AN ABSTRACT OF THE THESIS OF

Mitchell Black for the degree of Master of Science in Computer Science presented on May 27, 2021.

Title: Computing Bounded Chains and Surfaces in a Simplicial Complex with Bounded-treewidth 1-skeleton

Abstract approved: ______________________________

Amir Nayyeri

We consider three problems on simplicial complexes: the Optimal Bounded Chain Problem, the Optimal Homologous Chain Problem, and 2-Dim-Bounded-Surface. The Optimal Bounded Chain Problem asks to find the minimum weight d-chain in a simplicial complex K bounded by a given $(d-1)$-chain, if such a d-chain exists. The Optimal Homologous Chain problem asks to find the minimum weight $(d-1)$-chain in K homologous to a given $(d-1)$-chain. 2-Dim-Bounded-Surface asks whether or not there is a subcomplex of K homeomorphic to a given compact, connected surface bounded by a given subcomplex. All three problems are NP-hard, and the first two problems are hard to approximate within any constant factor assuming the Unique Games Conjecture. We prove that all three problems are fixed-parameter tractable with respect to the treewidth of the 1-skeleton of K.

©Copyright by Mitchell Black
May 27, 2021
All Rights Reserved

Computing Bounded Chains and Surfaces in a Simplicial Complex with Bounded-treewidth 1-skeleton

by

Mitchell Black

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 27, 2021
Commencement June 2021

Master of Science thesis of Mitchell Black presented on May 27, 2021.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Mitchell Black, Author

ACKNOWLEDGEMENTS

I am very grateful to have been advised by Amir Nayyeri in my research and in writing this thesis. I would like to thank Amir for his patience (especially after a year of Zoom meetings), his steady guidance, and his kindness. I was told graduate school might make me enjoy math less, but working with Amir, I've only enjoyed it more.

I would like to thank my committee—Cora Borradaile, Christine Escher, Mike Bailey, and Clayton Petsche—for volunteering their time and for their guidance. I would also like to thank my collaborators Will Maxwell and Eli Winkelman for their hard work and insight to our research.

Lastly, I would like to thank my family for a lifetime of support: my sister Margeaux, my brother Curtis, my father Greg, and, especially, my mother Ann. Without my mother, I wouldn't have made it through the past year, nor would I have made it to graduate school in the first place.

My research is supported by NSF grant CCF-1816442.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

Chapter 1: Introduction

A common type of problem in Computational Topology is to find a subspace that has a certain property within a topological space. In graphs, the Shortest Path problem asks what is the shortest path between a pair of vertices. In higher dimensional complexes, determining if two $(d-1)$-cycles are homologous is equivalent to determining if their sum bounds a d-chain. Similarly, determining if a knot is unknotted is equivalent to determining if the knot bounds an embedded disk.

Some of these problems, like determining the minimal genus of a surface bounded by a knot, are NP-hard [1, 15, 25, 21, 26]. In other cases, existing algorithms for these problems, like determining if two cycles are homologous, are based on linear algebraic methods with running times $\Omega(n^\omega)$ [11], where $\omega \geq 2$ is the matrix multiplication constant. It is natural to ask whether more efficient algorithms exist for any special family of simplicial complexes. In this paper, we study a subset of these problems for simplicial complexes with bounded-width tree decompositions of their 1-skeleton, i.e. bounded treewidth.

Tree decompositions have seen much success as an algorithmic tool on graphs. Often, bounded-treewidth graphs admit polynomial-time solutions to otherwise hard problems. A highlight of the algorithmic application of tree decompositions is Courcelle's Theorem [17], 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.[1]

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, 8, 9, 10]. Other graphs that have been used are the incidence graph of the 1- and 2-simplices [9] and the 1-skeleton [4].

Our algorithms use a tree decomposition of the 1-skeleton. The problems we consider

[1]Interested readers might read Chapter 7 of the book by Cygan et. al. [18] for an introduction to the algorithmic use of tree decompositions.

ask to find a chain or a subcomplex of a simplicial complex with certain properties. There can be exponentially many chains or subcomplexes, so we cannot simply check whether each chain or subcomplex has the desired property. Our approach is to incrementally build these chains or subcomplexes, checking along the way that they have the desired property. Tree decompositions of the 1-skeleton define a recursive structure on the complex that we use to incrementally build our solutions.

1.1 Our Results

We now summarize the main results of our paper. Throughout, n is the number of vertices in the input simplicial complex and k is the treewidth of the 1-skeleton of the input simplicial complex. Formal definitions of the terms used will be given in Chapter 3.

The Optimal Bounded Chain Problem (OBCP) asks to find the minimum weight d-chain c of a simplicial complex K bounded by a given $(d-1)$-chain b, if such a chain c exists. We work with chain groups with coefficients in $\mathbb{Z}_2$, so a d-chain can therefore be thought of as a set of d-simplices. The weight of a d-chain is the number of d-simplices it contains. We show that OBCP is fixed-parameter tractable with respect to the treewidth of the 1-skeleton.

Theorem 1.1. *OBCP can be solved in* $2^{O\left(k^d\right)}n \subset 2^{O\left(k^k\right)}n$ *time.*

As a corollary, our algorithm for OBCP can determine whether or not two $(d-1)$-chains b and h are homologous. The naive algorithm for this problem is to solve the system of linear equation $\partial c = b + h$ for c, which takes $O(n^\omega)$ time. Our algorithm runs in linear time on a bounded-treewidth simplicial complex.

Corollary 1.2. *We can determine whether two* $(d-1)$*-chains are homologous in* $2^{O\left(k^d\right)}n \subset 2^{O\left(k^k\right)}n$ *time.*

The Optimal Homologous Chain Problem (OHCP) asks to find the minimal weight $(d-1)$-chain h homologous to a given $(d-1)$-chain b. If the chains b and h are cycles, OHCP is also known as Homology Localization. We show that OHCP is fixed-parameter tractable algorithm with respect to the treewidth of the 1-skeleton.

Theorem 1.3. *OHCP can be solved in* $2^{O\left(k^d\right)}n \subset 2^{O\left(k^k\right)}n$ *time.*

Our results for both OBCP and OHCP easily generalize to weighted complexes. We discuss this at the end of Chapter 5.

The problem 2-Dim-Bounded-Surface (2DBS) is inspired by two problems in knot theory: the Unknot Recognition problem and the Knot Genus problem. The Unknot Recognition problems asks whether a given knot is the unknot. A knot is the unknot iff it bounds a disk, so one way to solve the Unknot Recognition problem is to decide whether the knot bounds a disk. The Knot Genus problem asks what is the minimal genus of a surface the knot bounds.

2DBS asks to find a subcomplex S of a simplicial complex K homeomorphic to a given compact, connected surface Ψ with given boundary $B \subset K$. The surface Ψ is specified by its genus and orientability; the number of boundary components of Ψ is implied by B. The difference between 2DBS and Unknot Recognition and Knot Genus is that 2DBS is only concerned with surfaces in the 2-skeleton, while the knot problems consider any embedded surfaces. Embedded surfaces are not necessarily contained in the 2-skeleton and may intersect higher-dimensional simplices. We show that 2DBS is fixed-parameter tractable with respect to the treewidth of the 1-skeleton.

Theorem 1.4. *2DBS can be solved in $k^{O(k^2)}gn \subset k^{O(k^2)}n^3$ time, where g is the genus of the input surface Ψ.*

In Section 8.11, we discuss how our algorithm can be adapted to handle optimization variants of 2DBS. We also discuss how our algorithm can find a surface with a specified *number* of boundary components, rather than a specified boundary.

Chapter 2: Related Work

In this chapter, we review the previous work on the problems OBCP, OHCP, and 2DBS.

OBCP. Dunfield and Hirani showed that OBCP is NP-hard [21]. Borradaile, Maxwell, and Nayyeri showed OBCP over coefficients in $\mathbb{Z}_2$ is hard to approximate within some constant factor if $P \neq NP$ and hard to approximate with any constant factor assuming the Unique Games Conjecture, even for embedded complexes [6]. On the positive side, there are algorithms to solve OBCP in embedded complexes [6], d-manifolds [12], with coefficients in $\mathbb{R}$ [14], in special cases with coefficients in $\mathbb{Z}$ [21], and with norms other than the Hamming norm [16].

Our algorithm for OBCP can also be used to test whether two chains are homologous and whether a cycle is null-homologous. The naive algorithm for both of these problems is to solve a system of linear equations. Dey gives algorithms to test whether a cycle embedded on a surface is null-homologous that runs in $O(n + l)$ time, where l is the length of the cycle, or with a $O(n)$ time preprocessing step, $O(g + l)$ time, where g is the genus of the surface [19]. There is an algorithm for testing the homology class of a cycle that runs in $O(gl)$ time with a $O(n^{\omega})$ preprocessing step, where g is the rank of the homology group and l is the number of simplices in the cycle [11].

OHCP and Homology Localization. Homology Localization is a special case of OHCP where the input chain is a cycle; thus, hardness results for homology localization hold for OHCP as well. Chen and Freedman showed that Homology Localization over coefficients in $\mathbb{Z}_2$ is NP-hard to approximate within any constant factor [15]. Chambers, Erickson, and Nayyeri showed that Homology Localization is NP-hard even for surface-embedded graphs [13]. Borradaile, Maxwell, and Nayyeri showed that OHCP is hard to approximate within any constant factor even for embedded complexes, assuming the Unique Games Conjecture [6].

On the positive side, there are parameterized algorithms to solve homology localization for 1-cycles in surface embedded graphs [13, 22] and 1-cycles in simplicial complexes

[11], and to solve OHCP in $(d+1)$-manifolds [6]. There is an algorithm to solve OHCP with a norm other than the Hamming norm [16]. There are also linear programming based approaches to solve OHCP in special cases with coefficients in $\mathbb{Z}$ [20].

2DBS. The problem 2-Dim-Bounded-Surface is a generalization of 2-Dim-Sphere, the problem of finding a subcomplex homeomorphic to the 2-sphere. In this case, Ψ is the sphere and the given boundary B is empty. Ivanov showed 2-Dim-Sphere is NP-hard [25]. Burton, Cabello, Kratsch, and Pettersson showed 2-Dim-Sphere is W[1]-Hard when parameterized by the number of 2-simplices of the sphere [7]. Burton, Cabello, Kratsch, and Pettersson also gave an algorithm for 2-Dim-Sphere that runs in $2^{O(t)}n^{O(\sqrt{t})}$ time, where t is the number of triangles in the sphere. In light of the $W[1]$-hardness result, their algorithm is optimal assuming the Exponential Time Hypothesis.

Chapter 3: Mathematical Background

In this chapter, we present the mathematical background needed for the rest of this thesis.

3.1 Simplicial complexes

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 τ is a ***face*** of a simplex σ if $\tau \subset \sigma$. Likewise, σ is a ***coface*** of τ. The simplices σ and τ are ***incident***.

A simplex σ with $|\sigma| = d + 1$ is a ***d-simplex***. The set of all d-simplices in K is denoted K_d. The ***d-skeleton*** of K is $K^d = \cup_{i=0}^{d} K_d$. In particular, the 1-skeleton of K is a graph. The ***dimension*** of a simplicial complex is the minimal integer d such that $|\sigma| \leq d + 1$ for each $\sigma \in K$. 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 union $V = \cup_{\sigma \in K} \sigma$ is the set of ***vertices*** of a simplicial complex. Each simplex in K is a subset of V. A subset $U \subset V$ defines a subcomplex of K. The ***subcomplex induced by U*** is $K[U] = \{\sigma \in K \mid \sigma \subset U\}$.

Let $\Sigma \subset K$. The ***closure*** of Σ is $\operatorname{cl}\Sigma := \{\tau \subset \sigma \mid \sigma \in \Sigma\}$. The closure $\operatorname{cl}\Sigma$ is defined only by the set Σ and not the complex K. The ***star*** of Σ is $\operatorname{st}_K \Sigma := \{\sigma \in K \mid \exists \tau \in \Sigma \text{ such that } \tau \subset \sigma\}$. The ***link*** of Σ is $\operatorname{lk}_K \Sigma = \operatorname{cl}\operatorname{st}_K \Sigma - \operatorname{st}_K \operatorname{cl}\Sigma$. If $\sigma = \Sigma$ is a single simplex, then $\operatorname{lk}_K \sigma = \{\tau \in \operatorname{cl}\operatorname{st}_K \sigma \mid \tau \cap \sigma = \emptyset\}$. Note that for any simplex $\tau_1 \in K$ and any simplex $\tau_2 \in \operatorname{lk}_K \tau_1$ that τ_1 and τ_2 are incident to a common coface in $\operatorname{st}_K \tau_1$.

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 ***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. A vertex $v \in S$ such that $\mathrm{lk}_S\, v$ is a simple path is a ***boundary vertex***. A vertex $v \in S$ such that $\mathrm{lk}_S\, v$ is a simple cycle is an ***interior vertex***. Each edge $e \in S$ has link $\mathrm{lk}_S\, e$ that is either one or two vertices because of the conditions placed on the links of the 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 ∂S. The boundary ∂S is a collection of simple cycles.

3.2 Homology

We work with homology with coefficients over $\mathbb{Z}_2$. Let K be a simplicial complex. The ***d^{th} chain group*** $C_d(K)$ of K is the free abelian group with coefficients over $\mathbb{Z}_2$ generated by K_d. An element $\gamma \in C_d(K)$ is a ***d-chain***. A d-chain γ with coefficients over $\mathbb{Z}_2$ naturally corresponds to a set of d-simplices in K. We overload notation and use γ to also refer to the set defined by the chain γ. The ***weight*** of a chain γ is the number of simplices γ contains and is denote $||\gamma||$; alternatively, $||\gamma||$ is the ***Hamming norm*** of γ. Addition of two d-chains takes the symmetric difference of the sets corresponding to these chains.

Let $\sigma \in K_d$. The ***boundary*** of σ is $\partial\sigma = \sum_{v\in\sigma} \sigma \setminus \{v\}$. The boundary map linearly extends the notion of boundary from d-simplices to d-chains. The ***boundary map*** is the homomorphism $\partial_d : C_d(K) \to C_{d-1}(K)$ such that $\partial_d\gamma = \sum_{\sigma\in\gamma} \partial\sigma$. When obvious, we drop the d and simply write ∂_d as ∂. A $(d-1)$-chain β ***bounds*** a d-chain γ if $\partial\gamma = \beta$, and the chain γ ***spans*** β.

The concatenation $\partial_{d-1} \circ \partial_d = 0$, the zero map. As $C_d(K)$ is abelian, then $\mathrm{im}\, \partial_d \subset \ker \partial_{d-1}$ is a normal subgroup. The ***(d-1)st homology group*** is the quotient group $H_{d-1}(K) = \ker \partial_{d-1} / \mathrm{im}\, \partial_d$. An element $\beta \in \ker \partial_{d-1}$ is a ***(d-1)-cycle***. A $(d-1)$-cycle β is ***null-homologous*** if $\beta = \partial\gamma$ for some d-chain γ. Two $(d-1)$-chains b and h are ***homologous*** if $b + h$ is null-homologous.

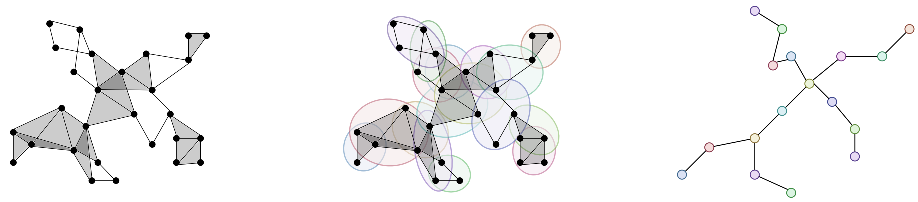

Figure 3.1: Left: A simplicial complex. Right and Center: A (not nice) tree decomposition of the simplicial complex. Each node of the tree corresponds to a subset of the vertices of the simplicial complex

3.3 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 $X = \{X_t \subset V \,|\, t \in I\}$ 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 a tree decomposition is $\max_{t\in I}|X_t| + 1$. The ***treewidth*** of a graph is the minimum width of any tree decomposition of G. Computing the treewidth of a graph is NP-hard [3], but there are algorithms to compute tree decompositions that are within a constant factor of the treewidth, e.g. see [5].

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' such that 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' such that 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 two children t', t'' such that $X_t = X_{t'} = X_{t''}$. A nice tree decomposition can always be computed from a tree decomposition, as described in Lemma 3.1.

Lemma 3.1 (Lemma 7.4 of [18])**.** *Given a tree decomposition (T, 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.*

Let K be a simplicial complex. Let (T, X) be a tree decomposition of the 1-skeleton of K. Each vertex and edge of K is contained in a bag of the tree decomposition by definition. In general, all simplices of K are contained in a bag of the tree decomposition. This is proved in Corollary 3.3, but to prove this, we need a lemma.

Lemma 3.2. *Let (T, X) be a tree decomposition of a graph $G = (V, E)$. If $Q \subset V$ forms a clique in G, then there is a node $t \in I$ such that $Q \subset X_t$. Moreover, the set of nodes $\{t \in I \,|\, Q \subset X_t\}$ form a connected subtree of T.*

Proof. A proof that Q is contained in X_t for some node t is given in [24]. For the second claim, let t_1 and t_2 be nodes such $Q \subset X_{t_1} \cap X_{t_2}$. Let t_3 be a node on the unique path connecting t_1 and t_2. For any vertex $v \in Q$, $v \in X_{t_3}$ as $v \in X_{t_1} \cap X_{t_2}$ and the set of nodes whose bags contain v form a connected subtree of T. So $Q \subset X_{t_3}$. □

Corollary 3.3. *Let (T, X) be a tree decomposition of the 1-skeleton K^1 of a simplicial complex K. Let $\sigma \in K$ be a simplex. There is a node $t \in I$ such that $\sigma \subset X_t$. Moreover, the set of nodes $\{t \in I \,|\, \sigma \subset X_t\}$ form a connected subtree of T.*

Proof. All subsets of σ are simplices in K. In particular, for any vertices $u, v \in \sigma$, the edge $\{u, v\} \in K$. Viewed as a set of vertices, σ forms a clique in the 1-skeleton of K. □

In light of Corollary 3.3, we define a ***tree decomposition of a simplicial complex K*** as a tree decomposition of the 1-skeleton of K. See Figure 3.1. The terms nice tree decomposition, width of a tree decomposition, and treewidth of a simplicial complex are all defined analogously.

Chapter 4: The Generic Algorithm

In this chapter, we present a generic version of the algorithm for our three problems. The following chapters discuss the specific algorithms for each problem.

Let (T, X) be a tree decomposition of a simplicial complex K. Subtrees of the tree T define subcomplexes of the simplicial complex. For a node t in the tree, we define V_t to be the union of the bags of each of t's descendants, including t itself. If K is a d-dimensional simplicial complex, we define the ***subcomplex rooted at t*** to be $K_t := K[V_t] \setminus (K[X_t])_d$. See Figure 4.1. If t' is a descendant of t, then $K_{t'} \subset K_t$. We can therefore recurse onto subcomplexes of the simplicial complex by recursing onto subtrees of the tree.

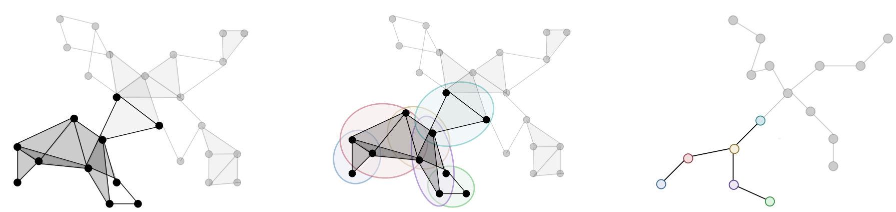

Figure 4.1: A subtree rooted at t and the corresponding subcomplex K_t. Notice that the triangle is excluded from the topmost bag X_t.

Each of our algorithms compute a set of *candidate solutions* at each node t. The exact definition of candidate solution varies between problems, but intuitively, a candidate solution at a node t is a d-chain in K_t that could be a subset of an optimal solution. In each of our algorithms, we are able to define our candidate solutions recursively: if γ is a candidate solution at t, then for each child t' of t, $\gamma \cap K_{t'}$ is a candidate solution at t'. This is the key fact our algorithm uses to find candidate solutions at t. Our algorithm attempts to build candidate solutions at t by adding d-simplices in $K_t \setminus K_{t'}$ to candidate solutions at t'.

We compute the set of candidate solutions at the nodes in a bottom-up fashion, starting at the leaves and moving towards the root. Our general approach for computing the set of candidate solutions at t is as follows. Take one candidate solution from each of t's children. Take the union of these candidate solutions. Add a subset of d-simplices

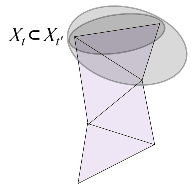

Figure 4.2: An example of a forget node t and its child t'. The triangle $\{u, v, w\}$ is contained in $K[X_{t'}]$ but not $K[X_t]$. As K_t excludes triangles in $K[X_t]$, then $\{u, v, w\} \in K_t \setminus K_{t'}$.

in K_t that are not in $K_{t'}$ for any of t's children t' to this union. Check if this union is a candidate solution at t. Repeat for all sets of candidate solutions at the children and all subset of d-simplices.

