all distinct. We say that $C$ has the vertex set $\{v_1, \ldots, v_l\}$. Two cycles, $C$ and $C'$, are said to be **vertex disjoint** if their vertex sets are disjoint. A family of cycles is said to be vertex disjoint if they are pairwise vertex disjoint.


**Figure 2** Left: A combinatorial surface. The vertex $v$ is an interior vertex. Right: A vertex $v$ with link that is neither a simple path or cycle. We conclude that $S$ is not a combinatorial surface. The point $v$ has no neighborhood homeomorphic to the plane or half-plane, so $S$ is not "locally 2-dimensional" at $v$.

## 2.2 Surfaces

Informally, a **surface with boundary** is a compact topological space where each point has a neighborhood homeomorphic to the plane or the half plane, and the **boundary** of the surface is all points with a neighborhood homeomorphic to the half plane. Intuitively, a surface is "locally 2-dimensional".

Any connected surface with boundary can be constructed by adding handles, crosscaps, and boundary components to a sphere. A **handle** is constructed by removing two disjoint disks from a surface and identifying the boundaries of the removed disks. A **crosscap** is constructed by taking the disjoint union of the surface and the real projective plane, removing a disk from each, and identifying the boundaries of the removed disks. A **boundary component** is constructed by removing a disk from a surface. A surface is **non-orientable** if it has a crosscap and **orientable** otherwise. The ***genus*** of an orientable surface is the number of handles on the surface, and the genus of a non-orientable surface is the number of crosscaps plus twice the number of handles.

In this paper, we are only concerned with surfaces that are also simplicial complexes, which we call combinatorial surfaces. A **combinatorial surface with boundary** is a pure 2-dimensional simplicial complex $S$ such that the link of each vertex is a simple path or a simple cycle. The condition on the link of the vertices is the combinatorial way of saying that a combinatorial surface is "locally 2-dimensional". A vertex $v \in S$ such that $\operatorname{lk}_S v$ is a simple path is a **boundary vertex**. A vertex $v \in S$ such that $\operatorname{lk}_S v$ is a simple cycle is an **interior vertex**. Figure 2 shows examples of an interior vertex and a vertex that is neither

an interior or boundary vertex. It follows from the condition on the links of the vertices that each edge $e \in S$ has link $\mathrm{lk}_S e$ that is either one or two vertices. An edge $e \in S$ such that $\mathrm{lk}_S e$ is a single vertex is a **boundary edge**. An edge $e \in S$ such that $\mathrm{lk}_S e$ is two vertices is an **interior edge**. A triangle $t \in S$ has empty link $\mathrm{lk}_S t = \emptyset$ as $S$ is a 2-dimensional simplicial complex. We denote the set of boundary vertices and boundary edges $\partial S$. The boundary $\partial S$ is a collection of simple cycles.

## 2.3 Tree decompositions

Let $G = (V, E)$ be a graph. A **tree decomposition** of $G$ is a tuple $(T, X)$, where $T = (I, F)$ is a tree with nodes $I$ and edges $F$, and $X = \{X_t \subset V \mid t \in I\}$ such that (1) $\cup_{t \in I} X_t = V$, (2) for any $\{v_1, v_2\} \in E$, $\{v_1, v_2\} \subset X_t$ for some $t \in I$, and (3) for any $v \in V$, the subtree of $T$ induced by the nodes $\{t \in I \mid v \in X_t\}$ is connected. A set $X_t$ is the **bag** of $T$. The **width** of $(T, X)$ is $\max_{t \in I} |X_t| - 1$. The **treewidth** of a graph $G$ is the minimum width of any tree decomposition of $G$. Computing the treewidth of a graph is NP-hard [2], but there are algorithms to compute tree decompositions that are within a constant factor of the treewidth, e.g. [8].

Tree decompositions are used to perform dynamic programs on graphs, and a certain type of tree decomposition, called a nice tree decomposition, makes defining dynamic programs easier. A **nice tree decomposition** is a tree decomposition with a specified root $r \in I$ such that (1) $X_r = \emptyset$, (2) $X_l = \emptyset$ for all leaves $l \in I$, and (3) all non-leaf nodes are either an introduce node, a forget node, or a join node, which are defined as follows. An **introduce node** is a node $t \in I$ with exactly one child $t'$, and for some $w \in V$, $w \notin X_{t'}$ and $X_t = X_{t'} \cup \{w\}$. We say $t$ **introduces** $w$. A **forget node** is a node $t \in I$ with exactly one child $t'$, and for some $w \in V$, $w \notin X_t$ and $X_t \cup \{w\} = X_{t'}$. We say $t$ **forgets** $w$. A **join node** is a node $t \in I$ with exactly two children $t'$ and $t''$ where $X_t = X_{t'} = X_{t''}$. The following lemma proves that we can convert any tree decomposition to a nice tree decomposition without increasing width.

