Computing Projective Implicit Representations from Poset Towers

Tamal K. Dey
Department of Computer Science, Purdue University, West Lafayette, USA

Florian Russold
Institute of Geometry, Graz University of Technology, Graz, Austria

Abstract

A family of simplicial complexes, connected by simplicial maps and indexed by a finite poset P, is called a *poset tower*. The concept of poset towers subsumes classical objects of study in the persistence literature, such as multi-parameter filtrations, zigzag filtrations, and one-parameter simplicial towers, while also allowing arbitrary finite posets and arbitrary simplicial maps. The homology of a poset tower gives rise to a P-persistence module. To compute this homology globally over P, in the spirit of the persistence algorithm, we consider the homology of the chain complex segment of P-persistence modules, $C_{\ell\text{-}1} \xleftarrow{\partial_\ell} C_\ell \xleftarrow{\partial_{\ell+1}} C_{\ell+1}$, induced by the simplices of the poset tower. Contrary to the case of one-critical multi-filtrations, the chain modules C_ℓ of a poset tower are not necessarily projective and can have a complicated structure. In this work, we address the problem of computing a replacement of such a chain complex segment by projective modules and P-graded matrices, resulting in what we call a projective implicit representation (PiRep), which preserves the homology. Such a representation plays the role of the graded boundary-matrix representation in the classical persistence algorithm: it converts the simplicial data of a poset tower into algebraic input on which persistent homology can be computed globally over P. In particular, a PiRep can be used as input to existing algorithms for computing minimal presentations of persistent homology. We give an efficient algorithm to compute a PiRep from a poset tower. Our algorithm constructs degreewise minimal presentations and asymptotically minimal second terms of projective resolutions of the chain modules C_ℓ, lifts the boundary maps ∂_ℓ to these resolutions, and assembles this data into a PiRep using an additional correction term. The method is tailored to the chain complexes induced by poset towers and computes the required algebraic data combinatorially by exploiting their special structure, avoiding general-purpose algebraic reduction. In the context of poset towers, it is fully general and can serve as a foundation for developing efficient algorithms on specific posets.

Keywords and phrases Posets, Towers, Chain complexes, Persistence, Presentations, Resolutions

Digital Object Identifier 10.57717/cgt.v5i1.94

Funding *Tamal K. Dey*: Partially supported by NSF grants CCF-2437030 and DMS-2301360
Florian Russold: Partially supported by Austrian Science Fund (FWF) grants P 33765-N and W1230

1 Introduction

Persistent homology is a tool to quantify topological information of a space. After its introduction in [17], persistent homology has become a broad and very active field of research, with numerous applications in data science [15, 16, 20]. Persistent homology tracks topological features of a space evolving over some indexing poset P. The spaces most commonly considered in applications are simplicial complexes. We call a family of simplicial complexes and simplicial maps, subject to commutativity relations, indexed by a finite poset P, a *poset tower*. A poset tower can be succinctly described as a functor $K\colon P \to \mathbf{SCpx}$ from the indexing poset P to the category of simplicial complexes $\mathbf{SCpx}$. Figure 1 shows examples of common indexing posets and Figures 2-6 show examples of poset towers. The ℓ-th homology of a poset tower, $H_\ell(K) \coloneqq H_\ell \circ K$, gives rise to a P-persistence module, i.e., a functor $P \to \mathbf{Vec}$ from P to the category of vector spaces. It can be defined as the homology