The above algorithm for computing the set of candidate solutions at a node is generally correct, but because we use a *nice* tree decomposition, we won't perform every step of this algorithm at any one node. Instead, certain steps of the algorithm are performed only at specialized nodes. For example, we only take the union of candidate solutions from multiple children at join nodes, as join nodes are the only type of node with multiple children. Also, we only add a subset of d-simplices in $K_t \setminus K_{t'}$ to the candidate solutions at forget nodes.

It might be counter-intuitive that forgetting a vertex adds d-simplices to K_t, so let's see why this is the case. Let t be a forget node and t' its unique child. The set of vertices $V_t = V_{t'}$, so $K[V_t] = K[V_{t'}]$. However, as the bags $X_{t'} \subsetneq X_t$, there may be d-simplices in $K[X_{t'}]$ that are not in $K[X_t]$. As we exclude the d-simplices in $K[X_{t'}]$ from $K_{t'}$, the d-simplices in $K[X_{t'}] \setminus K[X_t]$ will be in K_t but not $K_{t'}$. See Figure 4.2. Alternatively, for introduce and join nodes, no new d-simplices are added to K_t that do not appear in some $K_{t'}$ for a child t' of t. If t is an introduce node and t' is its unique child, we can prove $(K_t)_d = (K_{t'})_d$. While introducing a vertex v may add d-simplices to $K[V_t]$, we can prove these simplices are always contained in $K[X_t]$; see Lemma 5.3. Similarly, if t is a join node and t' and t'' are its two children, we can prove that $(K_t)_d = (K_{t'})_d \sqcup (K_{t''})_d$; see Lemma 5.12.

Chapter 5: Optimal Bounded Chain Problem

Let K be a simplicial complex and $b \in C_{d-1}(K)$ a $(d-1)$-chain. The ***Optimal Bounded Chain Problem*** asks to find the minimum weight d-chain $c \in C_d(K)$ bounded by b, if such a chain c exists. Let (T, X) a nice tree decomposition of K with root r. We will use a dynamic program on T to compute c.

At a node $t \in T$, we are interested in the portion of b contained in K_t; we define the ***partial boundary at t*** as $b_t := b \cap (K_t \setminus K[X_t])$. Our key observation is that any d-simplex that is incident to b_t is contained in K_t. This means that after the iteration in the algorithm where we process t, the portion of a candidate solution's boundary in K_t but outside of $K[X_t]$ cannot be changed by adding more d-simplices. Therefore, a candidate solution γ at t should be a d-chain in K_t that satisfies $\partial\gamma \setminus K[X_t] = b_t$. Otherwise, we place no restriction on the rest of γ's boundary, $\partial\gamma \cap K[X_t]$. There might be other simplices in $K \setminus K_t$ incident to $\partial\gamma \cap K[X_t]$, so even if $\partial\gamma \cap K[X_t] \neq b \cap K[X_t]$, this portion of $\partial\gamma$ might change later in the algorithm. We therefore let $\partial\gamma \cap K[X_t]$ be any $(d-1)$-chain of $K[X_t]$. We call this chain $\beta = \partial\gamma \cap K[X_t]$. A d-chain $\gamma \in C_d(K_t)$ is said to ***span β at t*** if $\partial\gamma \setminus K[X_t] = b_t$ and $\partial\gamma \cap K[X_t] = \beta$. See Figure 5.1.

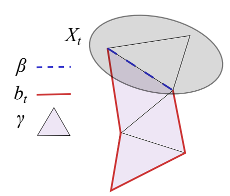

Figure 5.1: An example of a candidate solution for OBCP. The candidate solution γ is a 2-chain with boundary $\beta \cup b_t$.

For each $(d-1)$-chain β in $K[X_t]$ such that $\beta \cup b_t$ is a boundary cycle, we store the minimum weight of a d-chain that spans β at t in the dynamic programming table entry $\mathbf{V}[\beta, t]$. The bounded width of the decomposition implies a bounded number of $(d-1)$-chains in $K[X_t]$, so the table $\mathbf{V}$ is of bounded size. Lemma 5.1 tells us that the table $\mathbf{V}$ contains the weight of the optimal solution.

Lemma 5.1. *The entry* $\mathbf{V}[\emptyset, r]$ *is the weight of the minimum weight chain* c *such that* $\partial c = b$*, if such a chain* c *exists.*

Proof. Recall that the bag X_r of the root r in our tree decomposition is empty, so $K[X_r] = \emptyset$. Also, each node in the tree decomposition is a descendant of r, so $V_r = V$, $K_r = K$, and $b_r = b$. There is a unique $(d-1)$-chain $\emptyset \in C_{d-1}(K[X_r])$. The entry $\mathbf{V}[\emptyset, r]$ is the minimum weight of a d-chain $c \in C_d(K_r) = C_d(K)$ such that (1) $\partial c \cap X_r = \emptyset$ and, more importantly, (2) $\partial c \setminus K[X_r] = \partial c = b_r = b$. If no such chain c exists, then $\mathbf{V}[\emptyset, r] = \infty$. □

We are almost ready to present our algorithm for OBCP, but first, we need a lemma.

Lemma 5.2. *If* $\sigma \in K[V_t]$*, then there is a descendant* t_σ *of* t *such that* $\sigma \in K[X_{t_\sigma}]$*. In particular, if* $\sigma \in K[V_t]$*, then there is a descendant* t_σ *of* t *such that* $\sigma \in K[X_{t_\sigma}]$

Proof. Recall that $K_t = K[V_t] \setminus (K[X_t])_d$, so $K_t \subset K[V_t]$. We prove the more general statement that if $\sigma \in K[V_t]$, then there is a descendant t_σ of t such that $\sigma \in K[X_{t_\sigma}]$. Let $\sigma \in K[V_t]$. By Lemma 3.2 there is a node t_σ in T such that $\sigma \subset X_{t_\sigma}$. If t_σ is in the subtree rooted at t, we are done. So assume t_σ is not in the subtree rooted at t. For every $v \in \sigma$, there is a node t_v in the subtree rooted at t such that $v \in X_{t_v}$. As well, $v \in X_{t_\sigma}$ as $v \in \sigma \subset X_{t_\sigma}$. The nodes containing v form a connected subtree of T. So v is contained in the bag of each node on the unique path connecting t_v and t_σ. In particular, $v \in X_t$. Therefore, the bag X_t contains every $v \in \sigma$, so $\sigma \in K[X_t]$. □

We now give our dynamic program to compute $\mathbf{V}$ on a nice tree decomposition. We compute the table $\mathbf{V}$ starting at the leaves of T and moving towards the root. At each node t in our tree decomposition, we calculate the entries $\mathbf{V}[\beta, t]$ using the entries of $\mathbf{V}$ of t's children. Therefore, to specify our dynamic program, it suffices to specify how to calculate $\mathbf{V}$ at each type of node in a nice tree decomposition.

5.1 Leaf Nodes

Let t be a leaf node. Recall that $X_t = \emptyset$, so $K[X_t] = \emptyset$. Moreover, t has no children, so $V_t = K_t = b_t = \emptyset$. There is a unique $(d-1)$-chain $\beta = \emptyset \in C_{d-1}(K[X_t])$. There is also a unique d-chain $\gamma = \emptyset \in C_d(K_t)$. The chain γ spans β at t, so the unique table entry $\mathbf{V}[\emptyset, t] = 0$.

5.2 Introduce Nodes

Let t be an introduce node and t' the unique child of t. Recall that $X_t = X_{t'} \sqcup \{w\}$ for some vertex $w \in K$. Our first observation is that any simplex in the subcomplex $K[V_t]$ that contains w is only contained in the subcomplex at the bag $K[X_t]$.

Lemma 5.3. *Let $\sigma \in K[V_t]$ such that $w \in \sigma$. Then $\sigma \in K[X_t] \setminus K[V_{t'}]$.*

Proof. As $\sigma \in K[V_t]$, then by Lemma 5.2, there is a node t_σ in the subtree rooted at t such that $\sigma \in K[X_{t_\sigma}]$. Suppose for the purposes of contradiction that $t \neq t_\sigma$. Then t_σ is in the subtree rooted at t'. As $w \in \sigma$, then $w \in X_{t_\sigma}$. The set of nodes whose bags contain w form a connected subtree of T, so each node on the unique path connecting t and t_σ must contain w in its bag. The node t' is on this path, but $w \notin X_{t'}$, a contradiction. Thus, $t = t_\sigma$. As no node in the subtree rooted at t' contains σ, then by Lemma 5.2, $\sigma \notin K[V_{t'}]$. $\square$

As each simplex that contains w is contained in $K[X_t]$, then introducing w does not change the complexes K_t outside of $K[X_t]$. We prove this in the following lemma.

Lemma 5.4. *The complexes $K_{t'} \setminus K[X_{t'}] = K_t \setminus K[X_t]$.*

Proof. Observe that $K_t \setminus K[X_t] = (K[V_t] \setminus (K[X_t])_d) \setminus K[X_t] = K[V_t] \setminus K[X_t]$. We will therefore prove that $K[V_t] \setminus K[X_t] = K[V_{t'}] \setminus K[X_{t'}]$.

Let $\sigma \in K[V_{t'}] \setminus K[X_{t'}]$. It follows that $\sigma \in K[V_t]$ as $V_{t'} \subset V_t$. We need to show that $\sigma \notin K[X_t]$. Suppose $\sigma \in K[X_t]$. As $X_t \setminus X_{t'} = \{w\}$, then $w \in \sigma$. This cannot be the case, as any simplex containing w is not in $K_{t'} \subset K[V_{t'}]$ by Lemma 5.3. So $\sigma \notin K[X_t]$ and $\sigma \in K_t \setminus K[X_t]$

Now let $\sigma \in K[V_t] \setminus K[X_t]$. We conclude that $\sigma \notin K[X_{t'}]$ as $X_{t'} \subset X_t$. We need to show that $\sigma \in K[V_{t'}]$. Suppose $\sigma \notin K[V_{t'}]$. Each descendant of t is a descendant of t' except t itself. As $X_t \setminus X_{t'} = \{w\}$, then $V_t \setminus V_{t'}$ can only contain w. We conclude $w \in \sigma$. This is a contradiction, as any simplex containing w in K_t is contained in $K[X_t]$ by Lemma 5.3. Thus $\sigma \in K[V_{t'}] \setminus K[X_{t'}]$. $\square$

As introducing w does not change K_t outside of the bag X_t, then neither the chain group $C_d(K_t)$ nor the partial boundary b_t are changed by introducing w. We prove this in the following two lemmas.

Lemma 5.5. *The chain groups* $C_d(K_{t'}) = C_d(K_t)$.

Proof. We prove this by showing that $(K_{t'})_d = (K_t)_d$. This follows from Lemma 5.4 as $(K_t)_d = (K[V_t] \setminus (K[X_t])_d)_d = (K_t \setminus K[X_t])_d$. □

Lemma 5.6. *The chain* $b_t = b_{t'}$.

Proof. This follows from Lemma 5.4 as, $b_t = b \cap K_t \setminus K[X_t] = b \cap K_{t'} \setminus K[X_{t'}] = b_{t'}$. □

Introducing w to the bag X_t does not change $C_d(X_t)$ or b_t, so unsurprisingly, the values in the dynamic programming don't change either. The following lemma proves this and gives a formula for computing $\mathbf{V}[\beta, t]$.

Lemma 5.7. *Let* t *be an introduce node and let* t' *be the unique child of* t*. Let* $\beta \in C_{d-1}(K[X_t])$*. Then*

$$\mathbf{V}[\beta, t] = \begin{cases} \mathbf{V}[\beta, t'] & \textit{if } \beta \in C_{d-1}(K[X_{t'}]) \\ \infty & \textit{otherwise.} \end{cases}$$

Proof. Let $\beta \in C_{d-1}(K[X_{t'}])$. We claim that any chain $\gamma \in C_d(K_{t'})$ spanning β at t' spans β at t and vice versa. Let γ be a d-chain that spans β at t'. As $C_d(K_{t'}) = C_d(K_t)$, then γ is a d-chain in both K_t and $K_{t'}$; moreover, $\partial\gamma \subset K_t \cap K_{t'}$, so $\partial\gamma = \partial\gamma \cap K_t = \partial\gamma \cap K_{t'}$. Using this fact and Lemma 5.4, we see

$$\partial\gamma \setminus K[X_{t'}] = (\partial\gamma \cap K_{t'}) \setminus K[X_{t'}] = (\partial\gamma \cap K_t) \setminus K[X_t] = \partial\gamma \setminus K[X_t].$$

We use the fact that $\partial\gamma \cap K[X_{t'}] = \partial\gamma \cap K[X_t]$ to prove that $\partial\gamma \cap K[X_t] = b_t$. Indeed, as $\partial\gamma \setminus K[X_{t'}] = b_{t'}$ and $b_{t'} = b_t$ from Lemma 5.6, then $\partial\gamma \setminus K[X_t] = b_t$. Moreover, as $\partial\gamma = \beta \sqcup b_t$, then the rest of the $\partial\gamma$ is $\partial\gamma \cap K[X_t] = \beta$. This completes the proof that γ spans β at t. By the same argument, any chain spanning β at t spans β at t'. As the same set of chains span β at t and t', then $\mathbf{V}[\beta, t] = \mathbf{V}[\beta, t']$

Now let $\beta \in C_{d-1}(K[X_t]) \setminus C_{d-1}(K[X_{t'}])$. As $\beta \notin C_{d-1}(K[X_{t'}])$, there must be a simplex $\sigma \in \beta$ such that $w \in \sigma$. As any simplex σ such that $w \in \sigma$ is not contained in $K[X_{t'}]$, there is no chain $\gamma \in C_d(K_t)$ that spans β at t as $C_d(K_t) = C_d(K_{t'})$. Thus, $\mathbf{V}[\beta, t] = \infty$. □

5.3 Forget Nodes

Let t be a forget node and t' the unique child of t. Recall that $X_t \sqcup \{w\} = X_{t'}$. Forget nodes add new chains to the chain group $C_d(K_t)$. In particular, any d-simplex $\sigma \in (K[X_{t'}])_d$ that contains w is not contained in $K_{t'}$ but will be contained in K_t. We prove this in our first lemma. Let $(K[X_{t'}])_d^w = \{\sigma \in (K[X_{t'}])_d \,|\, w \in \sigma\}$ and let $C_d^w(K[X_{t'}])$ be the free abelian group on $(K[X_{t'}])_d^w$ with coefficients in $\mathbb{Z}_2$.

Lemma 5.8. *The chain group $C_d(K_t) = C_d(K_{t'}) \oplus C_d^w(K[X_{t'}])$.*

Proof. As $C_d(K_t)$ is generated by $(K_t)_d$, we can prove the lemma by showing that $(K_t)_d = (K_{t'})_d \sqcup (K[X_{t'}])_d^w$.

We first prove that $(K_{t'})_d \subset (K_t)_d$. Let $\sigma \in (K_{t'})_d$. Then $\sigma \in K[V_{t'}]$ and $\sigma \notin K[X_{t'}]$. We have that $\sigma \in K[V_t] = K[V_{t'}]$ as each descendant of t' is a descendant of t and $X_t \subset X_{t'}$. Furthermore, $\sigma \notin K[X_t]$ as $K[X_t] \subset K[X_{t'}]$ and $\sigma \notin K[X_{t'}]$. So $\sigma \in (K_t)_d$.

We next prove that $(K_t)_d \setminus (K_{t'})_d = (K[X_{t'}])_d^w$. Let $\sigma \in (K_t)_d \setminus (K_{t'})_d$. As $\sigma \in (K_t)_d$, then $\sigma \in K[V_t]$ and $\sigma \notin K[X_t]$. As $\sigma \notin (K_t)_d$, then either $\sigma \notin K[V_{t'}]$ or $\sigma \in K[X_{t'}]$. We know that $\sigma \in K[V_{t'}] = K[V_t]$, so $\sigma \in K[X_{t'}]$. It follows that $w \in \sigma$ as $X_{t'} \setminus X_t = \{w\}$, so $\sigma \in (K[X_{t'}])_d^w$.

Now let $\sigma \in (K[X_{t'}])_d^w$. We have that $\sigma \in K[X_{t'}] \subset K[V_t]$. As $w \in \sigma$, we also have that $\sigma \notin K[X_t]$, so $\sigma \in (K_t)_d$. Lastly, as $\sigma \in (K[X_{t'}])_d$, then $\sigma \notin (K_{t'})_d$ and $\sigma \in (K_t)_d \setminus (K_{t'})_d$. ☐

Intuitively, we can build the chain γ that attains $\mathbf{V}[\beta, t]$ by composing a minimal chain $\gamma' \in C_d(K[X_{t'}])$ that attains $V[\beta', t']$ and a chain $\gamma_w \in C_d^w(K[X_{t'}])$ of d-simplices that all contain w. We need to enforce that $\partial(\gamma' + \gamma_w) = \beta + b_t$. The chain γ' includes $b_{t'}$ in the portion of its boundary outside $K[X_{t'}]$. We want to choose γ_w to help cover the rest of the boundary $b_t \setminus b_{t'}$. Indeed, we can prove that the rest of the boundary $b_t \setminus b_{t'}$ is contained in $K[X_{t'}]$, i.e. $b_t \setminus b_{t'} = b_t \cap K[X_{t'}]$. So if $\partial(\gamma' + \gamma_w) \cap K[X_{t'}] \setminus K[X_t] = (\beta' + \partial\gamma_w) \setminus K[X_t] = b_t \cap K[X_{t'}]$, we will have covered all of b_t. We prove a lemma in this direction.

Lemma 5.9. *The chain $b_{t'} = b_t \setminus K[X_{t'}]$.*

Proof. The chains b_t and $b_{t'}$ are defined $b_{t'} = (b \cap K_{t'}) \setminus K[X_{t'}]$ and $b_t = (b \cap K_t) \setminus K[X_t]$.

If we consider the difference $b_t \setminus K[X_{t'}]$, we find that

$$b_t \setminus K[X_{t'}] = ((b \cap K_t) \setminus K[X_t]) \setminus K[X_{t'}] = (b \cap K_t) \setminus K[X_{t'}]$$

as $K[X_t] \subset K[X_{t'}]$. We will use this later.

The sets of vertices in the trees rooted at t and t' are equal, i.e. $V_t = V_{t'}$, but the vertices in the bags $X_t \subset X_{t'}$. We use these two facts to show that the complexes K_t and $K_{t'}$ only differ in the complexes induced by their bags, namely $K_t \setminus K[X_{t'}] = K_{t'} \setminus K[X_{t'}]$:

$$K_t \setminus K[X_{t'}] = K[V_t] \setminus (K[X_t])_d \setminus K[X_{t'}] = K[V_t] \setminus K[X_{t'}] = K[V_{t'}] \setminus (K[X_{t'}])_d \setminus K[X_{t'}] = K_{t'} \setminus K[X_{t'}].$$

We have proved that $b_t = b \cap (K_t \setminus K[X_{t'}])$. We have also proved that $K_t \setminus K[X_{t'}] = K_{t'} \setminus K[X_{t'}]$. We use these facts to prove that $b_t \setminus K[X_{t'}] = b_{t'}$ as

$$b_t \setminus K[X_{t'}] = b \cap K_t \setminus K[X_{t'}] = b \cap K_{t'} \setminus K[X_{t'}] = b_{t'}. \qquad \square$$

This lemma implies that $b_t \setminus b_{t'} = b_t \cap K[X_{t'}]$, as claimed above. We are now ready to give a formula for $\mathbf{V}[\beta, t]$.

Lemma 5.10. *Let $\beta \in C_{d-1}(K[X_t])$. Then*

$$\mathbf{V}[\beta, t] = \min_{\substack{\beta' \in C_{d-1}(K[X_{t'}]), \gamma_w \in C_d^w(K[X_{t'}]): \\ (\beta' + \partial\gamma_w) \cap K[X_t] = \beta \,\&\, (\beta' + \partial\gamma_w) \setminus K[X_t] = b_t \cap K[X_{t'}]}} \mathbf{V}[\beta', t'] + ||\gamma_w||.$$

Proof. First, we show that each d-chain on the right hand side of the equation in the lemma spans β at t. Let $\gamma' \in C_d(K_{t'})$ that attains $\mathbf{V}[\beta', t']$, and let $\gamma_w \in C_d^w(K[X_{t'}])$ such that $(\beta' + \partial\gamma_w) \cap K[X_t] = \beta$ and $(\beta + \partial\gamma_w) \setminus K[X_t] = b_t \cap K[X_{t'}]$ Let $\gamma = \gamma' + \gamma_w$; we show that γ spans β at t. The boundary

$$\partial\gamma = \partial\gamma' + \partial\gamma_w = b_{t'} + \beta' + \partial\gamma_w.$$

If γ were to span β at t, then $\partial\gamma \cap K[X_t] = \beta$ and $\partial\gamma \setminus K[X_t] = b_t$. The intersection of $\partial\gamma \cap K[X_t]$ is

$$\partial\gamma \cap K[X_t] = (b_{t'} + \beta' + \partial\gamma_w) \cap K[X_t] = (\beta' + \partial\gamma_w) \cap K[X_t] = \beta$$

as $b_{t'} \cap K[X_{t'}] = \emptyset$. The difference $\partial\gamma \setminus K[X_t]$ is

$$\partial\gamma \setminus K[X_t] = (b_{t'} + \beta' + \partial\gamma_w) \setminus K[X_t] = b_{t'} + (\beta' + \partial\gamma_w) \setminus K[X_t] = b_{t'} + b_t \cap K[X_{t'}]$$

as $b_{t'} \setminus K[X_t] = b_{t'}$. Lemma 5.9 tells us that $b_{t'} = b_t \setminus K[X_{t'}]$, so

$$\partial\gamma \setminus K[X_t] = b_{t'} + b_t \cap K[X_{t'}] = b_t \setminus K[X_{t'}] + b_t \cap K[X_{t'}] = b_t$$

and γ indeed spans β at t. Moreover, as γ' and γ_w are disjoint, then $||\gamma|| = ||\gamma'|| + ||\gamma_w||$. As γ' achieves $\mathbf{V}[\beta', t']$, then $||\gamma|| = \mathbf{V}[\beta', t'] + ||\gamma_w$.