▶ **Lemma 1** (Lemma 7.4 of [16]). *Given a tree decomposition $(T = (I, F), X)$ of width $k$ of a graph $G = (V, E)$, a nice tree decomposition of width $k$ with $O(kn)$ nodes can be computed in $O(k^2 \max\{|V|, |I|\})$ time.*

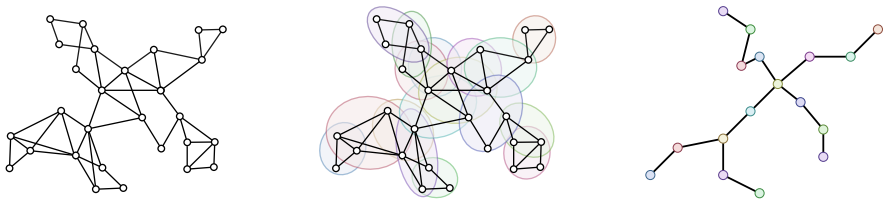


**Figure 3** Left: A graph. Right and Center: A (not nice) tree decomposition of the graph of width 3. Each node of the tree corresponds to a subset of the vertices of the graph.

A **path decomposition** is a special kind of tree decomposition $(T, X)$ where $T$ is a path. A **nice path decomposition** is a tree decomposition without join nodes, i.e. where every node is either an introduce node or a forget node. The **pathwidth** of a graph $G$ is the smallest width of any path decomposition of $G$. As any path decomposition is also a tree decomposition, the treewidth of $G$ is at most the pathwidth of $G$.

## 2.4 The exponential time hypothesis

When a new algorithm is discovered it is natural to ask if it is possible to improve it. To prove that the algorithm was sub-optimal it is enough to find a new and better algorithm. On the other hand, if the algorithm is actually the best possible, then the situation becomes more complicated. Although there are optimality results for a few problems in P,[2] none are known for algorithms solving NP-complete problems. Such a result would imply P $\neq$ NP, which remains famously unproven.

This theoretical barrier does not make the question of optimality less relevant. No one wants to spend years searching for improvements to an algorithm that cannot be improved! For instance, the algorithms in this paper need $2^{O(k \log k)} n^{O(1)}$ time, which may prompt the question "Why were you unable to deliver a $2^{O(k)}n$ time solution?".

A pragmatic and popular response to these kinds of questions is to prove that you have optimality under the Exponential Time Hypothesis (ETH). The ETH is a conjecture stating that there is no sub-exponential algorithm for 3-SAT. More precisely, let $n$ be the number of variables in a given instance of 3-SAT.

▶ **Hypothesis 1** (ETH). *3-SAT cannot be solved in time* $2^{o(n)}$.

Similar to NP-hardness, an ETH-lower bound is a way of connecting the hardness of a new and often poorly understood problem to problems we already have a good understanding of. The idea is to show that an improvement on the runtime of the currently best algorithm for a new problem would disprove the ETH. Although the ETH remains unproven, the continued absence of any algorithm for 3-SAT fast enough to disprove the ETH is itself strong empirical evidence in support of the hypothesis.

# 3 Overview of the algorithms

Our algorithms are all dynamic programs on a tree decomposition $(T, X)$ of the Hasse diagram of a simplicial complex $K$. For each node $t \in T$, starting at the leaves of $T$ and moving towards the root, we compute a set of candidate solutions to our problem, where a candidate solution is a subcomplex of $K$ that might be a subcomplex of a solution to our problem. We recursively use candidate solutions at the children of $t$ to build the candidate solutions at $t$. At the end of the algorithm, candidate solutions at the root of $t$ will be solutions to our problem. In this section, we explore how a candidate solution to our problem is defined, and how we can effectively store representations of these candidate solutions so that our final algorithm is FPT.

Certain nice tree decompositions[3] $(T, X)$ of the Hasse diagram of a simplicial complex $K$ define a recursively-nested set of subcomplexes of $K$. Recall that each bag of the tree decomposition is a set of simplices of $K$. For each node $t \in T$, the subcomplex $K_t \subset K$ is the union of the bags of each descendant of $t$ minus the triangles in the bag of $t$. These subcomplexes have the property that if $t'$ is a child of $t$, then $K_{t'} \subset K_t$.

We use this set of subcomplexes to recursively build solutions to our problems. Our algorithm computes a set of **candidate solutions** at each node $t$. The exact definition of candidate solution is given in Section 3.5 of the extended version of this paper [5], but

[2] One such example is sorting, which we know can at best be done in $\Omega(n \log n)$ time.

[3] Certain here means "closed", which is a type of tree decomposition of the Hasse diagram we define in Section 3.3 of the the extended version of this paper [5]. In particular, the set $K_t$ as defined above is a simplicial complex in a closed tree decomposition, which is not true for general tree decompositions.

intuitively, a candidate solution at a node $t$ is a subcomplex of $K_t$ that could be a subcomplex of a combinatorial surface in $K$. In particular, the link of each vertex in a candidate solution must be a subset of a simple path or simple cycle. Our definition of candidate solution works recursively: if $\Sigma$ is a candidate solution at $t$, then for each child $t'$ of $t$, the complex $\Sigma \cap K_{t'}$ is a candidate solution at $t'$. Our algorithm uses this fact to find candidate solutions at $t$. Specifically, our algorithm attempts to build candidate solutions at $t$ by growing candidate solutions at $t'$.

The main challenge with this approach is storing candidate solutions. There can be an exponential number of candidate solutions at a given node $t$, so we cannot simply store all candidate solutions. Generally, dynamic programs on tree decompositions work by storing some local representation of candidate solutions at $t$, where a local representation is a description of a candidate solution only in terms of vertices and edges in the bag $X_t$. Two candidate solutions with the same local representation are typically interchangeable in the sense that one candidate solution can be extended to a complete solution if and only if the other can too. The number of these local representations at $t$ is typically a function of the size of $X_t$, which allows for FPT algorithms parameterized by the treewidth.