© Tamal K. Dey and Florian Russold
licensed under Creative Commons License CC-BY 4.0

of a chain complex segment $C_{\ell+1}(K) \xrightarrow{\partial_{\ell+1}} C_\ell(K) \xrightarrow{\partial_\ell} C_{\ell-1}(K)$ of P-persistence modules. But for a general poset tower K, the persistence modules $C_\ell(K) \coloneqq C_\ell \circ K \colon P \to \mathbf{Vec}$, describing the evolution of simplicial ℓ-chains in K, can have a complicated structure, and the boundary maps ∂_ℓ cannot be globally represented by matrices. The goal of this work is to compute a chain complex segment $D_{\ell+1} \xrightarrow{d_{\ell+1}} D_\ell \xrightarrow{d_\ell} D_{\ell-1}$ such that each D_ℓ is a *projective module*, the maps d_ℓ are represented by P-graded matrices, and $H_\ell(K) \cong \ker(d_\ell)/\operatorname{im}(d_{\ell+1})$. We call such a representation a *projective implicit representation*, PiRep for short, of $H_\ell(K)$. We adopt this notion from the *free implicit representations* (FiRep) used in multi-parameter persistence [22, 25], but use projective modules because we work in the setting of finite indexing posets. Given a PiRep of $H_\ell(K)$, we can compute a minimal presentation of the homology module $H_\ell(K)$ using algorithms discussed in [6].

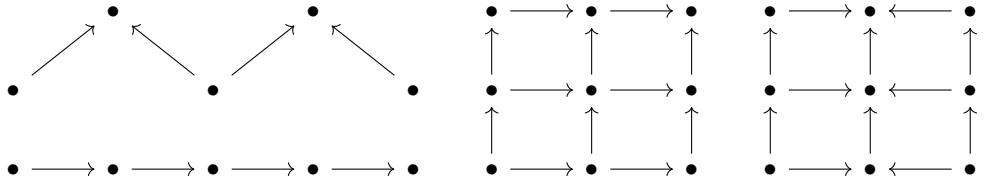


Figure 1 Common indexing posets: (lower left) one-parameter persistence, (upper left) zigzag persistence, (center) two-parameter persistence, (right) two-parameter zigzag persistence.

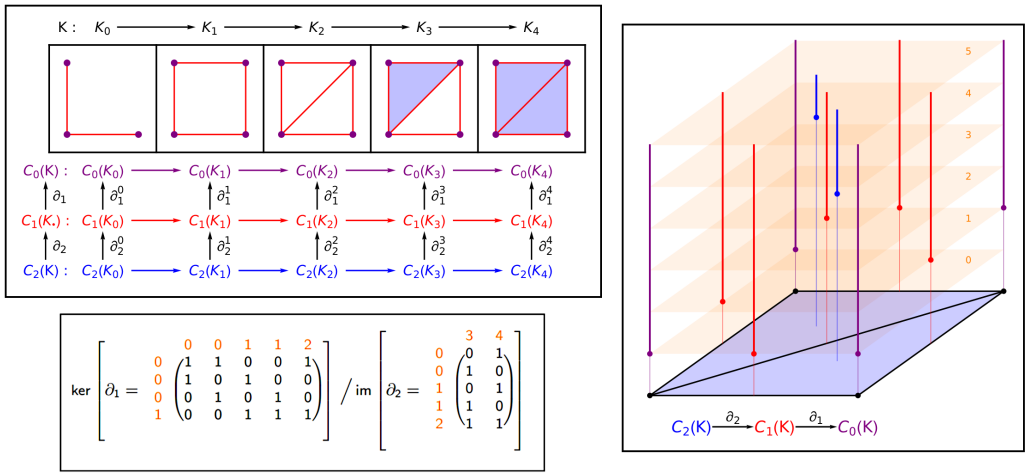


Figure 2 One-parameter persistent homology pipeline. Upper left: A filtered simplicial complex together with its algebraic representation by the simplicial chain complex. Right: The simplicial chain spaces as projective interval modules, represented by colored vertical lines, corresponding to the simplices in the filtration. Lower left: The representation of the boundary maps by graded matrices. The orange numbers on the right and in the lower left represent the grades in the filtration.