Next, we verify that the chain $\gamma \in C_d(K_t)$ that achieves $\mathbf{V}[\beta, t]$ is included in the right hand side of Equation (5.10). By Lemma 5.8, $\gamma = \gamma' + \gamma_w$ where $\gamma' \in C_d(K_{t'})$ and $\gamma_w \in C_d^w(K[X_{t'}])$. Let $\beta' = \partial\gamma' \cap K[X_{t'}]$. We need to verify that γ' spans β' at t' and achieves $\mathbf{V}[\beta', t']$, that $(\beta' + \partial\gamma_w) \cap K[X_t] = \beta$, and that $(\beta' + \partial\gamma_w) \setminus K[X_t] = b_t \cap K[X_{t'}]$. We first prove that γ' spans β' at t'. We use the alternative expressions for $\partial\gamma$ as $\partial\gamma = \partial\gamma' + \partial\gamma_w$ and $\partial\gamma = \beta + b_t$. Indeed,

$$\partial\gamma \setminus K[X_{t'}] = (\partial\gamma' + \partial\gamma_w) \setminus K[X_{t'}] = \partial\gamma' \setminus K[X_{t'}]$$

as $\partial\gamma_w \subset K[X_{t'}]$ and

$$\partial\gamma \setminus K[X_{t'}] = (\beta + b_t) \setminus K[X_{t'}] = b_t \setminus K[X_{t'}] = b_{t'}$$

as $\beta \subset K[X_{t'}]$ and as $b_t \setminus K[X_{t'}] = b_{t'}$ by Lemma 5.9. This proves that γ' spans β' at t'. We delay proving that γ' achieves $\mathbf{V}[\beta', t']$ until the end of the proof.

We now prove that $(\beta' + \partial\gamma_w) \cap K[X_t] = \beta$. As γ spans β at t, then $\partial\gamma \cap K[X_t] = \beta$ and $\partial\gamma \setminus K[X_t] = b_t$. We see that

$$\beta = \partial\gamma \cap K[X_t] = (\partial\gamma + \partial\gamma_w) \cap K[X_t] = (b_{t'} + \beta' + \partial\gamma_w) \cap K[X_t] = (\beta' + \partial\gamma_w) \cap K[X_t]$$

as $\beta_{t'} \cap K[X_t] = \emptyset$. Finally, we prove that $(\beta' + \partial\gamma_w) \setminus K[X_t] = b_t \cap K[X_{t'}]$. We consider the intersection $\partial\gamma \setminus K[X_t] \cap K[X_{t'}]$. Using the alternate expressions for $\partial\gamma$, we find that

$$\partial\gamma \setminus K[X_t] \cap K[X_{t'}] = b_t \cap K[X_{t'}]$$

and

$$\partial\gamma\setminus K[X_t]\cap K[X_{t'}] = (\partial\gamma'+\partial\gamma_w)\setminus K[X_t]\cap K[X_{t'}] = (b_{t'}+\beta'+\partial\gamma_w)\setminus K[X_t]\cap K[X_{t'}] = (\beta'+\partial\gamma_w)\setminus K[X_t]$$

as $b_{t'} \cap K[X_{t'}] = \emptyset$. We see that $b_t \cap K[X_{t'}] = (\beta' + \partial\gamma_w) \setminus K[X_t]$, as desired.

We are now ready to prove that γ' achieves $\mathbf{V}[\beta', t']$. Suppose not, and let $\gamma'_o \in C_d(K_{t'})$ be a chain that achieves $V[\beta', t']$. We showed in the previous paragraph that $(\beta'+\partial\gamma_w)\cap K[X_t] = \beta$ and $(\beta'+\partial\gamma_w)\setminus K[X_t] = b_t \cap K[X_{t'}]$. This implies that $\gamma'_o+\gamma_w$ spans β at t by the first paragraph of this proof. Moreover, $||\gamma'_o+\gamma_w|| < ||\gamma'+\gamma_w|| = ||\gamma||$, which contradicts the assumed optimality of γ. Thus, γ' achieves $V[\beta', t']$ and γ is included in the right hand side of Equation 5.10. $\square$

5.4 Join Nodes

Let t be a join node, and let t' and t'' be the two children of t. Recall that $X_t = X_{t'} = X_{t''}$. We first observe that $K_{t'}$ and $K_{t''}$ only overlap in $K[X_t]$.

Lemma 5.11. $K_t \setminus K[X_t] = (K_{t'} \setminus K[X'_t]) \sqcup (K_{t''} \setminus K[X_{t''}])$.

Proof. As $K[X_t] = K[X_{t'}] = K[X_{t''}]$, we see immediately that both $K_{t'} \setminus K[X_{t'}] \subset K_t \setminus K[X_t]$ and $K_{t''} \setminus K[X_{t''}] \subset K_t \setminus K[X_t]$ as both $K_{t'} \subset K_t$ and $K_{t''} \subset K_t$.

We now prove that $K_{t'} \setminus K[X_{t'}]$ and $K_{t''} \setminus K[X_{t''}]$ are disjoint. Suppose there is a simplex $\sigma \in (K_{t'} \setminus K[X_{t'}]) \cap (K_{t''} \setminus K[X_{t''}])$. Then by Lemma 5.2, there are nodes t'_σ and t''_σ in the subtrees rooted at t' and t'' such that $\sigma \in K[X_{t'_\sigma}]$ and $\sigma \in K[X_{t''_\sigma}]$. The set of nodes containing σ form a connected subtree. This is a contradiction, as t lies on the unique path connecting t'_σ and t''_σ and $\sigma \notin K[X_t]$ by assumption. Hence $K_{t'} \setminus K[X_{t''}] \cap K_{t''} \setminus K[X_{t''}] = \emptyset$.

We now prove that $K_t \setminus K[X_t] \subset (K_{t'} \setminus K[X_{t'}]) \sqcup (K_{t''} \setminus K[X_{t''}])$. Let $\sigma \in K_t \setminus K[X_t]$. By Lemma 5.2, there is a node t_σ in the subtree rooted at t such that $\sigma \in K[X_{t_\sigma}]$. We know $t_\sigma \neq t$ as $\sigma \notin K[X_t]$, so t_σ is either in the subtree rooted at t' or the subtree rooted at t''. $\square$

Lemma 5.11 implies that any d-chain in K_t is the sum of a d-chain in $K_{t'}$ and a d-chain in $K_{t''}$. We prove this in the following lemma.

Lemma 5.12. *The chain group* $C_d(K_t) = C_d(K_{t'}) \oplus C_d(K_{t''})$.

Proof. As $C_d(K_t)$ is generated by $(K_t)_d$, we can prove this by showing that $(K_t)_d = (K_{t'})_d \sqcup (K_{t''})_d$. We first note that

$$(K_t)_d = (K[V_t] \setminus (K[X_t])_d)_d = (K[V_t] \setminus K[X_t])_d$$

so it follows by Lemma 5.11 that

$$(K_t)_d = (K[V_t] \setminus K[X_t])_d = (K[V_{t'}] \setminus K[X_{t'}])_d \sqcup (K[V_{t''}] \setminus K[X_{t''}])_d = (K_{t'})_d \sqcup (K_{t''})_d. \qquad \square$$

Similarly, the boundary b_t is composed of a portion in $K_{t'}$, namely $b_{t'}$, and a portion in $K_{t''}$, namely $b_{t''}$.

Corollary 5.13. *The chain* $b_t = b_{t'} \sqcup b_{t''}$.

Proof. The boundary $b_t = b \cap K_t \setminus K[X_t]$. Using Lemma 5.11,

$$b_t = b \cap K_t \setminus K[X_t] = (b \cap K_{t'} \setminus K[X_t]) \sqcup (b \cap K_{t''} \setminus K[X_t]) = b_{t'} \sqcup b_{t''}. \qquad \square$$

These two lemmas tell us that a chain γ that spans β at t is the sum of chains γ' and γ''. These chains satisfy three conditions: $\partial\gamma' \setminus K[X_{t'}] = b_{t'}$, $\partial\gamma'' \setminus K[X_{t''}] = b_{t''}$, and $(\partial\gamma' + \partial\gamma'') \cap K[X_t] = \beta$. To find the *minimal* chain that spans β at t, we only need to consider minimal chains γ' and γ'' with properties. The following formula for $\mathbf{V}[\beta, t]$ confirms this.

Lemma 5.14. *Let* $\beta \in C_{d-1}(K[X_t])$. *Then*

$$\mathbf{V}[\beta, t] = \min_{\substack{\beta' \in C_{d-1}(K[X_{t'}]), \beta'' \in C_{d-1}(K[X_{t''}]) \\ \beta' + \beta'' = \beta}} \mathbf{V}[\beta', t'] + \mathbf{V}[\beta'', t'']$$

Proof. We first show that each chain on the right hand side of Equation (5.14) spans β at t. Let $\gamma' \in C_d(K_{t'})$ and $\gamma'' \in C_d(K_{t''})$ such that γ' spans β' at t', γ'' spans β'' at t'', and $\beta' + \beta'' = \beta$. We claim that $\gamma = \gamma' + \gamma''$ spans β at t. We have

$$\partial\gamma \cap K[X_t] = (\partial\gamma' \cap K[X_t]) + (\partial\gamma'' \cap K[X_t]) = \beta' + \beta'' = \beta.$$

and

$$\partial\gamma \setminus K[X_t] = (\partial\gamma' \setminus K[X_t]) + (\partial\gamma'' \setminus K[X_t]) = b_{t'} + b_{t''} = b_t$$

as $b_{t'} + b_{t''} = b_t$ by Corollary 5.13. So γ indeed spans β at t. Moreover, $||\gamma|| = ||\gamma'|| + ||\gamma''||$ as γ' and γ'' are disjoint. If γ' and γ'' achieve $\mathbf{V}[\beta', t']$ and $\mathbf{V}[\beta'', t'']$, then $||\gamma|| = ||\gamma'|| + ||\gamma''|| = \mathbf{V}[\beta', t] + \mathbf{V}[\beta'', t'']$.

Now let $\gamma \in C_d(K_t)$ be the minimum weight chain that spans β at t. By Lemma 5.12, $\gamma = \gamma' + \gamma''$ where $\gamma' \in C_d(K_{t'})$ and $\gamma'' \in C_d(K_{t''})$. Let $\beta' = \partial\gamma' \cap K[X_{t'}]$ and $\beta'' = \partial\gamma'' \cap K[X_{t''}]$. We will show that γ' and γ'' achieve $\mathbf{V}[\beta', t']$ and $\mathbf{V}[\beta'', t'']$ respectively. We first show that γ' spans β' at t'. We consider the intersection $(\partial\gamma \setminus K[X_t]) \cap K_{t'}$. We see that

$$(\partial\gamma \setminus K[X_t]) \cap K_{t'} = (\partial\gamma' \setminus K[X_t]) \cap K_{t'} + (\partial\gamma'' \setminus K[X_t]) \cap K_{t'}.$$

As $\partial\gamma'' \setminus K[X_t] \subset K_{t''} \setminus K[X_t]$, then $(\partial\gamma'' \setminus K[X_t]) \cap K_{t'} = \emptyset$ as $K'_t \setminus K[X_{t'}]$ and $K_{t''} \setminus K[X_{t''}]$ are disjoint by Lemma 5.11. So

$$(\partial\gamma \setminus K[X_t]) \cap K_{t'} = (\partial\gamma' \setminus K[X_t]) \cap K_{t'} = \partial\gamma' \setminus K[X_t]$$

as $\partial\gamma' \subset K_{t'}$. We can alternatively express $(\partial\gamma \setminus K[X_t]) \cap K_{t'}$ as

$$(\partial\gamma \setminus K[X_t]) \cap K_{t'} = b_t \cap K_{t'} = b_{t'} \cap K_{t'} + b_{t''} \cap K_{t''}.$$

using Corollary 5.13. As $b_{t''} \subset K_{t''} \setminus K[X_t]$, then $b_{t''} \cap K_{t'} = \emptyset$. So

$$(\partial\gamma \setminus K[X_t]) \cap K_{t'} = b_{t'} \cap K_{t'} = b_{t'}.$$

Thus $\partial\gamma' \setminus K[X_t] = (\partial\gamma \setminus K[X_t]) \cap K_{t'} = b_{t'}$ and γ' spans β' at t'. The proof that γ'' spans β'' at t'' is symmetrical.

As γ is the minimum weight chain that spans β at t, then γ' and γ'' must be the chains that achieve $\mathbf{V}[\beta', t']$ and $\mathbf{V}[\beta'', t'']$ respectively. We saw in the first paragraph that any two chains γ' and γ'' that span β' at t' and β'' at t'' sum to span β at t. So if (say) γ' did not achieve $\mathbf{V}[\beta', t']$, we could replace γ' with the chain γ'_o that achieves $\mathbf{V}[\beta', t']$ and $\gamma'_o + \gamma''$ would be a chain with strictly lower weight that spans β at t, contradicting the assumed optimality of γ. $\square$

5.5 Analysis

We now analyze the running time of the dynamic program, thereby proving Theorem 1.1.

Proof of Theorem 1.1. We first show that we can process each node in T in $2^{O\left(k^d\right)}$.

We can compute the entry at the leaf node in constant time.

We can compute the value $V[\beta, t]$ of each $(d-1)$-chain β at an introduce node in constant time. As there are $O(\binom{k+1}{d}) = O\left((k+1)^d\right)$ $(d-1)$-simplices in $K[X_t]$, then there are at most $2^{O\left(k^d\right)}$ $(d-1)$-chains in $C_{d-1}(K[X_t])$. Processing an introduce node thus takes $2^{O\left(k^d\right)}$ time in total.

We compute the entry at a forget node by performing nested iterations over $C_{d-1}(K[X_t])$ and $C_d^w(K[X_t])$. There are $O(\binom{k+1}{d}) = O\left((k+1)^d\right)$ $(d-1)$-simplices in $K[X_t]$, so there are $2^{O\left(k^d\right)}$ $(d-1)$-chains $\beta \in C_{d-1}(K[X_t])$. There are likewise $O(\binom{k+1}{d}) = O(k^d)$ d-simplices (we choose d vertices in addition to w) in $(K[X_{t'}])_d^w$ and $2^{O(k^d)}$ chains $\gamma_w \in C_d^w(K[X_{t'}])$. The size of both of the chain groups $C_{d-1}(K[X_{t'}])$ and $C_d^w(K[X_{t'}])$ is at most $2^{O\left(k^d\right)}$, so the nested iteration over both of these groups takes $2^{O\left(k^d\right)}$ time.

We compute the entry at a join node by iterating over the chain groups $C_{d-1}(K[X_{t'}])$ and $C_{d-1}(K[X_{t''}])$, which takes $2^{O\left(k^d\right)}$ time.

There are $O(kn)$ nodes in a nice tree decomposition. The dynamic programming table entry at any can be computed in $2^{O\left(k^d\right)}$ time. The algorithm takes $2^{O\left(k^d\right)}n$ time in total. Moreover, as each simplex in K is the subset of some bag in the tree decomposition, then $d \leq k$. The running time of our algorithm is therefore $2^{O\left(k^k\right)}n$.

$\square$

As a corollary, we can test whether two $d-1$ chains b and h are homologous by running our algorithm on $b + h$. The chains b and h are homologous if and only if $\mathbf{V}[\emptyset, r] < \infty$. Hence, we obtain Corollary 1.2. This implies we can use our algorithm to test whether or not a $(d-1)$-cycle b is null-homologous, which is to say, whether b is homologous to the empty chain.

Our algorithm uses the Hamming norm to measure the weight of a chain, but our algorithm can be easily adapted to solve OBCP in a simplicial complex with weight d-simplices. If $w : K_d \to \mathbb{R}^+$ is a weight function on the d-simplices, we define a weight function on d-chains $w : C_d(K) \to \mathbb{R}^+$ such that $w(\gamma) = \sum_{\sigma \in \gamma} w(\sigma)$ is the weight of the chain γ. If we substitute $w(\gamma)$ for $||\gamma||$ everywhere in our algorithm for OBCP, it is easy to see this adapted algorithm finds the minimum weight chain c bounded by b.

Chapter 6: Optimal Homologous Chain Problem

Let K be a simplicial complex and let $b \in C_{d-1}(K)$. The ***Optimal Homologous Chain Problem*** asks to find the minimum weight $(d{-}1)$-chain $h \in C_{d-1}(K)$ such that $b + h = \partial c$ for some $c \in C_d(K)$. Let (T, X) be a nice tree decomposition of K. We will find h using a dynamic program on T.

Any chain $c \in C_d(K)$ defines a chain homologous to b, namely $h = \partial c + b$. Instead of searching for $(d{-}1)$-chain h, we can therefore search for the d-chain c that minimizes the weight $||\partial c + b||$. The chain c is also defines the minimum chain homologous to b in a local sense. If we restrict this chain to a subcomplex K_t, we find that $c \cap K_t$ is the chain minimizes $||\partial(c \cap K_t) \setminus +b_t||$ over all chains with boundary $\partial(c \cap K_t) \cap K[X_t] = \beta$. Accordingly, for each $(d{-}1)$-chain $\beta \in C_{d-1}(K[X_t])$, we store the dynamic programming table entry

$$\mathbf{V}[\beta, t] = \min_{\substack{\gamma \in C_d(K_t): \\ \partial\gamma \cap K[X_t] = \beta}} ||\partial\gamma \setminus K[X_t] + b_t||.$$

A d-chain γ **spans $\boldsymbol{\beta}$ at** t if $\partial\gamma \cap K[X_t] = \beta$. Lemma 6.1 shows that the table $\mathbf{V}$ contains the weight of the optimal solution.

Lemma 6.1. *The entry* $\mathbf{V}[\emptyset, r]$ *is the minimum weight* $||\partial c + b||$ *for any chain* $c \in C_d(K)$.

Proof. The unique entry at the root $\mathbf{V}[\emptyset, r]$ contains the weight of the minimum weight $(d{-}1)$-chain homologous to b. The bag at the root X_t is empty, so $K[X_r] = \emptyset$. Moreover, $V_r = V$, so $K_r = K$ and $b_r = b$. Thus, $\mathbf{V}[\emptyset, r]$ contains the minimum weight $||\partial c \setminus K[X_r] + b|| = ||\partial c + b||$ for any chain $c \in C_d(K_r) = C_d(K)$. $\square$

Our algorithm for OHCP is similar to our algorithm for OBCP as we are searching for a d-chain c, except we are optimizing a different function. We perform a dynamic program on our tree decomposition, and at each node t, we store the weight of the chain $\gamma \in C_d(K_t)$ that spans each $(d{-}1)$-chain $\beta \in C_{d-1}(K[X_t])$ and minimizes the objective function. We construct the chain γ using the orthogonal decompositions of the chain groups $C_d(K_t)$ from Chapter 5. At each node, the two algorithms will loop over the same chain groups. Accordingly, the running time for our algorithm for OHCP is the same as the running time for our algorithm for OBCP, and we obtain Theorem 1.3.

We now present our dynamic program for OHCP. We compute the entries of $\mathbf{V}$ at a node t using the entries of $\mathbf{V}$ at the children of t. Accordingly, we only need to specify how to compute t at each type of node in a nice tree decomposition.

6.1 Leaf Nodes

Let t be a leaf node. Recall that $X_t = \emptyset$, so $K[X_t] = \emptyset$. Moreover, t has no children, so $V_t = K_t = b_t = \emptyset$. There is a unique $(d-1)$-chain $\beta = \emptyset \in C_{d-1}(K[X_t])$ and a unique d-chain $\gamma = \emptyset \in C_d(K_t)$. The chain γ spans β at t, so $\mathbf{V}[\emptyset, t] = 0$.

6.2 Introduce Nodes

Let t be an introduce node and t' the unique child of t. Recall that $X_t = X_{t'} \sqcup \{w\}$. Let $\beta \in C_{d-1}(K[X_t])$. We claim the following formula for $\mathbf{V}[\beta, t]$.

Lemma 6.2. *Let t be an introduce node and t' be its unique child. Let $\beta \in C_{d-1}(K[X_t])$. Then*

$$\mathbf{V}[\beta, t] = \begin{cases} \mathbf{V}[\beta, t'] & \text{if } \beta \in C_{d-1}(X_{t'}) \\ \infty & \text{otherwise} \end{cases}$$

Proof. We will show that any chain that spans β at t' spans β at t' and vice versa. Let $\gamma \in C_d(K_{t'})$. We can decompose $\partial\gamma$ into $\partial\gamma \cap K[X_{t'}]$ and $\partial\gamma \setminus K[X_{t'}]$ and into $\partial\gamma \cap K[X_t]$ and $\partial\gamma \setminus K[X_t]$. As $\partial\gamma \in K_t \cap K_{t'}$, then

$$\partial\gamma \setminus K[X_{t'}] = \partial\gamma \cap (K_{t'} \setminus K[X_{t'}]) = \partial\gamma \cap (K_t \setminus K[X_t]) = \partial\gamma \setminus K[X_t].$$

by Lemma 5.4. As a corollary, we see that $\partial\gamma \cap K[X_t] = \partial\gamma \cap K[X_{t'}]$. So if γ spans β at t, then γ spans β at t' and vice versa. Moreover, as $b_t = b_{t'}$, then $||\partial\gamma \cap K[X_t] + b_t|| = ||\partial\gamma \cap K[X_{t'}] + b_{t'}||$. So $\mathbf{V}[\beta, t] = \mathbf{V}[\beta, t]$ for each $(d-1)$-chain β in $C_{d-1}(K_{t'})$.