The local representation of candidate solutions for our problems should have several properties. First, they should represent a candidate solution using only simplices in $X_t$. Second, they should retain enough information that we can verify that a subcomplex is a candidate solution, i.e. it could be extended to a surface in $K$. In particular, we should be able to deduce information about the links of simplices in $X_t$ from the local representation. The first and second properties are at odds, as even if a simplex $\sigma$ is contained in $X_t$, the link of $\sigma$ need not be contained in $X_t$. Finally, we should be able to deduce the homeomorphism class of a candidate solution from the local representation. Again, this property is at odds with the first property, as topological properties like the genus and orientability of a surface are global, not local, properties of a surface. One of our contributions is introducing a data structure to store local representations of candidate solution with each of these properties called the **annotated cell complex**.

A (non-annotated) **cell complex** is an algebraic representation of a surface that was originally introduced by Ahlfors and Sario [1] to prove the Classification Theorem of Compact Surfaces. Intuitively, a cell complex is a collection of disks, called **faces**, joined by shared edges in their boundaries. The faces in a cell complex differ from triangles in a simplicial complex as the faces in a cell complex can have more than three edges in their boundary. A definition of cell complex and a discussion of their properties can be found in Section 3.2 of the extended version of this paper [5].

The advantage of using cell complexes rather than simplicial complexes to store surfaces is that there is a simple equivalence relation that partitions cell complexes into homeomorphism classes. This is of obvious benefit as the surface $S$ we are looking for may be specified by its homeomorphism class, but there is a secondary benefit. We define a set of **equivalence-preserving moves**, operations on cell complexes that preserve their homeomorphism class. We use these moves to compress the local representation of each candidate solution we keep during our algorithm. The most important benefit that these moves provide is the ability to merge two faces that share an edge.

To see why merging faces is helpful, suppose that we have a candidate solution $\Sigma$ at a node $t$ that is represented as a cell complex. We would like to store a local representation of $\Sigma$ using only edges in $X_t$. There would then be a bounded number of local representations of candidate solutions at a node $t$, as there are a bounded number of edges in $X_t$. To this end, each time we forget an edge $e$, we would like to merge the two faces incident to $e$ into a single face. See Figure 4, left panel.

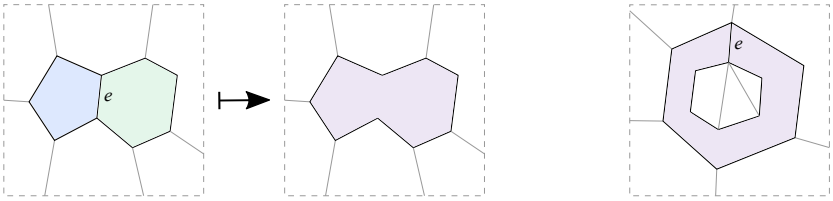



**Figure 4** Left: The edge $e$ is removed by merging the two incident faces. Right: The edge $e$ appears twice on the boundary of the same face, so $e$ cannot be removed by merging incident faces as this would make the interior of the face an annulus. We use annotated cell complexes to remove $e$.

The idea of merging faces when we forget $e$ works unless $e$ is incident to the same face twice; the right panel of Figure 4 gives an example. After merging some faces, it is possible that a face may have two edges on its boundary identified. If two edges on the boundary of the same face are identified, then we can no longer remove these edges by merging their incident faces, as then the interior of this face would no longer be a disk.

We therefore modify the definition of cell complex to allow for a more general type of face. Our first change is to allow a face to be a disk with multiple boundary components like in Figure 4, but we need to go a step further. Topological features like handles, crosscaps, and boundaries in cell complexes are the result of a single face having edges on its boundary identified in certain ways; thus, we need a way of removing the edges that constitute these topological features. An **annotated cell complex** annotates each face with the number of topological features like handles, crosscaps, and boundaries on this face, rather than storing these features explicitly with edges. In effect, an annotated cell complex is a representation of a surface where the interior of a face is allowed to be any compact connected surface.

## 4 Overview of the lower bounds

Here we present the main ideas that go into the proof of our lower bounds. The omitted details can be found in Section 4 of the extended version of this paper [5].

▶ **Theorem 2.** *Assuming the ETH, no algorithm can solve* Subsurface Recognition, Sum-of-Genus Subsurface Recognition *or* Subsurface Packing *in* $2^{o(k \log k)} n^{O(1)}$ *time. The parameter $k$ denotes the width of a given (nice) path decomposition of the Hasse diagram of the input simplicial complex.*

Since every path decomposition is also a tree decompositions, the treewidth of a graph is never higher than its pathwidth. Theorem 2 therefore implies that none of our problems can be solved in $2^{o(k \log k)} n^{O(1)}$ time, where $k$ is now the treewidth of the Hasse diagram.

We focus on proving the result for Subsurface Recognition. After this, it will be easy to modify our arguments to prove similar results for the two other problems. At a conceptual level there are two parts to the proof.

1. Define a reduction from Directed Cycle Packing to Subsurface Recognition.
2. Show that the reduction can always be chosen so that the pathwidth of the output space is bounded by some linear function of the pathwidth of the input graph.

### 4.1 The reduction

Directed Cycle Packing asks us to find as many vertex disjoint cycles in a graph as possible (see Figure 5). This problem is essentially a directed, 1-dimensional version of the SP problem, since we know that the only compact 1-manifolds are circles (cycles) and closed intervals (paths).

▶ **Problem 5.** *The* Directed Cycle Packing *(DCP) problem*
*INPUT: A directed graph $D$ on $n$ vertices and an integer $\ell$.*
*PARAMETER: The pathwidth $k$ of $D$.*
*QUESTION: Does $D$ contain $\ell$ vertex disjoint cycles?*

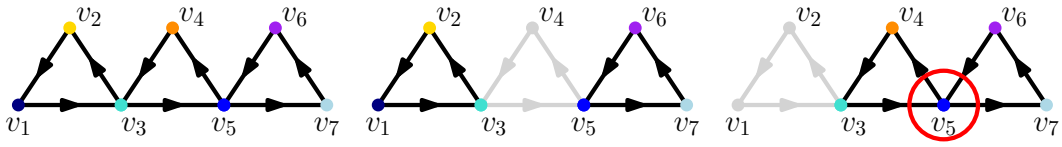



**Figure 5** A directed graph $D$ (left), two vertex disjoint cycles contained in $D$ (middle) and two cycles in $D$ intersecting at a common vertex (right). This will be a guiding example for this section.

The DCP problem is a good starting point for our reduction not only because of its similarity to the SP problem but also because of the following theorem.

▶ **Theorem 3** ([17]). *Assuming the ETH, the DCP problem cannot be solved in $2^{o(k \log k)} n^{O(1)}$ time, where the parameter $k$ denotes the width of a given (nice) path decomposition of the input graph.*

Given a digraph $D$, the reduction will construct a 2-dimensional simplicial complex $Y$ that contains $\ell$ disjoint tori if and only if $D$ contains $\ell$ vertex disjoint cycles. In fact, we show that the only connected subsurfaces without boundary in $Y$ are tori and that these are in a bijection with the directed cycles in $D$. Furthermore, any pair of these tori are *disjoint* if and only if the corresponding directed cycles are vertex disjoint.

In Figure 6 we introduce some shorthand notation that will help make the reduction clearer. Each column of the figure shows a different component that we will use when constructing the space $Y$. The first row shows the shorthand notation. The second row shows the "topological space" that the notation represents. The third and fourth row indicate which triangulation we use to represent this space.

The first column shows a cylinder, $S_1$. The second column shows a space $S_2$ consisting of two cylinders, $X_1'$ and $X_2'$. These cylinders are glued together at a single interior point, called a (0-dimensional) **singularity**. The third column shows a space $S_3$ consisting of three cylinders $X_1''$, $X_2''$ and $X_3''$, each with a single boundary component attached to the same circle. The fourth and final column shows the space $S_4$, obtained by gluing $S_2$ and $S_3$ together. More precisely, $S_4$ also consists of three cylinders, $X_1 = X_1' \cup X_1''$, $X_2 = X_2' \cup X_2''$ and $X_3 = X_3''$, each having a single boundary component attached to the same circle. Additionally, $X_1 \cup X_2$ contains a 0-dimensional singularity.

We establish some important properties of the spaces $S_1, S_2, S_3$ and $S_4$ from Figure 6. In order to describe these properties we temporarily extend the notion of a "boundary", a term usually reserved for manifolds, to the world of simplicial complexes. In the remainder of this section, the word boundary will refer to the closure of the set of 1-simplices in $X$ that only have a single coface. We denote this subcomplex by $\partial(X)$.

▶ Remark 4. Let $S_1, S_2, S_3$ and $S_4$ be the spaces introduced in Figure 6.

1. The only (non-empty) 2-manifold $X \subseteq S_1$ where $\partial(X) \subseteq \partial(S_1)$ is $S_1$ itself.
2. The only 2-manifolds $X \subseteq S_2$ where $\partial(X) \subseteq \partial(S_2)$ are $X_1'$ and $X_2'$.
3. The only 2-manifolds $X \subseteq S_3$ where $\partial(X) \subseteq \partial(S_3)$ are $X_1'' \cup X_2''$, $X_1'' \cup X_3''$ and $X_2'' \cup X_3''$.
4. The only 2-manifolds $X \subseteq S_4$ where $\partial(X) \subseteq \partial(S_4)$ are $X_1 \cup X_3$ and $X_2 \cup X_3$.