As $\partial\gamma \cap K[X_t] = \partial\gamma \cap K[X_{t'}]$ for each $\gamma \in C_d(K_t)$, then if $\beta \in C_d(K[X_t]) \setminus C_d(K[X_{t'}])$, no chain spans β at t. Accordingly, we set $\mathbf{V}[\beta, t] = \infty$. $\square$

6.3 Forget Nodes

Let t be a forget node and t' the unique child of t. Recall that $X_t \sqcup \{w\} = X_{t'}$. Let $\gamma \in C_d(K_t)$. We can decompose $\gamma = \gamma' + \gamma_w$ where $\gamma' \in C_d(K_{t'})$ and $\gamma_w \in C_d^w(K[X_t])$. Let $\beta' = \partial\gamma' \cap K[X_{t'}]$. We can decompose the sum $\gamma \setminus K[X_t] + b_t$ into $(\gamma \setminus K[X_t] + b_t) \cap K[X_t]$ and $(\gamma \setminus K[X_t] + b_t) \setminus K[X_{t'}]$. We find that $(\gamma \setminus K[X_t] + b_t) \cap K[X_t] = ((\beta' + \partial\gamma') \setminus K[X_{t'}]) + b_t \cap K[X_{t'}]$ and $(\gamma \setminus K[X_t] + b_t) \setminus K[X_{t'}] = (\partial\gamma' + b_{t'})$. This justifies the following formula for $\mathbf{V}[\beta, t]$.

Lemma 6.3. *Let t be a forget node and t' the unique child of t. Let $\beta \in C_{d-1}(K[X_t])$. Then*

$$V[\beta, t] = \min_{\substack{\beta' \in C_{d-1}(K[X_{t'}]), \gamma_w \in C_d^w(K[X_{t'}]): \\ (\beta' + \partial\gamma_w) \cap K[X_t] = \beta}} V[\beta', t'] + ||(\beta' + \partial\gamma_w) \setminus K[X_t] + b_t \cap K[X_{t'}]||$$

Proof. We first show that any chain on the right hand side of Equation 6.3 spans β at t. Let $\gamma' \in C_d(K_{t'})$ that achieves $\mathbf{V}[\beta', t]$ and let $\gamma_w \in C_d^w(K[X_{t'}])$ such that $(\beta' + \partial\gamma_w) \cap K[X_t] = \beta$. Let $\gamma = \gamma' + \gamma_w$. Observe that if we intersect $\partial\gamma \cap K[X_t] \cap K[X_{t'}] = \partial\gamma \cap K[X_t]$ as $K[X_t] \subset K[X_{t'}]$. We find that

$$\partial\gamma \cap K[X_t] = \partial\gamma \cap K[X_{t'}] \cap K[X_t] = (\partial\gamma' \cap K[X_{t'}] + \partial\gamma_w \cap K[X_{t'}]) \cap K[X_t] = (\beta' + \partial\gamma_w) \cap K[X_t] = \beta,$$

as $\gamma_w \subset K[X_{t'}]$, so γ indeed spans β at t.

Next, we show that $||\partial\gamma \setminus K[X_t] + b_t|| = \mathbf{V}[\beta, t] + ||(\beta' + \partial\gamma_w) \setminus K[X_t] + b_t \cap K[X_{t'}]||$. We can show this by showing that $\partial\gamma \setminus K[X_t] + b_t = (\partial\gamma' \setminus K[X_{t'}] + b_t) \sqcup ((\beta' + \partial\gamma_w) \setminus K[X_t] + b_t \cap K[X_{t'}])$. Again, we decompose $\partial\gamma \setminus K[X_t] + b_t$ as $(\partial\gamma \setminus K[X_t] + b_t) \cap K[X_{t'}]$ and $(\partial\gamma \setminus K[X_t] + b_t) \setminus K[X_{t'}]$. If we intersect $\partial\gamma \setminus K[X_t] + b_t$ with $K[X_{t'}]$, we find that

$$(\partial\gamma \setminus K[X_t] + b_t) \cap K[X_{t'}] = (\partial\gamma \setminus K[X_t] \cap K[X_{t'}]) + (b_t \cap K[X_{t'}]) = (\beta' + \partial\gamma_w) \setminus K[X_t] + b_t \cap K[X_{t'}]$$

as $\partial\gamma \cap K[X_{t'}] = (\partial\gamma' + \partial\gamma_w) \cap K[X_{t'}] = \beta' + \partial\gamma_w$. If we subtract $K[X_{t'}]$ from $\partial\gamma \setminus K[X_{t'}] + b_t$, we find

$$(\partial\gamma \setminus K[X_t] + b_t) \setminus K[X_{t'}] = \partial\gamma \setminus K[X_{t'}] + b_t \setminus K[X_{t'}] = \partial\gamma' \setminus K[X_{t'}] + b_{t'} = \mathbf{V}[\beta', t']$$

as $\partial\gamma \setminus K[X_{t'}] = (\partial\gamma' + \partial\gamma_w) \cap K[X_{t'}] = \partial\gamma'$.

We now show that the chain $\gamma \in C_d(K_t)$ that achieves $\mathbf{V}[\beta, t]$ is contained in the right hand side of Equation (6.3). Let $\gamma \in C_d(K_t)$ be any chain such that $\partial\gamma \cap K[X_t] = \beta$. By Lemma 5.8, we can decompose $\gamma = \gamma' + \gamma_w$ where $\gamma' \in C_d(K_{t'})$ and $\gamma_w \in C_d^w(K[X_{t'}])$. Let $\beta' = \partial\gamma' \cap K[X_{t'}]$. We first show that $(\beta' + \partial\gamma_w) \cap K[X_t] = \beta$. Observe that intersecting $\partial\gamma \cap K[X_t] \cap K[X_{t'}] = \partial\gamma \cap K[X_t]$ as $K[X_t] \subset K[X_{t'}]$. Thus,

$$\beta = \partial\gamma \cap K[X_t] = \partial\gamma \cap K[X_{t'}] \cap K[X_t] = ((\partial\gamma' \cap K[X_{t'}]) + (\partial\gamma_w \cap K[X_{t'}])) \cap K[X_t] = (\beta' + \partial\gamma_w) \cap K[X_t]$$

as $\partial\gamma_w \subset K[X_{t'}]$. Next, we prove that $\partial\gamma \setminus K[X_t] + b_t = (\partial\gamma' + b_{t'}) \sqcup ((\beta' + \partial\gamma_w) \setminus K[X_t] + (b_t \cap K[X_{t'}]))$. This will imply that the weight $||\partial\gamma \setminus K[X_t] + b_t|| = ||(\partial\gamma' + b_{t'})|| + ||(\beta' + \partial\gamma_w) \setminus K[X_t] + (b_t \cap K[X_{t'}]||$. From here, it is easy to see that γ' must be the chain that achieves $\mathbf{V}[\beta, t]$.

We consider the decomposition of $\partial\gamma \setminus K[X_t] + b_t$ into $(\partial\gamma \setminus K[X_t] + b_t) \cap K[X_{t'}]$ and $(\partial\gamma \setminus +b_t) \setminus K[X_{t'}]$. If we intersect $\partial\gamma \setminus K[X_t] + b_t$ with $K[X_{t'}]$, we have

$$(\partial\gamma \setminus K[X_t] + b_t) \cap K[X_{t'}] = ((\partial\gamma' + \partial\gamma_w) \setminus K[X_t] + b_t) \cap K[X_{t'}] = (\beta' + \partial\gamma_w) \setminus K[X_{t'}] + b_t \cap K[X_{t'}]$$

as $\partial\gamma' \cap K[X_{t'}] = \beta'$ and $\partial\gamma_w \subset K[X_{t'}]$. If we subtract $K[X_{t'}]$ from $\partial\gamma \setminus K[X_t] + b_t$, we find that

$$(\partial\gamma \setminus K[X_t] + b_t) \setminus K[X_{t'}] = \partial\gamma \setminus K[X_t] \setminus K[X_{t'}] + b_t \setminus K[X_{t'}] = \partial\gamma \setminus K[X_{t'}] + b_{t'} = \partial\gamma' \setminus K[X_{t'}] + b_{t'}$$

as $b_t \setminus K[X_{t'}] = b_{t'}$ by Lemma 5.9. □

6.4 Join Nodes

Let t be a join node, and let t' and t'' be the two children of t. Recall that $X_t = X_{t'} = X_{t''}$. Let $\beta \in C_{d-1}(K[X_t])$. We claim the following formula for $\mathbf{V}[\beta, t]$.

Lemma 6.4. *Let t be a joing node, and let t' and t'' be its two children. Let $\beta \in C_{d-1}(K[X_t])$. Then*

$$\mathbf{V}[\beta, t] = \min_{\substack{\beta' \in C_{d-1}(K[X_{t'}]), \beta'' \in C_{d-1}(K[X_{t''}]) \\ \beta' + \beta'' = \beta}} \mathbf{V}[\beta', t'] + \mathbf{V}[\beta'', t'']$$

Proof. Let $\beta \in C_{d-1}(K_t)$. Let γ span β at t. By Lemma 5.12, $\gamma = \gamma' + \gamma''$ where $\gamma' \in C_d(K_{t'})$ and $\gamma'' \in C_d(K_{t''})$. Let $\beta' = \partial\gamma' \cap K[X_{t'}]$ and $\beta'' = \partial\gamma'' \cap K[X_{t''}]$. Clearly,

$$\beta = \partial\gamma \cap K[X_t] = (\partial\gamma' + \partial\gamma'') \cap K[X_t] = \beta' + \beta''.$$

This proof works in the reverse direction as well. If γ' spans β' and t', γ'' spans β'' at t'', and $\beta' + \beta'' = \beta$, then $\gamma' + \gamma''$ spans β at t.

Next, we prove that $\partial\gamma \setminus K[X_t] + b_t = (\partial\gamma' \setminus K[X_{t'}] + b_{t'}) \sqcup (\partial\gamma'' \setminus K[X_{t''}] + b_{t''})$. We consider the intersection

$$\partial\gamma \setminus K[X_t] = \partial\gamma \cap K_t \setminus K[X_t] = (\partial\gamma' + \partial\gamma'') \cap (K_{t'} \setminus K[X_{t'}] + K_{t''} \setminus K[X_{t''}]).$$

As $\partial\gamma' \subset K_{t'}$, $\partial\gamma'' \subset K_{t''}$, and $K_{t'} \setminus K[X_{t'}]$ and $K_{t''} \setminus K[X_{t''}]$ are disjoint, we distribute and see that

$$\partial\gamma \setminus K[X_t] = \partial\gamma' \setminus K[X_{t'}] + \partial\gamma'' \setminus K[X_{t''}]$$

Now if we consider the intersection

$$\partial\gamma \setminus K[X_t] + b_t = (\partial\gamma' \setminus K[X_{t'}] + \partial\gamma'' \setminus K[X_{t''}]) + (b_{t'} + b_{t''}) = (\partial\gamma' \setminus K[X_{t'}] + b_{t'}) \sqcup (\partial\gamma' \setminus K[X_{t''}] + b_{t''})$$

As $\partial\gamma \setminus K[X_{t'}] + b_{t'} \subset K_{t'} \setminus K[X_{t'}]$ and $\partial\gamma \setminus K[X_{t''}] + b_{t''} \subset K_{t''} \setminus K[X_{t''}]$, these two sets are disjoint. Thus, $||\partial\gamma + b_t|| = ||\partial\gamma' + b_{t'}|| + ||\partial\gamma'' + b_{t''}||$.

As the sum of any two chains γ' and γ'' that span β' at t' and β'' at t'' spans β at t, and $\partial\gamma \setminus K[X_t] = \partial\gamma' + \partial\gamma''$, it is easy to see that the optimal chain that span β at t must be composed of optimal chains that span β' at t' and β'' at t'' respectively. $\square$

Chapter 7: Cell complexes

The problem 2DBS is similar to OBCP. In OBCP, we are looking for a d-chain c bounded by a $(d-1)$-chain b. In 2BDS, we are looking for a surface S that is bounded by a 1-dimensional simplicial complex B. Because the simplicial complex S is a pure 2-dimensional simplicial complex, we can reframe 2DBS as finding the 2-chain c such that $S = \operatorname{cl} c$. However, even by searching for the 2-chain c, we cannot solve 2DBS using our algorithm for OBCP for two reasons. First, not any chain c with closure $S = \operatorname{cl} c$ bounded by B is a surface. The links of the vertices of S must either be a simple path or simple cycle. Second, even if S is a surface, we require that S be homeomorphic to the input surface Ψ, meaning S has the same genus and orientability as Ψ. The algorithm for OBCP stores candidate solutions by how they intersect $K[X_t]$. Storing candidate solutions this way is useful as it allows us to store a number of candidate solutions that is a function of the treewidth. We cannot store candidate solutions for 2BCP by how they intersect $K[X_t]$ as these two properties, vertex links and homeomorphism class, are determined by interactions between simplices that may not be contained in the complex $K[X_t]$. To address this, we use a data structure, the annotated cell complex, that stores information about vertex links and homeomorphism class in a compressed representation using only edges and vertices in the bag $K[X_t]$. We will see that the number of candidate solutions we need to store is a function of treewidth and the genus of Ψ.

In this chapter, we introduce cell complexes and their basic properties. In Section 7.1, we explore the relationship between cell complexes and combinatorial surfaces. In Section 7.2, we see how cell complexes provide a concise way of classifying connected, compact surfaces up to homeomorphism. In Section 7.3, we introduce a generalization of the cell complex, the *annotated cell complex*, which stores the same information as a cell complex but in a more concise way that is needed for our algorithm.

A cell complex is an algebraic representation of a surface. The specific definition of a cell complex we use was introduced by Ahlfors and Sario [2] to give a proof of the Classification Theorem for Compact Surfaces, although many proofs of this theorem use an algebraic description of surfaces similar to cell complexes. See also Chapter 6 of the book by Gallier and Xu [23] for a modern treatment of cell complexes.

For a set X, let $X^{-1} = \{x^{-1} \mid x \in X\}$. We will let $(x^{-1})^{-1} = x$. For the time being, it is fine to treat $^{-1}$ as meaningless notation. A ***cell complex*** is a tuple $C = (F, E, B)$ where F and E are finite sets and $B : F \sqcup F^{-1} \to (E \sqcup E^{-1})^*$ is a map that assigns each element $A \in F \sqcup F^{-1}$ a cyclically ordered sequence $B(A) = (\overline{a_1 \dots a_m})$ where each $a_i \in E \sqcup E^{-1}$, such that (1) $B(A^{-1}) = (\overline{a_m^{-1} \dots a_1^{-1}})$ and (2) each element $e \in E \sqcup E^{-1}$ appears either once or twice in some sequence $B(A)$.

Elements of F are called ***faces*** and elements of $F \sqcup F^{-1}$ are called ***oriented faces***. Elements of E are called ***edges*** and elements of $E \sqcup E^{-1}$ are called ***oriented edges***. The sequence $B(A)$ is the ***boundary*** of A.

It is informative to visualize cell complexes as a collection of disks F identified along shared edges in their boundaries. The oriented edges a and a^{-1} are the two directed edges defined by the edge $a \in E$. An element $A \in F$ is a disk with edges on its boundary $B(A)$. The element A^{-1} is the same disk as A but with opposite orientation. The boundary $B(A^{-1})$ is the boundary of A traversed in the opposite direction as $B(A)$, i.e. $B(A)$ might be the order of the edges on the boundary when traversing the boundary clockwise and $B(A^{-1})$ would be the order of the edges when traversing the boundary counterclockwise.

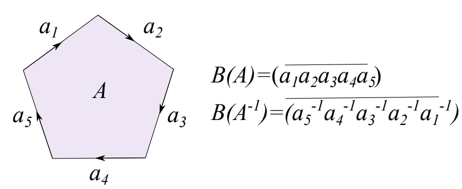

Figure 7.1: The oriented faces A and A^{-1} represent the two ways (clockwise and counterclockwise) of traversing the boundary of A.

If an oriented edge a appears in the boundary of two faces, then a on one face is glued to a^{-1} on the other face. In this way, oriented edges a and a^{-1} can be thought of as twin half-edges. This is why we store both the oriented faces A and A^{-1}. We need to ensure that both the oriented edges a and a^{-1} appear in the boundary of a face. If we repeat this for all oriented edges, then we can visualize our collection of disks F as part of a single (possibly disconnected) surface. We store both the oriented faces A and A^{-1}

to ensure that both the oriented edges a and a^{-1} appear in the boundary of a face.

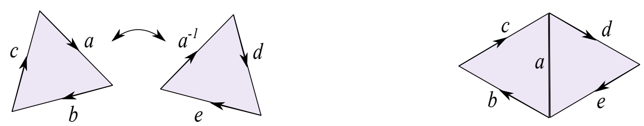

Figure 7.2: If an edge a appears on the boundary of two faces, we visualize this as a on one face being glued to a^{-1} on the other face.

A cell complex is ***connected*** if there is no partition $F_1 \sqcup F_2 = F$ and $E_1 \sqcup E_2 = E$ such that $(E_1, F_1, B|_{F_1 \sqcup F_1^{1}})$ and $(E_2, F_2, B|_{F_2 \sqcup F_2^{1}})$ are also cell complexes. If a cell complex C is disconnected, partitions $\{E_1, ..., E_k\}$ and $\{F_1, \dots, F_k\}$ such that each $(E_i, F_i, B|_{F_i \sqcup F_i^{-1}})$ is connected are the ***connected components*** of C.

Let $a \in E \sqcup E^{-1}$. A ***successor*** of a is an edge b such that ab is a substring of a boundary $B(A)$ for some $A \in F \sqcup F^{-1}$. As each oriented edge appears in at most two boundaries, then each oriented edge has at most two successors. If an oriented edge appears in two boundaries, we say it has a ***pair of successors***; otherwise, if an oriented edge appears in a single boundary, it has a ***single successor***. A ***sequence of successors*** is a sequence of edges $(a_1 \dots a_k)$ such that a_{i-1}^{-1} and a_{i+1}^{-1} are pairs of successors of a_i for $2 \leq i \leq k-1$, a_2^{-1} is the single successor of a_1, and a_{k-1}^{-1} is the single successor of a_k. If there is a face $(\overline{a})$ in a cell complex, then this face and its inverse $(\overline{a^{-1}})$ define the sequence of successors (aa^{-1}) as faces are cyclic sequences of edges. A ***cyclic sequence of successors*** is a cyclically ordered sequence of edges $(\overline{a_1 \dots a_k})$ such that a_{i-1}^{-1} and a_{i+1}^{-1} are a pair of successors for a_i, with indices taken $\bmod k$. We distinguish cyclic sequences of successors from sequences of successors with an overline. If aa^{-1} is the substring of a boundary, then $(\overline{a})$ is a cyclic sequence of successors. Sequence of successors describe sets of edges that all enter a common vertex and are analogous to rotation systems in surface graphs. Sequence of successors define vertices in cell complexes when the edges do not have their own notion of vertex; see [23].

7.1 Cell Complexes and Combinatorial Surfaces

A combinatorial surface S defines a cell complex $C = (E, F, B)$. The edges of S are the edges of C and the triangles of S are the faces of C. While an oriented edge is purely a formal construction, we associate the oriented edges of an edge $\{u, v\} \in S$ with the ordered pairs (u, v) and (v, u). For a triangle $A = \{u, v, w\} \in S$, we define the boundary of A as $B(A) = (\overline{(u, v), (v, w), (w, u)})$ and $B(A^{-1}) = (\overline{(u, w), (w, v), (v, u)})$. All edges in a combinatorial surface are incident to at most two triangles, so conditions (1) and (2) in the definition of cell complex are met.

The vertices of S define sequences of successors in C. Let (u, v) be an oriented edge. We say (u, v) ***enters*** v.

Proposition 7.1. *Let S be a combinatorial surface and C the cell complex defined by S. Let $v \in S$ be a vertex such that $\mathrm{lk}_S\, v$ is a simple cycle. The set of edges entering v form a cyclic sequence of successors.*

Proof. Let $\mathrm{lk}_K\, v = (\overline{v_1, \ldots, v_k})$. For each $v_i \in \mathrm{lk}_S\, v$, one can verify using the definition of the link that v is incident to the triangles $\{v, v_{i-1}, v_i\}$ and $\{v, v_i, v_{i+1}\}$. The boundary of $\{v, v_{i-1}, v_i\}$ in C is $(\overline{(v, v_{i-1}), (v_{i-1}, v_i), (v_i, v)})$. The boundary of the inverse of $\{v, v_i, v_{i+1}\}$ is $(\overline{(v_{i+1}, v_i), (v_i, v), (v, v_{i+1})})$. So $(v, v_{i-1}) = (v_{i-1}, v)^{-1}$ and $(v, v_{i+1}) = (v_{i+1}, v)^{-1}$ are a pair of successors to (v_i, v). $\square$

Proposition 7.2. *Let S be a combinatorial surface and C the cell complex defined by S. Let $v \in S$ such that $\mathrm{lk}_S\, v$ is a simple path. The set of edges entering v form a sequence of successors.*

Proof. The proof of this proposition is analogous to the proof of Proposition 7.1. $\square$

While a combinatorial surface can be represented as a cell complex, it is not obvious that any cell complex can be represented by a combinatorial surface. A face in a cell complex can have more than three edges on its boundary, so we cannot just reverse the construction. A cell complex describes a compact surface with boundary. In fact, there is a combinatorial surface homeomorphic to any compact surface with boundary. Phrased differently, any compact surface with boundary can be triangulated. This is a famous result known as Rado's Theorem; see for instance [23].

7.2 Surface Classification

Many cell complexes describe the same surface up to homeomorphism. We can define an equivalence relation on cell complexes that partitions cell complexes into homeomorphism classes. This is the famous Classification Theorem of Compact Surfaces.

Let C be a cell complex. A cell complex C' is an ***elementary subdivision*** of C if (1) a pair of oriented edges a and a^{-1} are replaced by bc and $c^{-1}b^{-1}$ respectively in all boundaries containing a or a^{-1} where b, c are distinct edges in K and not in K' or (2) a face A with $B(A) = (\overline{a_1 ... a_k a_{k+1} ... a_l})$ is replaced with two faces A', A'' such that $B(A') = (\overline{a_1, ..., a_k, c})$ and $B(A'') = (\overline{c^{-1} a_{k+1} ... a_l})$ where A', A'', c are not in K and the reverse operation is applied to A^{-1}.

Two cell complexes C and C' are ***equivalent*** if they are equivalent in the least equivalence relation containing the elementary subdivision relation.

Theorem 7.3 (Classification Theorem for Compact Surfaces [23])**.** *Each connected cell complex is equivalent to a cell complex $C = (F, E, B)$ where $F = \{A\}$ and either*

$$B(A) = (\overline{a_1 c_1 a_1^{-1} c_1^{-1} ... a_g c_g a_g^{-1} c_g^{-1} d_1 e_1 d_1^{-1} ... d_b e_b d_b^{-1}})$$

in which case C is an orientable surface of genus $g \geq 0$ with $b \geq 0$ boundary components, or

$$B(A) = (\overline{a_1 a_1 ... a_g a_g d_1 e_1 d_1^{-1} \dots d_b e_b d_b^{-1}})$$

in which case C is a non-orientable surface of genus $g \geq 1$ with $b \geq 0$ boundary components.

We call a substring of a boundary of the form $aca^{-1}c^{-1}$ a ***handle***, a substring of the form aa a ***crosscap***, and a substring of the form ded^{-1} a ***boundary*** if the edge e is only on the boundary of one face. A cell complex is ***non-orientable*** if it is equivalent to a cell complex with a crosscap and ***orientable*** otherwise. The above theorem states that a cell complex is characterized by the number of handles and boundaries or the number of crosscaps and boundaries it has. The ***genus*** of a cell complex is the number of handles or the number of crosscaps a cell complex has. An arbitrary cell complex is not connected, so we speak of a cell complex as being a collection of connected cell complexes of the above form.

7.3 Equivalence-Preserving Moves, Annotated Cell Complexes

One advantage of using cell complexes to describe surfaces is the equivalence relation defined by elementary subdivision. This equivalence relation allows us to develop a rich algebra on cell complexes. In this section, we give a concise algebraic description of cell complexes as a formal sum. We then define a collection of ***equivalence-preserving moves*** on these formal sums, operations we can perform on a cell complex that preserves its equivalence class. We will then use these equivalence-preserving moves to define a more general data structure, the annotated cell complex.

We now give an algebraic description of a cell complex that will allow us to easily describe and reduce cell complexes to their canonical form. Let a ***section*** be a map $\phi : F \to F \sqcup F^{-1}$ such that $\phi(A) \in \{A, A^{-1}\}$ for each $A \in F$. If we fix a section ϕ, we can represent a cell complex as the formal sum $\sum_{A \in F} B(\phi(A))$. Note that any such formal sum uniquely determines a cell complex regardless of the section ϕ used, as the boundary of any oriented face $A \notin \operatorname{im} \phi$ is determined by the boundary of $A^{-1} \in \operatorname{im} \phi$. We will express a cell complex using these formal sums as

$$\sum_{A \in F} B(\phi(A)) = (\overline{a_1 \ldots a_k}) + \ldots + (\overline{a_l \ldots a_m}).$$

We will use upper-case variables to denote substrings of boundaries, e.g. $X = a_i \ldots a_j$.

We are only interested in these formal sums up to the equivalence relation described in the previous section. We now introduce a series of operations on these formal sums that preserve the equivalence class of a cell complex.

1. The first elementary subdivision says that an edge a can be replaced with two edges bc. In the sum, we can replace summands $(\overline{aX}) + (\overline{aY}) = (\overline{bcX}) + (\overline{bcY})$ to get an equivalent cell complex. As the equivalence relation includes the symmetric closure of the elementary subdivisions, we can perform this move in reverse. If two edges bc appear consecutively in two boundaries, we can replace these two edges with a single edge a, i.e. $(\overline{bcX}) + (\overline{bcY}) = (\overline{aX}) + (\overline{aY})$.

2. The second elementary subdivision says we can replace a single face A with two faces A', A'' that share an edge. We can express this algebraically as replacing a face $(\overline{XY})$ with two faces $(\overline{Xa}) + (\overline{a^{-1}Y})$. As the equivalence relation includes the symmetric closure of the elementary subdivisions, we can also apply this move in reverse, i.e. $(\overline{Xa}) + (\overline{a^{-1}Y}) = (\overline{XY})$.

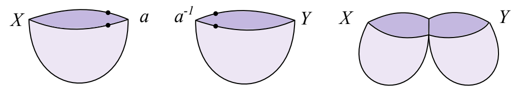

Figure 7.3: An example of Move 2.

3. Any section $\phi : F \to F \sqcup F^{-1}$ defines the same surface, so we can interchange the section ϕ for any other section ψ. This replaces one or more summands $B(A) = (\overline{a_1 \dots a_k})$ in the formal sum with $B(A^{-1}) = (\overline{a_k^{-1} \dots a_1^{-1}})$.

4. As each summand $(\overline{a_1 \dots a_k})$ is a cyclically ordered sequence, we can replace a face $(\overline{a_1 \dots a_k})$ with $(\overline{a_2 \dots a_k a_1})$.

5. We can remove any instance of aa^{-1} from a boundary. Lemma 7.4 proves this.

Lemma 7.4. *Let K be a cell complex. Let A be a face of K such that $B(A) = (\overline{Xaa^{-1}})$. There is an equivalent cell complex K' without a and A and with a new face A' such that $B(A') = (\overline{X})$.*

Proof. We apply a series of equivalence-preserving moves to $(\overline{Xaa^{-1}})$.

$$\begin{aligned}
(\overline{Xaa^{-1}}) &= (\overline{Xab}) + (\overline{b^{-1}a^{-1}}) && \text{by (2)} \\
&= (\overline{Xab}) + (\overline{ab}) && \text{by (3)} \\
&= (\overline{Xc}) + (\overline{c}) && \text{by (1)} \\
&= (\overline{Xc}) + (\overline{c^{-1}}) && \text{by (3)} \\
&= (\overline{X}) && \text{by (2)}
\end{aligned}$$

□

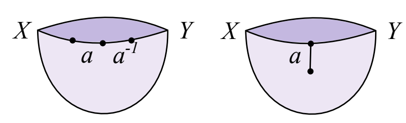

Figure 7.4: An example of Move 5

6. Boundary Components. The next two moves are inspired by a lecture of Wildberger [27]. If b and b^{-1} both appear in the boundary of a face $B(A) = (\overline{XbYb^{-1}})$, we think of X and Y as being separate boundary components of the sphere connected by a path b. As the interior of the face is an open disk and is therefore path connected, we can connect these boundary components anywhere along X or Y. The following lemma formalizes this idea.

Lemma 7.5. *Let K be a cell complex. Let A be a face of K such that $B(A) = (\overline{X_1X_2bYb^{-1}})$. There is an equivalent cell complex K' without b and A and with a new edge c and a new face A' such that $B(A') = (\overline{X_2X_1cYc^{-1}})$.*

Proof. We prove this by repeating applying moves (4) and (2) to the boundary of A.

$$\begin{aligned}
(\overline{X_1X_2bYb^{-1}}) &= (\overline{b^{-1}X_1X_2bY}) && \text{by (4)}\\
&= (\overline{b^{-1}X_1c}) + (\overline{c^{-1}X_2bY}) && \text{by (2)}\\
&= (\overline{X_1cb^{-1}}) + (\overline{bYc^{-1}X_2}) && \text{by (4)}\\
&= (\overline{X_1cYc^{-1}X_2}) && \text{by (2)}\\
&= (\overline{X_2X_1cYc^{-1}}) && \text{by (4)}
\end{aligned}$$

□

The lemma tells us that for a face containing the edges b and b^{-1} of the form $(\overline{XbYb^{-1}})$, we can cyclically permute X and Y arbitrarily so long as they are connected by some edge and its inverse. We express this choice of connection by introducing new notation. We write $(\overline{XbYb^{-1}}) = (\overline{X})(\overline{Y})$ where the formal multiplication of boundaries denotes $(\overline{X})$ and $(\overline{Y})$ are connected by some edge b and its inverse b^{-1}. If $B(A) = (\overline{X})(\overline{Y})$, we call each multiplicand a ***boundary component*** of A.

The individual boundary components of A have the same properties as the entire boundary $B(A)$. The lemma implies that $(\overline{X_1X_2})(\overline{Y}) = (\overline{X_2X_1})(\overline{Y})$, i.e. that we can cyclically permute a boundary component while maintaining equivalence. While we cannot invert just one of $(\overline{X})$ or $(\overline{Y})$ while maintaining equivalence, it is easy to verify that $(\overline{X})(\overline{Y}) = (\overline{Y^{-1}})(\overline{X^{-1}})$.

We can also have more than two boundary component. If we have boundary components $(\overline{X})(\overline{YaZa^{-1}})$, we can write write this $(\overline{X})(\overline{Y})(\overline{Z})$. We can arbitrarily permute boundary components while maintaining equivalence. This can be seen as

$$(\overline{X})(\overline{Y})(\overline{Z}) = (\overline{X})(\overline{YaZa^{-1}}) = (\overline{X})(\overline{ZaYa^{-1}}) = (\overline{X})(\overline{Z})(\overline{Y}).$$

For a face $(\overline{XbYb^{-1}})$, if we remove b to create the face $(\overline{X})(\overline{Y})$, we define the successors of edges as substrings of the individual boundary components. In particular, in the face $(\overline{a_1 \dots a_k})(\overline{X})$, then we defined the successor of a_k to be a_1.

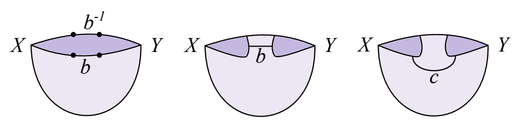

Figure 7.5: If we identify the edges b and b^{-1} on a face, then the face turns into a sphere with two boundary components. We can connect then these boundary components with any edge c, not just b.

7. **Annotations.** Theorem 7.3 tells us that a surface is completely characterized by the number of handles or crosscaps and number of boundaries it has. We take this idea a step further and prove that up to equivalence a handle, crosscap, or boundary on a face is independent of the rest of the face. The following lemmas formalize this idea.

Lemma 7.6. *Let A be a face of K such that $B(A) = (\overline{aba^{-1}b^{-1}XY})$. There is an equivalent cell complex K' without A, a, and b and with a new face A' and edges e, f such that $B(A') = (\overline{efe^{-1}f^{-1}YX})$.*

Lemma 7.7. *Let A be a face of K such that $B(A) = (\overline{aaXY})$. There is an equivalent cell complex K' without A and a a new face A' and edge d such that $B(A') = (\overline{ddYX})$.*

Lemma 7.8. *Let A be a face of K such that $B(A) = (\overline{bab^{-1}XY})$ such that a appears once in the boundary of all faces. There is an equivalent cell complex K' without A and b, a new face A' and edges c, d such that $B(A') = (\overline{cdc^{-1}YX})$.*

Proofs of these Lemmas can be found in Appendix A.

If we have a face $(\overline{HX})$ where H is a handle, we can store a face equivalent to $(\overline{HX})$ by storing $(\overline{X})$ as a face and noting that $(\overline{X})$ has a handle. We can attach a handle anywhere along $(\overline{X})$ by Lemma 7.6 and regain an equivalent cell complex. The same applies if H is a crosscap or a boundary. These lemmas motivates a new data structure, the ***annotated cell complex***. An annotated cell complex is a cell complex where each face is annotated with a genus, number of boundary components, and a boolean to indicate whether or not this face is orientable.

With this extra information, we can store an annotated cell complex equivalent to a cell complex that is defined with fewer edges. Furthermore, by Theorem 7.3, all cell complexes are equivalent to an annotated cell complex with *no* edges. For example, the cell complex of the torus $(\overline{aba^{-1}b^{-1}})$ is equivalent to the annotated cell complex () with the face annotated to have genus 1, 0 boundary components, and to be orientable. If a cell complex is disconnected, then the cell complex is equivalent to a sum of faces () with no edges and just annotations. We call a face with no edges an ***empty face***.

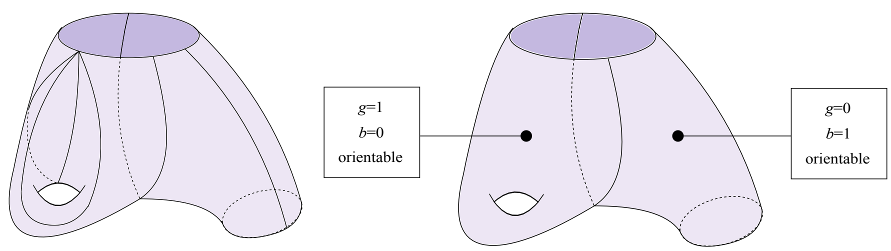

Figure 7.6: Annotations allow us to store cell complexes using fewer edges. The cell complex in this example has a handle and a boundary. We are able to record these features with annotations instead of the edges that actually make up these features.

7.4 Cell Complex Miscellany

We now present several lemmas without commentary that we will use later for classifying surfaces. The lemmas give more general criteria for identifying a handle or crosscap on a face than having substrings of the form $aba^{-1}b^{-1}$ or aa in its boundary. Each of the lemmas is from the proof of Lemma 6.1 in [23]; each of the corollaries is some restatement of the lemmas using the notation of boundary components.

Lemma 7.9. *Let A be a face of K such that $B(A) = (\overline{aUbVa^{-1}Xb^{-1}Y})$. There is an equivalent cell complex K' without A, a, and b and with a new face A' and new edges c, d such that $B(A') = (\overline{cdc^{-1}d^{-1}YXVU})$.*

Corollary 7.10. *Let A be a face of K such that $B(A) = (\overline{aX})(\overline{a^{-1}Y})$. There is an equivalent cell complex K' without A and a and with a new face A' and new edge c, d such that $B(A') = (\overline{cdc^{-1}d^{-1}YX})$.*

Lemma 7.11. *Let A be a face of K such that $B(A) = (\overline{aXaY})$. There is an equivalent cell complex K' without A and a and with a new face A' and new edge b such that $B(A') = (\overline{bbY^{-1}X})$.*

Corollary 7.12. *Let A be a face of K such that $B(A) = (\overline{aX})(\overline{aY})$. There is an equivalent cell complex K' without A and a, a new face A' and new edges b, c such that $B(A') = (\overline{ccXbbY^{-1}})$.*

Proof. This follows from Lemma 7.7.

$$\begin{aligned} (\overline{aX})(\overline{aY}) &= (\overline{aXbaYb^{-1}}) && \text{by (6)} \\ &= (\overline{ccXbbY^{-1}}) && \text{by Lemma 7.7} \end{aligned}$$

□

Lemma 7.13 (Dyck's Theorem)**.** *Let A be a face with a handle and a crosscap. There is an equivalent cell complex K' without A with a new face A' such that A' has three crosscaps.*

Chapter 8: 2-Dim-Bounded-Surface

Let K be a 2-dimensional simplicial complex. Let $B \subset K$ be a disjoint union of simple cycles in K. Let Ψ be a connected, compact surface specified by its genus g and orientability. The problem **2-Dim-Bounded-Surface** asks to find a connected combinatorial surface S in K homeomorphic to Ψ with boundary B. Let (T, X) a nice tree decomposition of K. We will use a dynamic program on T to compute S.

Let t be a node of T. We consider the same subcomplexes K_t and $K[X_t]$ as we did in the Chapter 5 with $d = 2$. Assume a combinatorial surface $S \subset K$ with boundary B exists. Consider the intersection

$$S_t := S \cap K_t.$$

Note that S_t need not be a surface, or even a pure 2-complex. So, the link of a vertex $v \in S_t$ need not be homeomorphic to a simple cycle or a simple path. As $\operatorname{lk}_{S_t} v \subset \operatorname{lk}_S v$, then $\operatorname{lk}_{S_t} v$ is either a simple cycle or a collection of simple paths and vertices. However, Lemma 8.2 proves that it can only be the case that $\operatorname{lk}_{S_t} v \neq \operatorname{lk}_S v$ if $v \in X_t$. Intuitively, S_t is "surface-like" everywhere except possibly in the intersection $S_t \cap K[X_t]$.

Lemma 8.1. *Let t be a node in a tree decomposition. Let $\tau \in K_t \setminus K[X_t]$ and $\sigma \in K$ a coface of τ. Then $\sigma \in K_t \setminus K[X_t]$.*

Proof. As $\tau \notin K[X_t]$ and $\tau \subset \sigma$, we immediately have that $\sigma \notin K[X_t]$. We now prove that $\sigma \in K_t$. By Corollary 3.3, there is a node t_σ in T such that $\sigma \in K[X_{t_\sigma}]$. Suppose t_σ is not in the subtree rooted at t. As $\tau \subset \sigma$, then $\tau \in K[X_{t_\sigma}]$ as well. There is a node t_τ in the subtree rooted at t such that $\tau \in K[X_{t_\tau}]$ by Lemma 5.2. The set of nodes containing τ form a connected subtree of T by Corollary 3.3; however, $\tau \notin K[X_t]$ by assumption, a contradiction as t lies on the unique path between t_τ and t_σ. So t_σ must be in the subtree rooted at t. $\square$

Lemma 8.2. *Let $v \in K_t \backslash K[X_t]$ be a vertex. Then $lk_S\, v = lk_{S_t}\, v$.*

Proof. As $S_t \subset S$, we immediately have that $\operatorname{lk}_{S_t} v \subset \operatorname{lk}_S v$. We need only show that $\operatorname{lk}_S v \subset \operatorname{lk}_{S_t} v$. Any simplex $\tau \in \operatorname{lk}_S v$ is incident to a common coface σ with v. Since

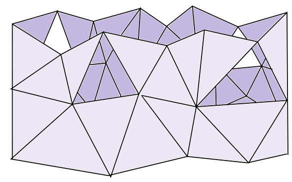

Figure 8.1: A candidate solution is "surface-like" everywhere except possibly at $K[X_t]$.

$v \in K_t \setminus K[X_t]$ and σ is a coface of v, we have $\sigma \in K_t \setminus K[X_t]$ by Lemma 8.1. Thus, $\tau \in K_t \setminus K[X_t]$ because $\tau \subset \sigma$, and $\operatorname{lk}_S v \subset \operatorname{lk}_{S_t} v$. □

Lemma 8.2 gives a way of finding candidate solutions at a node t, subcomplexes $\Sigma \subset K_t$ that could be extended to a solution S. We want to verify that for each vertex $v \in K_t$ that $\operatorname{lk}_\Sigma v$ could equal $\operatorname{lk}_{S_t} v$ for a surface S with boundary B, assuming such a surface exists. If $v \in K_t \setminus K[X_t]$, then by Lemma 8.2, we want $\operatorname{lk}_\Sigma v = \operatorname{lk}_S v$. For a vertex $v \in (K_t \setminus K[X_t]) \setminus B$, this is true if $\operatorname{lk}_\Sigma v$ is a simple cycle or if $\operatorname{lk}_\Sigma v$ is empty (in the case that $v \notin \Sigma$.) For a vertex $v \in (K_t \setminus K[X_t]) \cap B$, this is true if $\operatorname{lk}_\Sigma v$ is a simple path with endpoints that are the neighbors of v in B. Note that it *must* be the case that $v \in \Sigma$ if $v \in B$. We say that $\operatorname{lk}_\Sigma v$ is ***complete*** if this is the case, depending on whether or not $v \in B$. The link of a vertex $v \in K_t \cap K[X_t]$ may not be complete, but we would like to be able to make it complete by adding more triangles. A link of a vertex $\operatorname{lk}_\Sigma v$ is ***admissible*** if $\operatorname{lk}_\Sigma v$ is a either a single simple cycle or a (possibly empty) collection of simple paths. A complete link is also admissible.