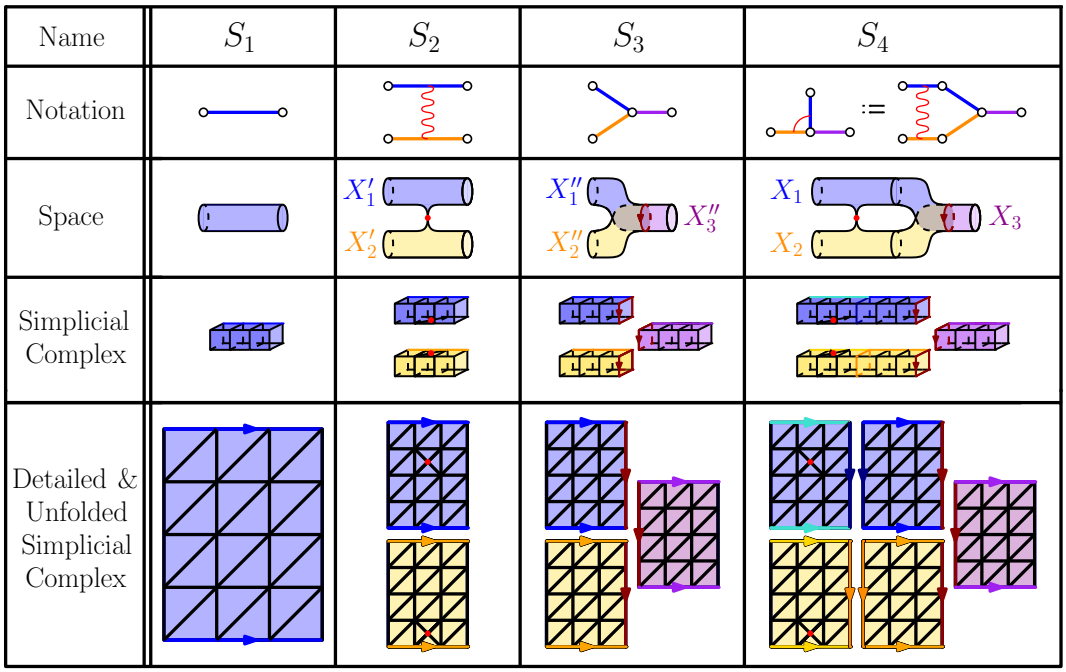



**Figure 6** Shorthand notation for specific triangulations of $S_1, \ldots, S_4$ that we will use frequently throughout the section.

That these properties holds is intuitively obvious. Formally, this can be proved easily by brute force: Simply go through all the 2-simplices in $S_i$ and assume that it is contained in a submanifold $X$. It is then easy to see which adjacent 2-simplices must necessarily also be contained in the same submanifold. Whenever a choice has to be made, simply branch and try all possibilities.

The reduction is perhaps best understood in terms of vertex gadgets and edge gadgets. In particular, Figure 7 shows how a vertex $\xi$ is mapped to the vertex gadget $Y^\xi$, using the notation from Figure 6. The figure also shows six edge gadgets (in black), three corresponding to the edges entering $\xi$ and three corresponding to the edges leaving $\xi$. The edge gadgets are unlabeled in the figure but can be identified by the vertex gadgets they are attached to. We think of each vertex gadget as composed of two sub-cylinders, one half for the incoming edge gadgets and the other half for outgoing edge gadgets. To better see this separation we draw the vertex gadget with a U-turn at the location of this divide in our figures.

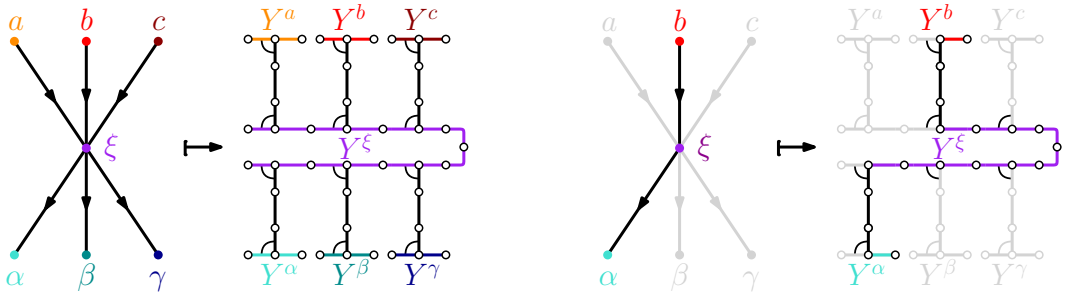



**Figure 7** A local view of how a vertex $\xi$ is mapped to its vertex gadget $Y^\xi$ (left) and an illustration of how a directed cycle passing through the vertex $\xi$ is mapped to a submanifold in the space (right).

Each edge gadget is connected to the vertex gadgets corresponding to each of its two ends through a copy of $S_4$. The edge gadget contains the cylinder $X_1$ while the vertex gadget contains the other cylinders $X_2$ and $X_3$. Both the incoming and outgoing part of the vertex gadget consists primarily of a sequence of smaller cylinders, $X_2 \cup X_3$, one for each incoming/outgoing edge. The boundary of the $X_3$ corresponding to one edge is attached to the boundary of the copy of $X_2$ corresponding to the next edge. The boundary of the "last" $X_3$ of the incoming edges is attached to one boundary component of a single additional cylinder, while the "last" $X_3$ of the outgoing edges is attached to the other boundary component.

By repeated use of property 4, any potential manifold contained in this space must contain precisely one incoming and one outgoing edge gadget per vertex, assuming the manifold is not allowed to have a boundary. This is illustrated in Figure 8. This figure also shows the importance of the 0 dimensional singularities in the reduction. The resulting space could otherwise contain tori that do not correspond to any directed cycle. An example of the correspondence between disjoint tori and vertex disjoint directed cycles is shown in Figure 9.

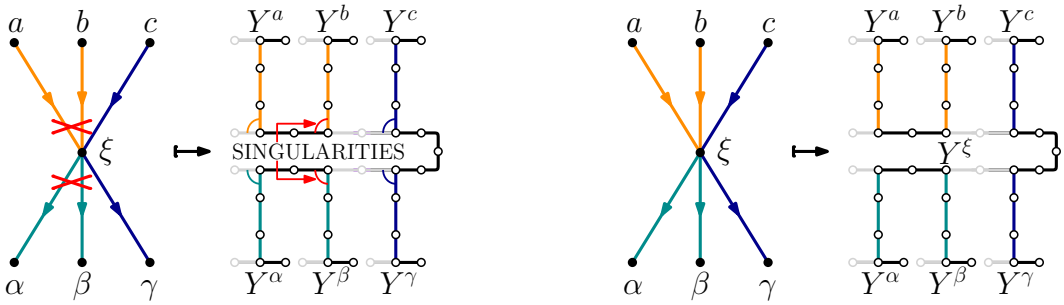



**Figure 8** The leftmost figure shows how the singularities keeps "badly behaved" subcomplexes from becoming manifolds. The rightmost figure shows how the reduction would fail without the use of singularities between the vertex gadgets and edge gadgets.

We see in Figure 9 that we can associate any pair of vertex disjoint cycles in the input graph to a pair of non-intersecting tori in the output space in an obvious way. Concretely, a cycle is mapped to a torus by sending the edges to edge gadgets and by then connecting these through the vertex gadgets. This association turns out to be a bijection with an inverse that maps a submanifold to the set of edges whose edge gadgets intersects the submanifold. That this inverse is well-defined is proved for the pathwidth-preserving reduction in Section 4.5 of the extended version of the paper; see [5].

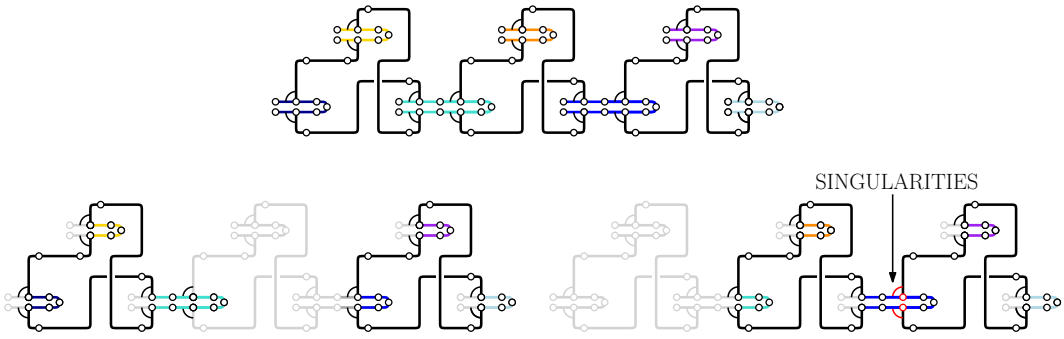



**Figure 9** An illustration of how the graph from Figure 5 is mapped to spaces and how valid/invalid subsets of edges are mapped to manifolds/non-manifolds respectively.

## 4.2 Pathwidth preservation

The main idea of this section can be summarized in a single sentence: By carefully choosing the order in which we attach edge/vertex gadgets to each other, we can make a space that has a similar structure to a nice path decomposition of the input graph. This is an absolutely necessary "fine tuning" of the reduction we saw in the previous subsection. Without it, we have no guarantee that the Hasse diagram of the space we construct will have low pathwidth. In fact, if the ordering is chosen in an adversarial way, we may end up mapping a graph of bounded pathwidth to spaces whose Hasse diagram has arbitrarily large pathwidth.

We discuss this in detail in Section 4.4.1 of the extended version of the paper [5], but the rough idea is captured in Figure 10. Here we see a graph of pathwidth 2 being mapped to two very different spaces. While both are constructed in a way that is compatible with the reduction described in Section 4.1, intuitively it is the topmost space that has retained most of the "pathlikeness" of the input graph. This intuition is reflected in the fact that the Hasse diagram of the lower figure really is higher than that of the one above it.

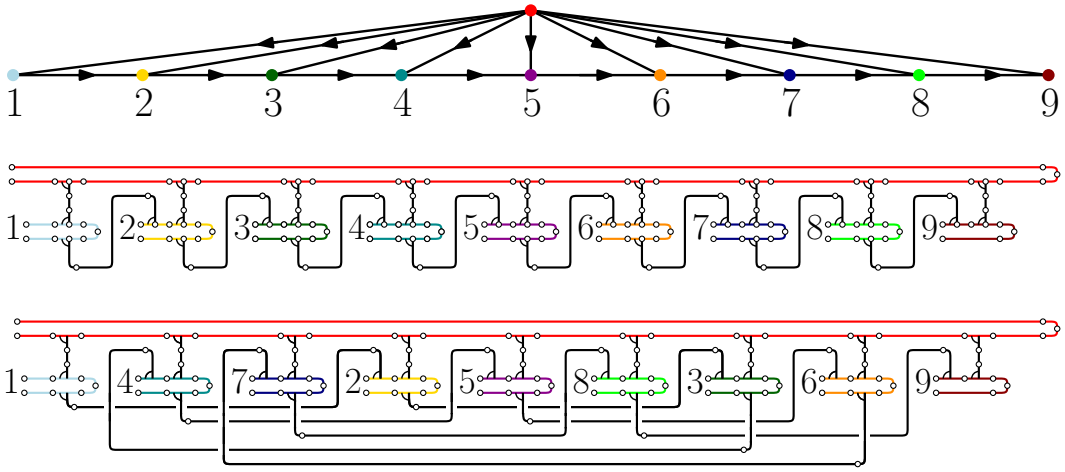



**Figure 10** A directed graph of pathwidth 2 (top) together with a "sensible" version of the reduction explained in Section 4.1 (middle) and an "adversarial" version of the reduction (bottom).