Our algorithm considers a set of candidate solutions at each bag. A surface S in K is a pure 2-complex, so it is natural to associate S with a 2-chain $c \in C_2(K)$ where $S = \operatorname{cl} c$. A ***candidate solution at a node t*** is a 2-chain $\gamma \in C_2(K_t)$ with closure $\Sigma = \operatorname{cl}\gamma$ such that

(1) $\operatorname{lk}_\Sigma v$ is admissible for each vertex $v \in K_t \cap K[X_t]$

(2) $\operatorname{lk}_\Sigma v$ is complete for each vertex $v \in K_t \setminus K[X_t]$, and

(3) Σ is connected, or each connected component of Σ intersects $K[X_t]$.

A candidate solution γ at the root r defines a surface $\Sigma = \operatorname{cl}\gamma$ with boundary B. As $X_r = \emptyset$ and $K_r = K$, each vertex $v \in K_r \setminus K[X_r] = K$ has complete link. A simplex $\sigma \in \Sigma$ is therefore on the boundary of Σ iff $\sigma \in B$. As $\Sigma \cap K[X_r] = \emptyset$, then Σ must be connected. We verify that Σ is a solution to 2DBS by checking that Σ has the correct genus and orientability.

8.1 Storing Candidate Solutions

Let $\mathbf{D}[\mathbf{t}]$ be the set of candidate solutions at a node t. We will not store the set of chains $\mathbf{D}[\mathbf{t}]$ explicitly, as there can be exponentially many candidate solutions at a node. Instead, we store a set of annotated cell complexes equivalent to the closures of the set of candidate solutions in a table $\mathbf{V}$ indexed by the nodes of T. The set of annotated cell complexes $\mathbf{V}[\mathbf{t}]$ is computed as follows. For each candidate solution $\gamma \in \mathbf{D}[\mathbf{t}]$ with $\Sigma = \operatorname{cl} \gamma$, we use equivalence preserving moves to remove each edge incident to vertices $v \notin X_t$ from Σ. In particular, the entry at the root $\mathbf{V}[\mathbf{r}]$ will contain annotated cell complexes with a single face that has no edges explicitly in its boundary and instead only stores topological information in an annotation. We give more details on how we remove edges in the Section 8.2.

The size of an annotated cell complex in $\tilde{\Sigma} \in \mathbf{V}[\mathbf{t}]$ is $O(k^2)$. Intuitively, this is because there are $O(k^2)$ edges in $K[X_t]$, although we also add $O(k)$ dummy edges not in $K[X_t]$. We will see shortly when we add these dummy edges.

After removing each of these edges, then many candidate solutions may be transformed into the same annotated cell complex. All candidate solutions that are transformed into the same annotated cell complex are homeomorphic as they are equivalent to a common cell complex, so we only keep one of these annotated cell complexes.

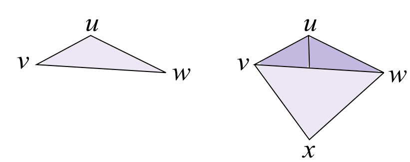

Figure 8.2: Multiple candidate solutions may reduce to the same cell complex after removing edges $e \notin K[X_t]$. If $X_t = \{u, v, w\}$, then both of the above candidate solutions reduce to the same cell complex.

We emphasize that γ is a 2-chain, Σ is a subcomplex of K, and $\tilde{\Sigma}$ is an annotated cell complex equivalent to Σ. Let us justify why we care about three redundant objects γ, Σ, and $\tilde{\Sigma}$. While we are only looking for the subcomplex Σ, we consider the 2-chain γ as a candidate solution as we know how chain groups interact with tree decomposition as developed in Chapter 5. In our algorithm for OBCP, we indexed our candidate solutions by how they interacted with the current bag. Topological properties of Σ like its genus are global properties and cannot be determined simply by how Σ interacts with a single bag in the tree decomposition. We could store the entire 2-chain γ; however, this would make our algorithm too slow as there can be too many candidate solutions at a given node. By compressing candidate solutions to annotated cell complexes and only storing these annotated cell complexes up to homeomorphism, we can reduce the size of a table entry $\mathbf{V}[\mathbf{t}]$ to only be exponentially dependent on the treewidth k.

8.2 Removing Edges

Each time we forget a vertex w, we remove all the edges incident to w from each of our candidate solution. Let t be a node that forgets w and let t' be the child of t. Recall that $X_t \sqcup \{w\} = X_{t'}$. Let γ be a candidate solution at t and let $\tilde{\Sigma}$ be an annotated cell complex equivalent to $\Sigma = \operatorname{cl} \gamma$ with edges between vertices in $X_{t'}$. We use equivalence-preserving moves to remove all edges incident to w. As γ is a candidate solution at t, then w must have complete link in Σ. We prove that this condition is sufficient to be able to remove all edges incident to w.

Lemma 8.3. *Let $w \in \Sigma$ be a vertex such that $\mathit{lk}_\Sigma\, w$ is complete. We can remove each edge incident to w with equivalence-preserving moves.*

This section develops the background needed to prove Lemma 8.3; we prove Lemma 8.3 at the end of this section. We begin with the following lemma. We define the term *boundary dummy edge* in the following paragraphs.

Lemma 8.4. *Let $\Sigma \subset K_t$. Let $\tilde{\Sigma}$ be an annotated cell complex equivalent to Σ with edges between vertices in $K[X_t]$. The link of a vertex $v \in K[X_t]$ is complete if and only if one of the following conditions hold:*

1. *$v \notin B$ and either*
   - *(i) no edges in $\tilde{\Sigma}$ enter v.*
   - *(ii) the edges entering v in $\tilde{\Sigma}$ form a cyclic sequence of successors $(\overline{a_1 \ldots a_k})$.*

2. $v \in B$ and the edges entering v in $\tilde{\Sigma}$ form a sequence of successors $(a_1 \dots a_k)$ such that a_1 and a_k are either in B or are boundary dummy edges.

A proof of Lemma 8.4 can be found in Appendix B.

Let a be an edge incident to w. Below is a case analysis of all ways we can remove a from our annotated cell complex. Let $\tilde{\Sigma} = (\overline{a_1 \dots a_k}) + \dots + (\overline{a_l \dots a_m})$. If a only appears once in $\tilde{\Sigma}$, we will remove all edges incident to w that appear twice in $\tilde{\Sigma}$ before we remove a.

1. **Edge or Inverse on Different Faces.** If a is the boundary of two faces $(\overline{Xa}) + (\overline{Ya})$, we can invert one of the faces $(\overline{Ya}) = (\overline{a^{-1}Y^{-1}})$ and combine the faces $(\overline{Xa}) + (\overline{a^{-1}Y^{-1}}) = (\overline{XY^{-1}})$ using move (2). For other combinations of a or a^{-1}, we can invert faces as necessary so that a appears in one face and a^{-1} appears in the other. If either $(\overline{Xa})$ or $(\overline{a^{-1}Y^{-1}})$ is non-orientable, then $(\overline{XY^{-1}})$ is non-orientable. If $(\overline{Xa})$ had genus g_1 and $(\overline{Ya})$ had genus g_2, then $(\overline{XY^{-1}})$ has genus $g_1 + g_2$. If one face is orientable and the other is non-orientable, then by Lemma 7.13, we double the genus of the orientable face before adding the two genuses. The number of boundary components of $(\overline{XY^{-1}})$ is likewise the sum of the number of boundary components of $(\overline{Xa})$ and $(\overline{Ya})$. See Figure 7.3.

2. **Edge and Inverse Non-Consecutively on Same Boundary Component of Same Face.** If a and a^{-1} appear non-consecutively on the same boundary component of a face $(\overline{XaYa^{-1}})$, we can break this boundary component into two boundary components $(\overline{X})(\overline{Y})$ using move (6). See Figure 7.5.

3. **Edge Twice on Same Boundary Component of Same Face.** If a appears twice on the boundary component of the same face $(\overline{XaYa})$, then by Lemma 7.11, this face is equivalent to $(\overline{bbY^{-1}X})$. We can remove the substring bb, keep the face $(\overline{Y^{-1}X})$, and update the face's surface information. The face is non-orientable. If the face was orientable before removing bb and had genus g, there are g handles on the face f. One handle is equivalent to two crosscaps in the presence of a crosscap by Lemma 7.13, so there are $2g + 1$ crosscaps after removing bb. If the face was non-orientable before removing bb, there are now $g + 1$ crosscaps.

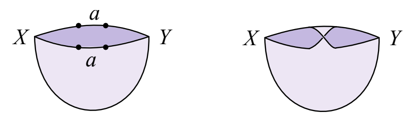

Figure 8.3: An example of Case 3

4. **Edge and Inverse on Different Boundary Components of Same Face.** If a and a^{-1} appear on the boundary of the same face but on different boundary components $(\overline{Xa})(\overline{Ya^{-1}})$, then by Corollary 7.10, we can combine these boundaries into a single boundary component $(\overline{cdc^{-1}d^{-1}YX})$. We can remove the edges c, d and annotate the face $(\overline{YX})$ to have $+1$ genus if the face is orientable and $+2$ genus if the face is not orientable.

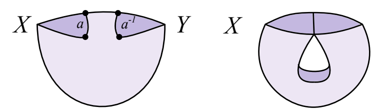

Figure 8.4: An example of Case 4

5. **Edge Twice on Different Boundary Components of Same Face.** If a appears twice on different boundary components of the same face $(\overline{Xa})(\overline{Ya})$, by Corollary 7.12 this face is equivalent to $(\overline{bbccXY^{-1}})$. We update the boundary of the face $(\overline{XY^{-1}})$. This face is non-orientable. If this face was orientable with genus g before removing $bbcc$, we update the genus to $2g + 2$. If this face was non-orientable with genus g before removing $bbcc$, we update the genus to $g + 2$.

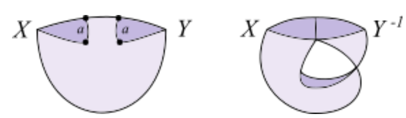

Figure 8.5: An example of Case 5

6. **Edge and Inverse Consecutively on Same Boundary Component of Same Face.** If a and a^{-1} appear consecutively as in $(\overline{aa^{-1}X})$, we would like to use move (5) to simplify $(\overline{aa^{-1}X}) = (\overline{X})$. If a enters w (i.e. $a = (v, w)$), then we remove a. If a does not enter w, (i.e. $a = (w, v)$), then we have to take an additional step before removing a. As link $\mathrm{lk}_\Sigma v$ is admissible and a forms a cyclic sequence of

successors $(\overline{a})$, then a and a^{-1} must be the only edges incident to v in $\tilde{\Sigma}$. Erasing a would remove all edges incident to v from $\tilde{\Sigma}$. We use the edges entering a to store the information on the link of v, so we need to take an additional step to remember this information about v.

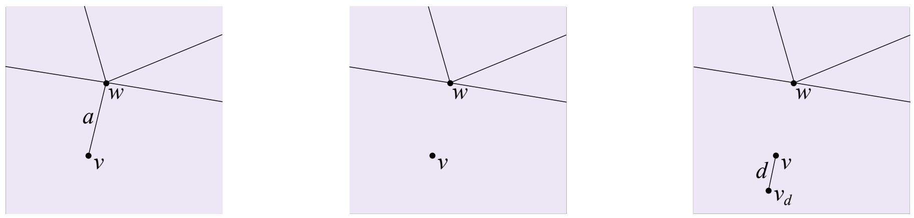

Figure 8.6: Forgetting the edge $a = (w, v)$ will remove any reference of v from $\tilde{\Sigma}$. We remember v with a dummy edge d.

We will keep a record of v with ***interior dummy edges***. Before removing a, we first add in dummy edges $(\overline{add^{-1}a^{-1}X})$ using move (5). We can imagine these edges as connecting v to a dummy vertex v_d, or $d = (v, v_d)$. We then use move (6) to break the face into boundary components $(\overline{dd^{-1}})(\overline{X})$. The edges dd^{-1} are added to $\tilde{\Sigma}$ while maintaining equivalence, so adding the substring dd^{-1} does not change the topology of $\tilde{\Sigma}$. A ***real edge*** is an edge in $\tilde{\Sigma}$ that is not a dummy edge. We assume dummy edges are marked.

The sole purpose of the dummy edge d is to denote that v is still in the cell complex, even after all vertices that share an edge with v have been forgotten. Note that before removing a, the edges incident to v formed a cyclic sequence of successors $(\overline{a})$ and $\mathrm{lk}_\Sigma\, v$ was complete. After adding d and removing a from the boundary, the edges incident to v still form a cyclic sequence of successors $(\overline{d})$, so by Lemma 8.4, our algorithm will still recognize $\mathrm{lk}_\Sigma\, v$ as complete. When we forget a vertex v incident to dummy edges dd^{-1}, we will simply remove the boundary component $(\overline{dd^{-1}})(\overline{X}) = (\overline{X})$. Note that a vertex v can only be incident to at most 2 dummy edges as $\mathrm{lk}_\Sigma\, v$ is complete. No more edges incident to v will be added to $\tilde{\Sigma}$ by our algorithm as this would make $\mathrm{lk}_\Sigma\, v$ inadmissible. The number of edges in $\tilde{\Sigma}$ is still $O(k^2)$, even with the addition of interior dummy edges.

7. **Edge on Boundary of One Face.** Let a be an edge incident to w that only appears once in the cell complex $\tilde{\Sigma}$. As w passed the condition of Lemma 8.4, then

the edges incident to w form a sequence of successors $(a_1 \ldots, a_k)$. After removing all edges a_i with $2 \le i \le k-1$ that appeared in the boundary of two faces, then the only edges entering w are a_1 and a_k. We conclude that a_1 and a_k form a sequence of successors $(a_1 a_k)$ as $\mathrm{lk}_\Sigma\, w$ is complete.

If $a_1 = a$ and $a_k = a^{-1}$, then a must be the only edge in a boundary component $(\overline{a})(\overline{X})$. This face is equivalent to $(\overline{a})(\overline{X}) = (\overline{abXb^{-1}}) = (\overline{b^{-1}abX})$. The string $b^{-1}ab$ is a boundary component, so we remove $b^{-1}ab$ and update the record of this face to have $+1$ boundary components.

If $a_1 \neq a_k^{-1}$, then a_1 and a_k^{-1} will be consecutive on the same face $(a_1 a_k^{-1} X)$. We can then use move (1) and replace $a_1 a_k^{-1}$ with an edge d. In particular, if $a_1 = (w_1, w)$ and $a_k = (w, w_2)$, then $d = (w_1, w_2)$.

Figure 8.7: Replacing the edges a_1 and a_k^{-1} with a boundary dummy edge d.

Note that the edge $\{w_1, w_2\}$ is not necessarily in K, and if $\{w_1, w_2\} \in K$, we want to distinguish the edge $\{w_1, w_2\} \in K$ from the edge $d = \{w_1, w_2\}$ that replaces $a_1 a_k^{-1}$. We define a ***boundary dummy edge*** as an edge that replaces two edges ab where $a, b \in B$ or a and b are dummy edges. We assume dummy edges are marked.

A boundary dummy edge $\{w_1, w_2\}$ always replaces a segment $(w_1, \ldots, w_2)$ of B. As a vertex w_1 is incident to either zero or two edges in B, then w_1 is incident to at most two dummy edges. An annotated cell complex at a node t is defined by $O(k^2)$ edges, dummy and real.

We are now ready to prove Lemma 8.3. We prove another lemma first.

Lemma 8.5. *Let a be an edge in $\tilde{\Sigma}$ with a pair of successors $b^{-1} \neq a^{-1}$ and $c^{-1} \neq a^{-1}$. After removing the edge a, the edges b and c^{-1} will be successors to one another.*

Proof. We verify this in each of the cases for removing the edge a. We do not need to verify cases 6 and 7, as in case 6, a^{-1} is a successor to a, and in case 7, a has a single successor.

1. $(\overline{Xa}) + (\overline{a^{-1}Y}) = (\overline{b^{-1}X'a}) + (\overline{a^{-1}Y'^{-1}c}) = (\overline{b^{-1}X'Y'^{-1}c})$

2. $(\overline{XaYa^{-1}}) = (\overline{Xab^{-1}Y'ca^{-1}}) = (\overline{X})(\overline{b^{-1}Y'c})$

3. $(\overline{XaYa}) = (\overline{c^{-1}X'ab^{-1}Y'a}) = (\overline{Y'^{-1}bc^{-1}X'})$

4. $(\overline{Xa})(\overline{Ya^{-1}}) = (\overline{c^{-1}X'a})(\overline{Y'ba^{-1}}) = (\overline{Y'bc^{-1}X'})$

5. $(\overline{Xa})(\overline{Ya}) = (\overline{c^{-1}X'a})(\overline{b^{-1}Y'a}) = (\overline{c^{-1}X'Y^{-1}b})$

In case 2, note that this holds even if Y only contains a single edge c. In this case, c and c^{-1} are a pair of successors to a. After removing a, then $(c^{-1})^{-1} = c$ is a successor to c in the boundary component $(\overline{c})$. □

Proof of Lemma 8.3. If $w \in B$, then the set of edges entering w form a sequence of successors $(a_1 \dots a_k)$. After removing an edge a_i with $2 \leq i \leq k-1$, then by Lemma 8.5, the set of edges entering w will form a sequence of successors $(a_1 \dots a_{i-1}a_i \dots a_k)$. Eventually, the set of edges entering w will form a sequence of successors (a_1a_k), at which point, we can remove both edges as in Case 7.

If $w \notin B$, then the set of edges entering w form a cyclic sequence of successors $(\overline{a_1 \dots a_k})$. By Lemma 8.5, removing an edge a_i makes a_{i-1} and a_{i+1} successors. So after removing a_i, the set of edges entering w still form a cyclic sequence of successors $(\overline{a_1 \dots a_{i-1}a_{i+1} \dots a_k})$. Eventually, the set of edges entering w will form a cyclic sequence of successors with a single edge $(\overline{a})$. At this point, aa^{-1} is the substring of some boundary and we can remove a. □

8.3 Checking Candidacy on $\tilde{\Sigma}$

The conditions for a 2-chain γ to be a candidate solution use properties of the closure $\Sigma = \operatorname{cl}\gamma$. We do not store the chain γ or the closure Σ, but instead only store the compressed representation $\tilde{\Sigma}$. Without γ or Σ, it is not obvious that we can check the conditions for candidacy.

We can in fact check the conditions of candidacy only using $\tilde{\Sigma}$. We have already seen in Lemma 8.4 a condition to check that the link of a vertex $v \in K[X_t]$ is complete.

Checking the condition of this lemma can be performed in poly(k) time by iterating over the edges in $\tilde{\Sigma}$ and building the sequence of successors. If $\tilde{\Sigma}$ does not satisfy this condition, then $\mathrm{lk}_\Sigma v$ is not complete and we discard the annotated cell complex $\tilde{\Sigma}$.

Lemma 8.6 gives a similar condition to check that the link of a vertex $v \in K[X_t]$ is admissible.

Lemma 8.6. *Let $\Sigma \subset K_t$. Let $\tilde{\Sigma}$ be an annotated cell complex equivalent to Σ with edges between vertices in $K[X_t]$. The link of a vertex $lk_\Sigma v$ is complete if and only if one of the following conditions hold.*

1. *the edges entering v in $\tilde{\Sigma}$ form a cyclic sequence of successors $(\overline{a_1 \dots a_k})$.*
2. *the edges entering v in $\tilde{\Sigma}$ form a (possibly empty) collection of sequence of successors.*

Again, we check that the link of a vertex v is admissible by iterating over the edges in $\tilde{\Sigma}$ and building the sequences of successors. In particular, we just need to verify that if some of the edges entering v form a cyclic sequence of successors, then no other edge enters v. Note that this condition excludes the possibility that an edge appears three times in an annotated cell complex.

We also have a condition to verify that Σ is connected or each connected component of Σ intersects $K[X_t]$. Lemma 8.7 gives a condition to verify whether Σ has a connected component that does not intersect $K[X_t]$.

Lemma 8.7. *A cell complex Σ has a connected component that does not intersect $K[X_t]$ if and only if $\tilde{\Sigma}$ has an empty face.*

We therefore verify that Σ is connected or each connected component of Σ intersects $K[X_t]$ by verifying that $\tilde{\Sigma}$ has an empty face only if this is only face of $\tilde{\Sigma}$.

The proofs of Lemmas 8.4, 8.6, and 8.7 can be found in Appendix B.

8.4 Dynamic Program on Tree Decomposition

We now give a dynamic program to compute the set of annotated cell complexes $\mathbf{V}[\mathbf{t}]$ at each node in our tree decomposition. Throughout, $\Sigma = \mathrm{cl}\,\gamma$ for the chain γ. For chains with modifiers like subscripts or primes such as γ_w or γ', the closures of these chains will have the same modifiers, e.g. $\Sigma_w = \operatorname{cl}\gamma_w$ and $\Sigma' = \operatorname{cl} C'$. Additionally, an added tilde always denotes an annotated cell complex $\tilde{\Sigma}$ equivalent to a simplicial complex Σ after removing edges $e \notin K[X_t]$.

8.5 Leaf Nodes

Let t be a leaf node. The subcomplex K_t is empty, so there are no candidate solutions at t and $\mathbf{V}[\mathbf{t}] = \emptyset$.

8.6 Introduce Nodes

Lemma 8.8. *Let t be an introduce node and let t' be its unique child. We have $\mathbf{D}[\mathbf{t}] = \mathbf{D}[\mathbf{t'}]$.*

Proof. Recall that $X_t = X_{t'} \sqcup \{w\}$ for some vertex w. Recall also by Lemma 5.5 that $C_2(K_t) = C_2(K_{t'})$. Let $\gamma \in C_2(K_t)$. We show that γ is a candidate solution at t if and only if γ is a candidate solution at t'.