It can be quite hard to prove lower bounds on the path-/treewidth of graphs/spaces, but for this particular family it is reasonably straightforward. Once you generalize the above "adversarial" layout for any input graph on $n^2 + 1$ vertices there is a nice geometric argument that the Hasse diagram of the outputted space will always contain an $n \times n$-grid as a graph minor. It is well known that such graphs have treewidth at least $n$ which gives us our desired lower bound.

If we are given a less structured graph than the one we saw in Figure 10, it might be hard to see how we can best glue the gadgets together. Our way around this is to construct a space where the order in which vertex gadgets are attached to each other is determined by the order in which the nodes are forgotten in the nice path decomposition of the input graph. The idea is that a vertex gadget is attached to a neighbouring vertex gadget in the current bag when it (or its neighbour) is forgotten, see Figure 11.

The way we make the above idea precise is rather technical. It is in essence all about structural induction over the given nice path decomposition, which we use to construct a nested sequence of spaces $Y_0 \subset \cdots \subset Y_r$, one for each bag. We also compute an accompanying path decomposition for the Hasse diagram of each of the nested spaces. These path decompositions

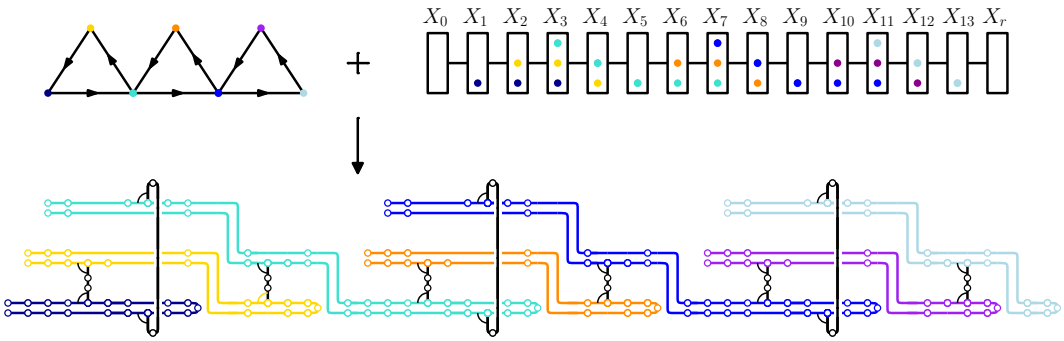



**Figure 11** An illustration of how the graph from Figure 5 (top left) is mapped to a space (bottom) having the same "structure"/"order" as the given nice path decomposition (top right) of the graph.