First, let γ be a candidate solution at t. We first verify that the link of each vertex $v \in K_{t'} \backslash K[X_{t'}]$ is complete. Let $v \in K_{t'} \backslash K[X_{t'}]$. By Lemma 5.4 $K_{t'} \backslash K[X_{t'}] = K_t \backslash K[X_t]$, so $v \in K_t \setminus K[X_t]$ and $\operatorname{lk}_\Sigma v$ is complete as γ is a candidate solution at t. We next verify that the link of each vertex $v \in K_{t'} \cap K[X_{t'}]$ is admissible. As $K[X_{t'}] \subset K[X_t]$, then the link of v is admissible as γ is a candidate solution at t. Finally, we verify that either Σ is connected or each connected component of Σ intersects $K[X_{t'}]$. As γ is a candidate solution at t, then either Σ is connected or each connected component of Σ intersects $K[X_t]$. If a connected component of Σ intersects $K[X_t]$ but not $K[X_{t'}]$, then w is in this connected component. This cannot be the case as $\Sigma \subset K_{t'}$ but $w \notin K_{t'}$ by Lemma 5.3, so the condition holds. We conclude that γ is a candidate solution at t'.

Second, suppose that γ is a candidate solution at t'. We first verify that the link of each $v \in K_t \setminus K[X_t]$ is complete. By Lemma 5.4 $K_t \setminus K[X_t] = K_{t'} \setminus K[X_{t'}]$, so $v \in K_t \setminus K[X_t]$ and $\operatorname{lk}_\Sigma v$ is complete as γ is a candidate solution at t. We next verify that the link of each vertex $v \in K_t \cap K[X_t]$ is admissible. As $X_t = X_{t'} \sqcup \{w\}$, then $v \in X_{t'}$ or $v = w$. If $v \in X_{t'}$, then the link of v is admissible as γ is candidate solution at t'. It cannot be the case that $v = w$ as $\Sigma \subset K_{t'}$ and $w \notin K_{t'}$ by Lemma 5.3. Hence the link of each vertex $v \in \Sigma \cap K[X_t]$ is admissible. Finally, we verify that either Σ

is connected or each connected component of Σ intersects $K[X_t]$. As γ is a candidate solution at t', then either Σ is connected or each connected component of Σ intersects $K[X_{t'}]$. If Σ is not connected, then each connected component of Σ intersects $K[X_t]$ as $K[X_{t'}] \subset K[X_t]$. Thus, the condition holds. We conclude γ is a candidate solution at t. $\square$

The set $\mathbf{V}[\mathbf{t}']$ contains a set of annotated cell complexes equivalent to $\mathbf{D}[\mathbf{t}']$. Accordingly we set $\mathbf{V}[\mathbf{t}] = \mathbf{V}[\mathbf{t}']$

8.7 Forget Nodes

Let t be a forget node and t' the unique child of t. Recall that $X_t \sqcup \{w\} = X_{t'}$. Recall that $(K[X_{t'}])_2^w = \{\sigma \in (K[X_{t'}])_2 \,|\, w \in \sigma\}$ and the chain group $C_2^w(K[X_{t'}])$ is the chain group on $(K[X_{t'}])_2^w$. Lemma 5.8 tells us that $C_2(K_t) = C_2(K_{t'}) \oplus C_2^w(K[X_{t'}])$. We claim the following formula for $\mathbf{D}[\mathbf{t}]$.

$$\mathbf{D}[\mathbf{t}] = \left\{ \gamma = \gamma' + \gamma_w \;\middle|\; \begin{array}{l} (1)\ \gamma' \in \mathbf{D}[\mathbf{t}'],\ \gamma_w \in C_2^w(K[X_{t'}]). \\ (2)\ \text{The link } \mathrm{lk}_\Sigma v \text{ of each vertex } v \in K[X_t] \text{ has admissible link.} \\ (3)\ \text{The link of the forgotten vertex } \mathrm{lk}_\Sigma w \text{ is complete.} \\ (4)\ \Sigma \text{ is connected, or each connected component of } \Sigma \text{ intersects } K[X_t]. \end{array} \right\} \tag{8.1}$$

Lemma 8.9. *Assuming that* $\mathbf{D}[\mathbf{t}']$ *contains the set of candidate solutions at* t'*, Equation (8.1) correctly describes the set of candidate solutions at* $\mathbf{D}[\mathbf{t}]$*.*

Proof. We first prove that any candidate solution γ at t is contained in the right hand side of Equation (8.1). As $\gamma \in C_2(K_t)$, we can decompose $\gamma = \gamma' + \gamma_w$ where $\gamma' \in C_2(K_{t'})$ and $\gamma_w \in C_2^w(K[X_t])$. As $\gamma_w \in C_2^w(K[X_t])$, then conditions (2), (3), and (4) of Equation (8.1) hold because γ is a candidate solution at t, it remains to show that $\gamma' \in \mathbf{D}[\mathbf{t}']$.

We need to show that the conditions of candidacy holds for γ'. We first show that all vertices $v \in K_{t'} \cap K[X_{t'}]$ have admissible link in Σ'. Let $v \in K_{t'} \cap K[X_{t'}]$. As $X_{t'} = X_t \sqcup \{w\}$, then either $v \in X_t$ or $v = w$. The link $\mathrm{lk}_{\Sigma'} v \subset \mathrm{lk}_\Sigma v$, so in either case, it is easy to see that $\mathrm{lk}_{\Sigma'} v$ is admissible.

We next show that all vertices $v \in K_{t'} \setminus K[X_{t'}]$ have complete link in Σ'. Let $v \in K_{t'} \setminus K[X_{t'}]$, which implies $v \in K_t \setminus K[X_t]$, as $K_{t'} \subset K_t$ and $K[X_{t'}] \supset K[X_t]$.

Let $\tau \in \operatorname{lk}_\Sigma v$, and let σ be the coface shared between τ and v. By Lemma 8.1, $\sigma \in K_{t'} \setminus K[X_{t'}]$. So, $\tau \in \operatorname{lk}_{\Sigma'} v$. Therefore, $\operatorname{lk}_\Sigma v \subset \operatorname{lk}_{\Sigma'} v$. On the other hand, as $\Sigma' \subset \Sigma$, we have $\operatorname{lk}_\Sigma v \subset \operatorname{lk}_{\Sigma'} v$. Hence, $\operatorname{lk}_\Sigma v = \operatorname{lk}_{\Sigma'} v$. But, as γ is a candidate solution at t and $v \in K_t \setminus K[X_t]$, $\operatorname{lk}_{\Sigma'} v$ is complete.

Lastly, we verify that either Σ' is connected or each connected component of Σ' intersects $K[X_t']$. If $\Sigma_w = \emptyset$ then as $K[X_t] \subset K[X_{t'}]$ and $\Sigma = \Sigma'$, then this is true as γ is a candidate solution at t. If $\Sigma_w \neq \emptyset$, then we claim each connected component of Σ' must intersect $K[X_{t'}]$. Suppose Σ' is connected and $\Sigma' \cap K[X_{t'}] = \emptyset$. As $\Sigma_w \subset K[X_{t'}]$, then Σ' and Σ_w are disconnected and $\Sigma = \Sigma' \sqcup \Sigma_w$. Thus Σ is disconnected and has a connected component Σ that does not intersect $K[X_t]$ as $K[X_t] \subset K[X_{t'}]$, and γ is not a candidate solution at t, a contradiction. A similar argument shows that if Σ' is disconnected, then each connected component of Σ' intersects $K[X_{t'}]$. Thus $\gamma' \in \mathbf{D}[\mathbf{t}']$.

Now let $\gamma' \in \mathbf{D}[\mathbf{t}']$ and $\gamma_w \in C_2^w(K[X_{t'}])$ with $\gamma = \gamma' + \gamma_w$ that satisfies the conditions of Equation (8.1). We show $\gamma \in \mathbf{D}[\mathbf{t}]$.

We assume that each vertex $v \in K_t \cap K[X_t]$ has admissible link, so the first condition of candidacy holds. We also assume that either Σ is connected, or each connected component of Σ intersects $K[X_t]$, so the third condition of candidacy holds.

We now verify that each vertex $v \in K_t \setminus K[X_t]$ has complete link. We can decompose $K_t \setminus K[X_t]$ into $(K_t \setminus K[X_t]) \cap K[X_{t'}]$ and $(K_t \setminus K[X_t]) \setminus K[X_{t'}]$. Let $v \in (K_t \setminus K[X_t]) \cap K[X_{t'}]$. Note that $(K_t \setminus K[X_t]) \setminus K[X_{t'}] = K_t \setminus K[X_{t'}] = K_{t'} \setminus K[X_{t'}]$ as $K[X_t] \subset K[X_{t'}]$ and $K_t \setminus K_{t'} \subset K[X_{t'}]$. Any vertex $v \in (K_t \setminus K[X_t]) \setminus K[X_{t'}]$ therefore has complete link in Σ'. As $v \notin K[X_{t'}]$, then all cofaces of v are contained in $K_{t'} \setminus K[X_{t'}]$ by Lemma 8.1. In particular, no coface of v is contained in $\Sigma_w \subset K[X_{t'}]$. So $\operatorname{lk}_\Sigma v = \operatorname{lk}_{\Sigma'} v$ is complete. Alternatively, if $v \in (K_t \setminus K[X_t]) \cap K[X_{t'}]$ then $v = w$ as w is the only vertex in $K[X_{t'}] \setminus K[X_t]$. We assume $\operatorname{lk}_\Sigma w$ is complete. Each vertex in $K_t \setminus K[X_t]$ therefore has complete link in Σ. This proves $\gamma \in \mathbf{D}[t]$. $\square$

The complete algorithm for forget nodes is as follows. For each $\tilde{\Sigma}' \in \mathbf{V}[\mathbf{t}']$ and each chain $\gamma_w \in C_2^w(K[X_{t'}])$, add the annotated cell complexes $\tilde{\Sigma} = \tilde{\Sigma}' + \tilde{\Sigma}_w$. Verify that each vertex $v \in K_t \cap K[X_t]$ has admissible link using Lemma 8.6. Verify that w has complete link using Lemma 8.4. Remove all edges incident to w. Verify that either Σ is connected, or each connected component of Σ intersects $K[X_t]$ using Lemma 8.7. If $\tilde{\Sigma}$ satisfies the conditions of Equation (8.1), add $\tilde{\Sigma}$ to $\mathbf{V}[\mathbf{t}]$.

8.8 Join Nodes

Let t be a join node and let t', t'' be its two children. Recall by Lemma 5.8 that $C_2(K_t) = C_2(K_{t'}) \oplus C_2(K_{t''})$. We claim the following formula for $\mathbf{D}[\mathbf{t}]$.

$$\mathbf{D}[\mathbf{t}] = \left\{ \gamma = \gamma' + \gamma'' \;\middle|\; \begin{array}{l} (1)\ \gamma' \in \mathbf{D}[\mathbf{t}'], \gamma'' \in \mathbf{D}[\mathbf{t}''] \\ (2) \text{ Each vertex in } v \in K[X_t] \text{ has admissible link in } \Sigma. \\ (3)\ \Sigma \text{ is connected, or each connected component of } \Sigma \text{ intersects } K[X_t]. \end{array} \right\} \tag{8.2}$$

Lemma 8.10. *Assuming that* $\mathbf{D}[\mathbf{t}']$ *contains the set of candidate solutions at* t' *and* $\mathbf{D}[\mathbf{t}'']$ *contains the set of candidate solutions at* t''*, Equation (8.2) correctly describes the set of candidate solutions at* $\mathbf{D}[\mathbf{t}]$*.*

Proof. Let γ be a candidate solution at t. We show that γ is contained on the right hand side of Equation 8.2. By Lemma 7.2, $\gamma = \gamma' + \gamma''$ where $\gamma' \in C_2(K_{t'})$ and $\gamma'' \in C_2(K_{t''})$. Note that $\Sigma = \Sigma' \cup \Sigma''$ As γ is a candidate solution, Conditions (2) and (3) of Equation 8.2 are immediately satisfied. We now verify Condition (1), that γ' and γ'' are candidate solutions at t' and t'' respectively.

Let $v \in K_{t'} \setminus K[X_{t'}]$. We will show that $\operatorname{lk}_{\Sigma'} v$ is complete. We claim that $\operatorname{lk}_{\Sigma'} v = \operatorname{lk}_{\Sigma} v$. Clearly $\operatorname{lk}_{\Sigma'} v \subset \operatorname{lk}_{\Sigma} v$ as $\Sigma' \subset \Sigma$. We now show that $\operatorname{lk}_{\Sigma} v \subset \operatorname{lk}_{\Sigma'} v$. Let $\tau \in \operatorname{lk}_{\Sigma} v$. The simplex τ and v are incident to a common coface $\sigma \in \Sigma$. By Lemma 5.2, $\sigma \in K_{t'} \setminus K[X_{t'}]$. As $\sigma \in K_{t'} \setminus K[X_{t'}]$, then by Lemma 5.11, $\sigma \notin K_{t''} \setminus K[X_{t''}]$. As $\Sigma'' \subset K_{t''}$, we conclude that $\sigma \notin \Sigma''$, so $\sigma \in \Sigma'$. So $\operatorname{lk}_{\Sigma'} v \subset \operatorname{lk}_{\Sigma} v$ and $\operatorname{lk}_{\Sigma'} v = \operatorname{lk}_{\Sigma} v$. As $\operatorname{lk}_{\Sigma} v$ is complete, then so is $\operatorname{lk}_{\Sigma'} v$.

We next show that the link $\operatorname{lk}_{\Sigma'} v$ is admissible for each $v \in K_{t'} \cap K[X_{t'}]$. The link $\operatorname{lk}_{\Sigma'} v \subset \operatorname{lk}_{\Sigma} v$, so it is easy to see that $\operatorname{lk}_{\Sigma'} v$ is admissible as $\operatorname{lk}_{\Sigma} v$ is admissible.

Finally, we show that either Σ' is connected, or each connected component of Σ' intersects $K[X_{t'}]$. As γ is a candidate solution, then either Σ is connected, or each connected component of Σ intersects $K[X_t]$. If Σ is connected and does not intersect $K[X_t]$, then by Lemma 5.11, either $\Sigma \subset K_{t'} \setminus K[X_t]$ or $\Sigma \subset K_{t''} \setminus K[X_t]$. In either case, the condition holds. If Σ is disconnected, let Φ be a connected component of Σ. We claim $\Phi \cap \Sigma'$ intersects $K[X_{t'}]$. Indeed, this follows as $K[X_t] \cap K_t = K[X_{t'}] \cap K_{t'}$. So γ' is a candidate solution at t'. By a symmetric argument, γ'' is a candidate solution at t''

Now let $\gamma' \in \mathbf{D}[\mathbf{t}']$, $\gamma'' \in \mathbf{D}[\mathbf{t}'']$ and $\gamma = \gamma' + \gamma''$ such that the conditions of Equation

8.2 are satisfied. We prove that $\gamma \in \mathbf{D}[\mathbf{t}]$. Let $v \in K_t \setminus K[X_t]$. We will prove that $\mathrm{lk}_\Sigma v$ is complete. By Lemma 5.11, either $v \in K_{t'} \setminus K[X_t]$ or $v \in K_{t''} \setminus K[X_{t'}]$. Assume $v \in K_{t'} \setminus K[X_t]$. Then $v \in \Sigma' \setminus K[X_{t'}]$ and $\mathrm{lk}_{\Sigma'} v$ is complete. By Lemma 8.2, each coface of v and likewise each simplex in $\mathrm{lk}_{\Sigma'} v$ is contained in $K_{t'} \setminus K[X_{t'}]$. In particular, $\mathrm{lk}_{\Sigma''} v = \emptyset$ as $\Sigma'' \subset K_{t''}$. As $\mathrm{lk}_\Sigma v = \mathrm{lk}_{\Sigma'} v \cup \mathrm{lk}_{\Sigma''} v = \mathrm{lk}_{\Sigma'} v$, then $\mathrm{lk}_\Sigma v$ is complete. Similarly, each vertex in $K_{t''} \setminus K[X_t]$ has a complete link. By assumption, the link $\mathrm{lk}_\Sigma v$ of each vertex $v \in K_t \cap K[X_t]$ is admissible and either Σ is connected, or each connected component of Σ intersects $K[X_t]$. Thus, γ is a candidate solution at t. $\square$

The table entries $\mathbf{V}[\mathbf{t}']$ and $\mathbf{V}[\mathbf{t}'']$ contain annotated cell complexes equivalent to each candidate solution at t' and t''. To compute $\mathbf{V}[\mathbf{t}]$, we add together each cell complex $\tilde{\Sigma}' \in \mathbf{V}[\mathbf{t}']$ and $\tilde{\Sigma}'' \in \mathbf{V}[\mathbf{t}'']$ and check conditions (2) and (3) of Equation (8.2). We can check these conditions using Lemmas 8.6 and 8.7.

8.9 Analysis

We now analyze the running time of our algorithm. We first bound the number of annotated cell complexes $\tilde{\Sigma} \in \mathbf{V}[\mathbf{t}]$ at a node t.

Lemma 8.11. *Let t be a node in the tree decomposition. There are $(k+gb)^{O(k^2)}$ annotated cell complexes in $\mathbf{V}[\mathbf{t}]$, where b is the number of connected components of B and g is the genus of Ψ.*

Proof. There can be $O(k^2)$ edges, real and dummy, in any annotated cell complex in $\tilde{\Sigma} \in \mathbf{V}[\mathbf{t}]$. An annotated cell complex is characterized completely as a bijection between a subset of these edges. (This subset may be empty, as is the case of an annotated cell complex which is a single empty face.) The bijection defines the successor of each edge, and the orbits of this bijection define the boundaries of the annotated cell complex. There are $2^{O(k^2)}$ subsets of edges in X_t, so there are $2^{O(k^2)}O((k^2)!) = k^{O(k^2)}$ such bijections. For each bijection, we must group each boundary into faces. Each boundary contains an edge, or there are no edges in the cell complex by Lemma 6.8. There are thus $O(k^2)$ boundaries in an annotated cell complex and $O(k^2)^{O(k^2)} = k^{O(k^2)}$ ways to partition boundaries into faces. There are thus $k^{O(k^2)}$ cell complexes at a node t, not accounting for annotations.

Each face has an annotation containing its genus, number of boundary components, and orientability. If g is the input genus of the surface Ψ, we can discard any candidate solution with a face that exceeds genus g. If b is the number of components of connected components B, then no face will ever have more than b boundary components. There are $k^{O(k^2)}$ annotated cell complexes in an entry $\mathbf{V}[\mathbf{t}]$. Each cell complex has $O(k^2)$ faces and each face has one of $O(gb)$ annotations. In total, there are $(k+gb)^{O(k^2)}$ annotated cell complexes in $\mathbf{V}[\mathbf{t}]$. $\square$

We claim there can be at most $O(n^4)$ possible annotations for any given face. We prove this with the following lemma.

Lemma 8.12. *Let S be a connected, combinatorial surface. Let n be the number of vertices in S, g the genus of S, and b the number of boundary components of S. Then $g, b = O(n^2)$.*

Proof. Let n, m, and l be the number of vertices, edges, and triangles of S respectively. The Euler characteristic of the surface is $n - m + l = 2 - 2g - b$ if S is orientable and $n-m+l = 2-g-b$ if S is non-orientable. The bound $g+b \leq m-n-l+2$ holds in either case. As each triangle is incident to three edges and each edge is incident to at least one triangle, it follows that $m \leq 3l$. Therefore, $g+b \leq m-n-l+2 \leq m-n-(1/3)m+2$. As $m = O(n^2)$, it follows that $g, b = O(n^2)$. $\square$

Combining the previous two lemmas, we see that in the worst case there are $k^{O(k^2)}n^{O(k^2)}$ annotated cell complexes in an entry $\mathbf{V}[\mathbf{t}]$.

Theorem 8.13. *Let K be a 2-dimensional simplicial complex such that the 1-skeleton of K has treewidth k. Let Ψ be a connected, compact surface of genus g. Let $B \subset K$ be a disjoint union of simple cycles. There is an algorithm to determine if there is a subcomplex $S \subset K$ homeomorphic to Ψ with boundary B in $(k+gb)^{O(k^2)}n = (k+n)^{O(k^2)}n$ time, where b is the number of connected components of B.*

Proof. There are at most $O(kn)$ nodes in a nice tree decomposition, so we just need to prove each node can be processed in $k^{O(k^2)}gb^{O(k^2)}$ time.

Leaf nodes can be processed in $O(1)$ time.

Introduce nodes require no work as $\mathbf{V}[\mathbf{t}] = \mathbf{V}[\mathbf{t}']$ so processing an introduce node takes $O(1)$ time.

Forget nodes add each of the $2^{O(k^2)}$ chains in $C_2^w(K[X_t])$ to each of the $k^{O(k^2)}gb^{O(k^2)}$ cell complexes in the table entry of the child $\mathbf{V}[\mathbf{t}']$ and verify that the cell complex is a candidate solution at t. It takes poly(k) to verify that an annotated cell complex is a candidate solution: poly(k) to check that the link of each vertex is either admissible or complete, and $O(k^2)$ time to check that if the cell complex has an empty face, that this is the only face of the cell complex. We also have to remove all edges incident to the forgotten vertex w from an annotated cell complex. There are $O(k)$ edges incident to w. Removing an edge a takes $O(k^2)$ time: $O(k^2)$ to find the other appearance of a and $O(k^2)$ to modify the faces containing a. Processing a forget node takes $k^{O(k^2)}gb^{O(k^2)}$ time in total.