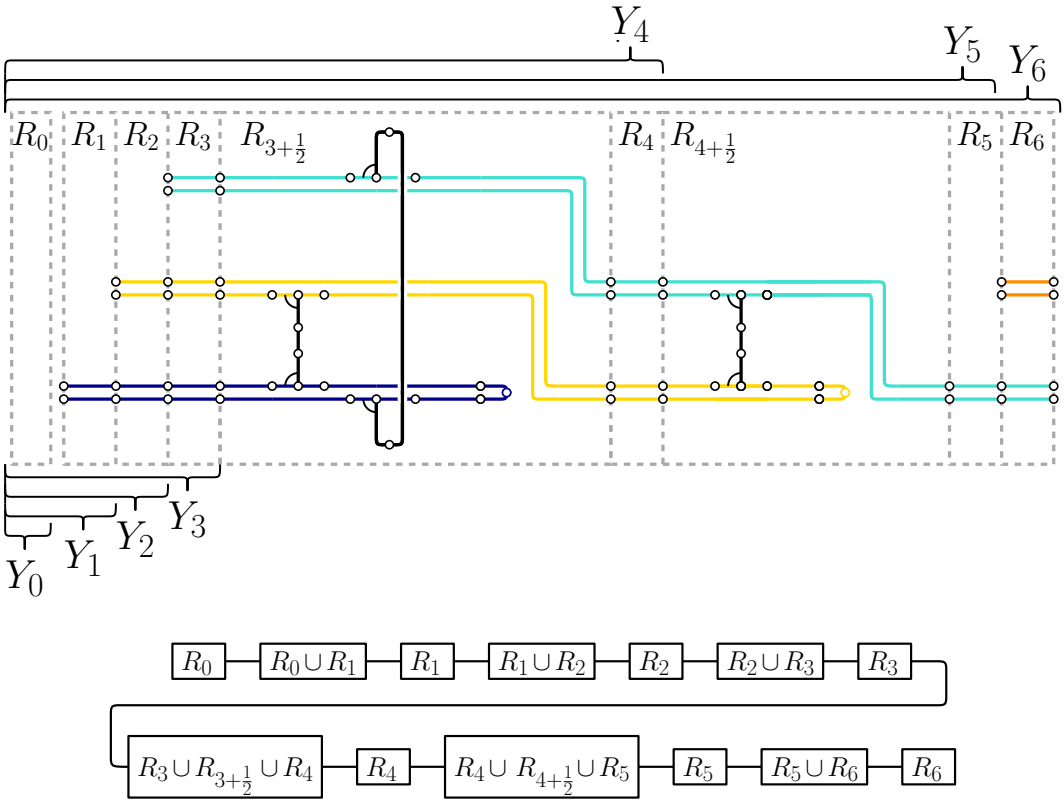



**Figure 12** The space $Y_6$ (top) associated to bag $X_6$ in the nice path decomposition of the graph in Figure 11. The location of the sub complexes $Y_0 \subset \cdots \subset Y_5$ are indicated. Below is the path decomposition of $Y_6$. Path decomposition of the other spaces $Y_i, 0 \leq i \leq 5$ are all present as the path decomposition induced by "sub-paths" starting at the bag containing $R_0$ and ending at the bag containing $R_i$.

are not optimal, but their width is bounded above by the width of the inputted nice path decomposition times a constant, which is sufficient for our purposes. The induction involves going trough a lot of elementary claims about the space we have constructed at each step. For details on this, see Section 4.4.2 of the extended version of this paper [5]. The space $Y_6$ and its path decomposition are shown in Figure 12.

## 5 Conclusion

In this paper, we consider the parameterized complexity of several variants of the problem of finding surfaces in 2-dimensional simplicial complexes with respect to the treewidth of the Hasse diagram. We give ETH-optimal algorithms for the Sum-of-Genus Subsurface Recognition and Subsurface Packing problems. We also give an ETH-based lower bound for Subsurface Recognition and an FPT algorithm for Connected Subsurface Recognition. Several questions surrounding subsurface recognition remain open, such as

- whether the algorithm presented in this paper for Connected Subsurface Recognition is ETH-optimal;
- whether or not the Subsurface Recognition Problem is W[1]-hard when parameterized by the treewidth of the Hasse diagram.

Future work could either attempt to find better parameterized algorithms or prove stronger lower bounds for these problems.

## References


1 L.V. Ahlfors and L. Sario. *Riemann Surfaces*. Princeton mathematical series. Princeton University Press, 2015. URL: https://books.google.com/books?id=4C4PAAAAIAAJ.

2 Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. doi:10.1137/0608024.

3 László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 684–697, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2897518.2897542.

4 Bhaskar Bagchi, Basudeb Datta, Benjamin A. Burton, Nitin Singh, and Jonathan Spreer. Efficient Algorithms to Decide Tightness. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2016.12.

5 Mitchell Black, Nello Blaser, Amir Nayyeri, and Erlend Raa Vågset. ETH-tight algorithms for finding surfaces in simplicial complexes of bounded treewidth. *CoRR*, abs/2203.07566, 2022. arXiv:2203.07566.

6 Nello Blaser, Morten Brun, Lars M. Salbu, and Erlend Raa Vågset. The parameterized complexity of finding minimum bounded chains. *CoRR*, 2021. arXiv:2108.04563.

7 Nello Blaser and Erlend Raa Vågset. Homology localization through the looking-glass of parameterized complexity theory, 2020. arXiv:2011.14490.

8 Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016. doi:10.1137/130947374.

9 Benjamin Burton, Sergio Cabello, Stefan Kratsch, and William Pettersson. The Parameterized Complexity of Finding a 2-Sphere in a Simplicial Complex. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2017.18.

10 Benjamin Burton and Rodney Downey. Courcelle's theorem for triangulations. *Journal of Combinatorial Theory, Series A*, 146, March 2014. doi:10.1016/j.jcta.2016.10.001.

11 Benjamin A. Burton, Thomas Lewiner, João Paixão, and Jonathan Spreer. Parameterized complexity of discrete Morse theory. *ACM Transactions on Mathematical Software*, 42(1):1–24, March 2016. doi:10.1145/2738034.

12 Benjamin A. Burton and Jonathan Spreer. The complexity of detecting taut angle structures on triangulations. *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2013. doi:10.1137/1.9781611973105.13.

13 A.V. Chernavsky and V.P. Leksine. Unrecognizability of manifolds. *Annals of Pure and Applied Logic*, 141(3):325–335, 2006. Papers presented at the Second St. Petersburg Days of Logic and Computability Conference on the occasion of the centennial of Andrey Andreevich Markov, Jr. doi:10.1016/j.apal.2005.12.011.

14 Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. Association for Computing Machinery. doi:10.1145/800157.805047.

15 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

16 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, 2015. doi:10.1007/978-3-319-21275-3.

17 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

18 Wolfgang Haken. Theorie der normalflächen. *Acta Mathematica*, 105(3):245–375, September 1961. doi:10.1007/BF02559591.

19 Sergei Ivanov. Computational complexity. MathOverflow. URL: https://mathoverflow.net/q/118428.

20 A. Markov. The insolubility of the problem of homeomorphy. *Dokl. Akad. Nauk USSR*, 12(2):218–220, 1958.

21 Hyam Rubinstein. The solution to the recognition problem for $\mathbb{S}^3$. Lecture, 1992.

22 Abigail Thompson. Thin position and the recognition problem for $\mathbf{S}^3$. *Mathematical Research Letters*, 1(5):613–630, 1994.