Join nodes attempt to combine each of the cell complexes in the table entry $\mathbf{V}[\mathbf{t}']$ and $\mathbf{V}[\mathbf{t}'']$ for its two children. Attempting to join two cell complexes takes poly(k) time; to check that each vertex still has admissible link and that if the annotated cell complex has an empty face, that this is the only face. There are $k^{O(k^2)}gb^{O(k^2)}$ candidates at each of the two children, so processing a join node takes $k^{O(k^2)}gb^{O(k^2)}$ time. □

8.10 Improving the Running Time, Proof of Theorem 1.4

We now sketch a way to improve the running time of our algorithm for 2DBS to obtain Theorem 1.4. In our algorithm, we store the genus, number of boundary components, and orientability of each face in the annotated cell complex; however, as the surface we are looking for is connected, then this level of detail is unnecessary. We can instead store a global genus, number of boundary components, and orientability of an annotated cell complex. A disconnected candidate solution will either become connected at a later stage in our algorithm, in which case, the genus and number of boundary components of the cell complex will be the sum of the genus and number of boundary components of each face anyway, or a candidate solution will remain disconnected until the end of the algorithm, in which case we will discard it. Moreover, as the boundary B is given, we don't need to store the number of boundary components either, as a solution has a predetermined number of boundary components.

In this new algorithm, we no longer distinguish surfaces up to homeomorphism. For example, if cell complexes C and C' have two identical faces with both faces having genus 1 in C and one face having genus 2 and the other genus 0 in C' as in Figure 8.8, then C and C' would not be distinguished by the new algorithm, even though they are not homeomorphic. At any node, there will thus be $k^{O(k^2)}g$ annotated cell complexes. The algorithm is the same, except updates are always made to the global genus instead of the face's genus.

Figure 8.8: The candidate solutions on the left and the right are not homeomorphic but would not be distinguished by our algorithm.

8.11 Variants of 2DBS

The problem 2DBS is phrased to be analogous to OBCP; we want to find a surface that bounded by a set of cycles. While this suits our paper well, there are other contexts in which one might want to find a subcomplex homeomorphic to a surface. In this section we briefly discuss how our algorithm can be adapted to several variants of 2DBS.

2DBS is an existence problem: Does there exist a surface homeomorphic to Ψ bounded by B? There are also optimization problems similar to 2DBS: What is the minimal genus surface that bounded by B? What is the minimal area surface bounded by B? Our algorithm for 2DBS can solve these problems as well. Our algorithm for 2DBS solves the minimal genus surface as the table entry at the root $\mathbf{V}[r]$ contains an annotated cell complex homeomorphic to each surface bounded by B. We can just search this entry for the minimal genus surface. Our algorithm can also be adapted to find the minimum area surface bounded by B. For each annotated cell complex in a table entry $\mathbf{V}[\mathbf{t}]$, store the number of triangles of the minimum candidate solution homeomorphic to this annotated cell complex. Whenever two candidate solutions correspond to the same annotated cell complex, take the candidate solution that uses the fewer number of triangles. At the end of the algorithm, we can search $\mathbf{V}[\mathbf{r}]$ for the bounded surfaces that uses the fewest number of triangles.

One might ask whether or not a simplicial complex has a subcomplex homeomorphic to a surface with a certain number of boundary components rather than explicitly specifying the boundary B. In this case, the input surface would be a compact, connected surface Ψ specified by its genus g, number of boundary components b, and orientability. For this problem, we allow the link of any vertex to be complete if it is either a simple path or a simple cycle, as opposed to our algorithm for 2DBS which decides in advance whether the link of each vertex should be a simple cycle or a simple path. In our algorithm for 2DBS, we can bound the number of boundary dummy edges incident to any single vertex as any vertex will only be incident to two edges in B that might be contracted to boundary dummy edges. In this variant, we cannot make this argument to bound the number of boundary dummy edges as a vertex might be incident to many edges that contract to boundary dummy edges. We therefore need to hardcode the requirement that any vertex can be incident to at most two boundary dummy edges to bound the size of our candidate solutions. This requirement does not preclude any potential solutions, because if a vertex is incident to more than two dummy edges, its link can never be extended to be a simple cycle or simple path as no triangles are added incident to boundary dummy edges.

The algorithm for this variant is slower than the algorithm for 2DBS as we can no longer ignore counting boundary components. For each cell complex, there can now be $O(gb) = O(n^4)$ different annotations. The algorithm for this variant runs in $k^{O(k^2)}gbn = k^{O(k^2)}O(n^5)$ time.

8.12 Open Problems

Our algorithm for 2DBS only searches for connected surfaces. It is unclear if finding a disconnected surface is still fixed-parameter tractable. The difficulty in finding a disconnected surface is that at any node t, an annotated cell complex can have many connected components that do not intersect $K[X_t]$. These correspond to empty faces in an annotated cell complex $\tilde{\Sigma}$. As these face have no edges, we were not able to find a way to bound the complexity of $\tilde{\Sigma}$ by a function of the treewidth.

The problem 2DBS is inspired by problems on knots. Knot problems look for *normal surfaces* in a 3-manifold, surfaces not contained in the 2-skeleton that are allowed to intersect tetrahedra. It is unclear whether there are fixed-parameter tractable algorithms with respect to the treewidth of the 1-skeleton for finding normal surfaces.

Bibliography

[1] Ian Agol, Joel Hass, and William P. Thurston. The computational complexity of knot genus and spanning area, 2002.

[2] L.V. Ahlfors and L. Sario. *Riemann Surfaces*. Princeton mathematical series. Princeton University Press, 2015.

[3] 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.

[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.

[5] Hans Bodlaender, Pål G. Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $O(c^k n)$ 5-approximation algorithm for treewidth, 2013.

[6] Glencora Borradaile, William Maxwell, and Amir Nayyeri. Minimum Bounded Chains and Minimum Homologous Chains in Embedded Simplicial Complexes. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[7] 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.

[8] Benjamin Burton and Rodney Downey. Courcelle's theorem for triangulations. *Journal of Combinatorial Theory, Series A*, 146, 03 2014.

[9] 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, Mar 2016.

[10] 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*, Jan 2013.

[11] Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K. Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In Fedor V. Fomin and Petteri Kaski, editors, *Algorithm Theory – SWAT 2012*, pages 189–200, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[12] J. Carvalho, Mikael Vejdemo-Johansson, Danica Kragic, and Florian Pokorny. An algorithm for calculating top-dimensional bounding chains. *PeerJ Computer Science*, 4:e153, 05 2018.

[13] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, SCG '09, page 377–385, New York, NY, USA, 2009. Association for Computing Machinery.

[14] Erin Wolf Chambers and Mikael Vejdemo-Johansson. Computing minimum area homologies. *Computer Graphics Forum*, 34(6):13–21, 2014.

[15] Chao Chen and Daniel Freedman. Hardness results for homology localization. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, page 1594–1604, USA, 2010. Society for Industrial and Applied Mathematics.

[16] David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic Optimal Homologous Chains and Applications to Point Cloud Triangulations. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[17] Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12 – 75, 1990.

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

[19] Tamal K. Dey. Efficient algorithms to detect null-homologous cycles on 2-manifolds. *International Journal of Computational Geometry & Applications*, 07(03):167–174, 1997.

[20] Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. *SIAM Journal on Computing*, 40(4):1026–1044, 2011.

[21] Nathan M. Dunfield and Anil N. Hirani. The least spanning area of a knot and the optimal bounding chain problem. In *Proceedings of the 27th Annual Symposium on Computational Geometry, SCG'11*, Proceedings of the Annual Symposium on Computational Geometry, pages 135–144, July 2011. 27th Annual ACM Symposium on Computational Geometry, SCG'11 ; Conference date: 13-06-2011 Through 15-06-2011.

[22] Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the 2011 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1166–1176, 2011.

[23] Jean Gallier and Dianna Xu. *A Guide to the Classification Theorem for Compact Surfaces*. Springer Berlin Heidelberg, 2013.

[24] Ofir (https://math.stackexchange.com/users/2125/ofir). A clique in a tree decomposition is contained in a bag. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/1228108 (version: 2015-04-10).

[25] Sergei Ivanov. computational complexity. MathOverflow. `https://mathoverflow.net/questions/118357/computational-complexity`.

[26] Marc Lackenby. Some conditionally hard problems on links and 3-manifolds, 2017.

[27] N.J. Wildberger. An algebraic ZIP proof of the classification — Algebraic Topology — NJ Wildberger. YouTube, November 2011. `https://www.youtube.com/watch?v=-GJs7_NdLm8`.

APPENDICES

Appendix A: Proofs of Lemma 7.6, 7.7, and 7.8

Proof of Lemma 7.6. We will use our equivalence-preserving moves to show $(\overline{aba^{-1}b^{-1}XY}) = (\overline{efe^{-1}f^{-1}YX})$.

$$\begin{aligned}
(\overline{aba^{-1}b^{-1}XY}) &= (\overline{a^{-1}b^{-1}XYab}) && \text{by (4)}\\
&= (\overline{a^{-1}cYaXc^{-1}}) && \text{by Lemma 7.5}\\
&= (\overline{Xc^{-1}a^{-1}cYa}) && \text{by (4)}\\
&= (\overline{Xc^{-1}d^{-1}Ycd}) && \text{by Lemma 7.5}\\
&= (\overline{dXc^{-1}d^{-1}Yc}) && \text{by (4)}\\
&= (\overline{dXeYd^{-1}e^{-1}}) && \text{by Lemma 7.5}\\
&= (\overline{e^{-1}dXeYd^{-1}}) && \text{by (4)}\\
&= (\overline{e^{-1}f^{-1}YXef}) && \text{by Lemma 7.5}\\
&= (\overline{efe^{-1}f^{-1}YX}) && \text{by (4)}
\end{aligned}$$

□

Proof of Lemma 7.7. We first prove the sub-lemma that $(\overline{aaXY}) = (\overline{bYbX^{-1}})$.

$$\begin{aligned}
(\overline{aaXY}) &= (\overline{YaaX}) && \text{by (4)}\\
&= (\overline{Yab}) + (\overline{b^{-1}aX}) && \text{by (2)}\\
&= (\overline{Yab}) + (\overline{X^{-1}a^{-1}b}) && \text{by (3)}\\
&= (\overline{bYa}) + (\overline{a^{-1}bX^{-1}}) && \text{by (4)}\\
&= (\overline{bYbX^{-1}}) && \text{by (2)}
\end{aligned}$$

We now use this sub-lemma to prove that $(\overline{aaXY}) = (\overline{ddYX})$.

$$
\begin{aligned}
(\overline{aaXY}) &= (\overline{bYbX^{-1}}) && \text{by the sub-lemma} \\
&= (\overline{XcY^{-1}c}) && \text{by (3), where } c = b^{-1} \\
&= (\overline{cXcY^{-1}}) && \text{by (4)} \\
&= (\overline{ddYX}) && \text{by the sub-lemma}
\end{aligned}
$$

$\square$

Proof of Lemma 7.8. Rearranging XY to YX follows immediately from move (6). As a appears only in the boundary of a, we can replace a with new edges edges ef by move (1) without changing the boundary of any other faces. We can then replace ef with a new edge d by move (1). $\square$

Appendix B: Proofs of Lemma 8.4, 8.6, and 8.7

We now prove Lemmas 8.4, 8.6, and 8.7. We need to prove a few lemmas first, though.

Lemma B.1. *If a vertex $v \in K[X_t] \cap \Sigma$, there is an edge incident to v in $\tilde{\Sigma}$.*

Proof. When a vertex v is first added to $\tilde{\Sigma}$, there is an edge incident to v. The only way v would disappear from $\tilde{\Sigma}$ is if each vertex w that shared an edge with v was forgotten.

If $v \notin B$, then each edge incident to v would have to incident to two faces an no edge incident to v is in B. It follows at some point during the algorithm the set of edges entering v formed a cyclic sequence of successors. When the last vertex adjacent to v was forgotten, the set of edges entering v formed the cyclic sequence of edges $(\overline{a})$. At this point, a would be replaced with an interior dummy edge. This dummy edge is only removed when v is forgotten.

If $v \in B$, then each edge incident to v was incident to two faces or v was incident to an edge in B. In the first case, an edge incident to v would be replaced with an interior dummy edge. In the second case, an edge incident to v would be replaced with a boundary dummy edge. which would not be removed until v was removed. □

Lemma B.2. *Each face A in an annotated cell complex $\tilde{\Sigma}$ corresponds to a 2-chain $\gamma_A \in C_d(K_t)$.*

Proof. We can see this by induction. If $A \in K$, then this is obviously true. If $A \notin K$, then A is the merge of two smaller faces that shared an edge, each of which corresponds to a 2-chain. □

Lemma B.3. *A boundary dummy edge $\{w_1, w_2\}$ replaces a simple segment $(w_1, u_1, \ldots, u_{l-1}, w_2)$ of B.*

Proof. We can prove this induction on the length l of the segment. A dummy edge $\{w_1, w_2\}$ always replaces a pair of edges $\{w_1, w\}$ and $\{w, w_2\}$ when w is forgotten. If $\{w_1, w\}$ and $\{w, w_2\}$ are both real, then $\{w_1, w_2\}$ replaces the segment (w_1, w, w_2) and the statement is true for $l = 2$. If either $\{w_1, w\}$ or $\{w, w_2\}$ are dummy edges, the dummy edge replace paths $B_1 = (w_1, \dots, w)$ and $B_2 = (w, \dots, w_2)$ of B. We claim the paths B_1 and B_2 are disjoint except possibly at their endpoints. Any non-endpoint vertex $u \in B_1$ is incident to two edges e_1 and e_2 in B. If $u \in B_1 \cap B_2$, then e_1 must be incident to two triangles in f, a contradiction as an endpoint of e_1 had complete link when its was forgotten. The concatenation of B_1 and B_2 is therefore a simple path. $\square$

Let A be a face in the cell complex. We would like to say that $\mathrm{lk}_A\, w$ - the portion of $\mathrm{lk}_\Sigma\, w$ contained in $\mathrm{cl}\,\gamma_A$ - is a collection of simple paths or a simple cycle. We prove a lemma in this direction.

Lemma B.4. *Let A_1 be a face in an annotated cell complex $\tilde{\Sigma}$. Let (w_1, w, w_2) be a segment on the boundary of A_1. There is a simple path P_1 in $lk_{A_1}\, w$ such that*

1. *If $\{w_1, w\}$ is a real edge, then one endpoint of P_1 is w_1.*
2. *If $\{w_1, w\}$ is an interior dummy edge, the endpoints of P_1 are equal.*
3. *If $\{w_1, w\}$ is a boundary dummy edge that replaces a segment $B_1 = \{w_1, u_{l-1} \dots, u_1, w\}$ of B, then one endpoint of P_1 is u_1.*
4. *If w appears in another segment (w_3, w, w_4) on the boundary of a face A_2 in $\tilde{\Sigma}$, then the path P_2 in $lk_{A_2}\, v$ and P_1 are disjoint.*

Proof. We prove that if the lemma is true on an annotated cell complex $\tilde{\Sigma}$, then it is true after removing an edge from $\tilde{\Sigma}$. The condition of the lemma is defined facewise, so adding together two triangle-disjoint annotated cell complexes won't violate the condition.

An annotated cell complex defined by our algorithm is initially a collection of triangles. The base case of our proof is a single triangle $\{w_1, w, w_2\} \in K$. The link of w in $\{w_1, w, w_2\}$ is the simple path (w_1, w_2). If w appears in a distinct triangle $\{w_3, w, w_4\}$, the link of w in $\{w_3, w, w_4\}$ is the simple path (w_3, w_4). As the triangles are distinct, the paths (w_1, w_2) and (w_3, w_4) can share at most one endpoint.

Now assume the lemma is true for an annotated cell complex $\tilde{\Sigma}$. We will show the lemma is true after removing a real edge $\{w, u\}$ from two (possibly the same) faces. Let (w_1, w, u) and (u, w, w_2) be segments on the boundary of faces A_1 and A_2. By the lemma, there is a path P_1 in $\mathrm{lk}_{A_1}\, w$ with u as an endpoint. Likewise, there is a path P_2 in $\mathrm{lk}_{A_2}\, w$ with u as an endpoint. Merging the faces A_1 and A_2 at $\{w, v\}$ will create a face

A_3 with (w_1, w, w_2) on its boundary. The paths P_1 and P_2 are disjoint except possibly at their endpoints, so the concatenation of P_1 and P_2 is a simple path with endpoints w_1 and w_2. Merging P_1 and P_2 creates a new path with u in its interior. The vertex u does not appear in any other path P_3 in $\operatorname{lk}_\Sigma w$. If it did, it would have to be the endpoint of P_3 as $\tilde{\Sigma}$ satisfied the condition of the lemma before removing $\{w, u\}$; however, such an annotated cell complex would be discarded by our algorithm as $\{w, u\}$ appears three times. The lemma is true for any other segment (x, y, z) of the boundary of A_1 (say) because x and z were connected by a path $P \subset \operatorname{lk}_{A_1} y \subset \operatorname{lk}_A y$.

We now verify this is true after adding interior dummy edges. If the segment of the boundary with the interior dummy edges is (w_1, w, w_1), then previously w was on a segment of the boundary (u, w, u) where (w, u) is a real edge. So by assumption, there was a path P_1 in $\operatorname{lk}_A w$ with both endpoints equal to u.

We now verify this is true after merging two edges into a boundary dummy edge. Let (w_1, w, u) and (w, u, w_2) be segments of a face f. We replace the edges $\{w, u\}$ and $\{u, w_2\}$ with a dummy edge $\{w, w_2\}$. This replacement does not change the triangles that compose A so $\operatorname{lk}_A w$ is the same before and after the replacement. If $\{w, u\}$ replaced a segment $(w, u_1, \ldots, u)$ of β, then $\{w, w_2\}$ replaces some segment of the boundary $(w, u_1, \ldots, u, \ldots, w_2)$ and the lemma holds.

$\square$

Proof of Lemma 8.4. We begin with the case that $v \notin B$. No edges enter v if and only if $\operatorname{lk}_\Sigma v$ is empty. If no edges enter v, then by Lemma B.1 $v \notin \Sigma$ and $\operatorname{lk}_\Sigma v$ empty. Conversely, if $\operatorname{lk}_\Sigma v = \emptyset$, then obviously no edges in $\tilde{\Sigma}$ enter v.

Now assume there are edges that enter v. Assume the edges entering v form a cyclic sequence of successors $(\overline{a_1 \ldots a_k})$ with $a_i = \{v, w_i\}$. A pair of consecutive edges $a_i a_{i+1}$ corresponds to a segment (w_i, v, w_{i+1}) on the boundary of a face f_i. If $k = 2$ and a_1 and a_2 are interior dummy edges, then this is immediately true by Lemma B.4. So assume the edges a_i and a_{i+1} are real. Then w_i and w_{i+1} are connected by a path P_i in $\operatorname{lk}_F v$. Moreover, the two paths P_i and P_j are disjoint, except possibly at their endpoints. The path P_i shares endpoints with P_{i-1} and P_{i+1}, so the concatenation of the paths forms a simple cycle.

Now assume that $\operatorname{lk}_\Sigma v$ is a simple cycle. By Proposition 7.1, the edges entering v in Σ form a cyclic sequence of successors. The annotated cell complex $\tilde{\Sigma}$ is obtained from Σ by removing edges $e \notin K[X_t]$. We will show that each of these moves preserves the property of the lemma. Let $(a_1 \dots a_k)$ be the cyclic sequence of successors entering v. Removing an edge not incident to v does not change $(\overline{a_1 \dots a_k})$. If we do remove an edge a_i, by Lemma 8.5 the edges entering v form a cyclic sequence of successors $(\overline{a_1 \dots a_{i-1} a_{i+1} \dots a_k})$. If there is a single edge a entering v, then forgetting a results in the creation of an interior dummy edge d. As discussed above, an interior dummy edge forms a cyclic sequence of successors $(\overline{d})$.

The proof is nearly identical if $v \in B$, except we use Lemma B.4 to show that P_1 and P_k share a single endpoint with other paths and the other endpoints of these paths are v's neighbors in β. Likewise, the edges entering v form a sequence of successors $(a_1 \dots a_k)$ in Σ, and forgetting an edge a_i maintains this property. Forgetting the edges a_1 or a_k creates a dummy edge d entering v that had the same successors as a_1 or a_k. □

Proof of Lemma 8.6. The proof of this lemma is almost identical to the proof of Lemma 8.4. □

Proof of Lemma 8.7. Let Σ such that Σ has a connected component that does not intersect $K[X_t]$. Let $\Phi \subset \Sigma$ be this connected component. As no edges in Φ are in $K[X_t]$, then no face of Φ would contain any edges in $\tilde{\Sigma}$. Thus, $\tilde{\Sigma}$ would have an empty face.

Assume $\tilde{\Sigma}$ has an empty face (). Each face has edges when added to Σ, so all the edges on () must have been removed. The empty face () does not share an edge with any other face in $\tilde{\Sigma}$ as removing an edge shared by two faces merges the two faces that contain this edge. Also, () does not share any vertices with any other face of $\tilde{\Sigma}$. Any vertex v in () was forgotten; otherwise, by Lemma B.1, there would be an edge in $\tilde{\Sigma}$ containing this vertex. Forgetting a vertex merges all faces incident to this vertex into a single face, so no other face contains this vertex. □