From filtered simplicial complexes to poset towers. Originally, persistent homology was introduced for filtered simplicial complexes (Figure 2 upper left), which are described by functors $K \colon P \to \mathbf{SCpx}$, where $P = \{1, \dots, n\}$ with the standard order, and $K(x \leq y)$ is an inclusion for all $x \leq y \in P$. A cornerstone of the field of persistent homology is the so-called persistence algorithm. It efficiently computes an invariant, called the barcode, completely capturing the persistent homology of a filtered simplicial complex [17, 34]. It is based on the fact that each chain module $C_\ell(K)$ (Figure 2 upper left) is projective and can be described

as a sum of upset interval modules[1], corresponding to the ℓ-simplices that start at the point where the simplex appears first in the filtration (Figure 2 right). The significance of $C_\ell(K)$ being projective is that, in this case, we can represent the boundary maps ∂_ℓ by graded matrices (Figure 2 lower left). This matrix representation can be read off immediately from the input and allows us to compute the persistent homology as $\ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$ via matrix operations on these graded matrices.

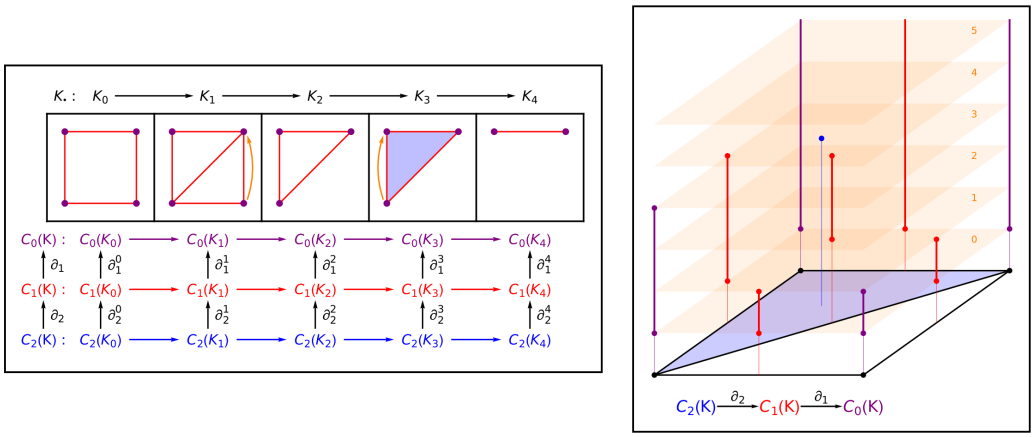


Figure 3 Left: A simplicial tower with its algebraic representation by a simplicial chain complex. The orange arrows represent simplicial collapses. Right: The chain spaces represented by interval modules corresponding to the simplices. Contrary to the case of filtrations, an interval can end if the corresponding simplex dies in the tower.

Since its introduction, the classical one-parameter persistent homology of filtered spaces has been generalized to various other settings and indexing posets (see Figure 1). Simplicial towers [11] generalize filtrations by allowing the maps $K(x \leq y)$ to be arbitrary simplicial maps (see Figure 3). In this situation, $C_\ell(K)$ is also a sum of interval modules, where the intervals correspond to the simplices in the tower (Figure 3 right). The difference from the case of filtrations is that intervals can end if a simplex gets collapsed. In other words, the chain modules $C_\ell(K)$ can have relations (killing simplices) and are no longer projective. To compute the persistent homology via graded matrices, we first have to construct such a matrix representation from K. This can be achieved by either converting it to a filtration via geometric methods [11, 23], or via algebraic methods [13]. We can then compute the barcode using the standard persistence algorithm. Similar statements hold for zigzag persistence [7, 12], which generalizes the indexing poset by allowing the arrows to point in both directions.

In multi-parameter persistence [3, 5, 8, 27], we increase the dimension of the indexing poset to obtain d-dimensional grid-posets. For $d > 1$, the persistent homology can no longer be described by a complete discrete invariant, like a barcode. In this case, a minimal presentation of $H_\ell(K)$ has been proposed as an alternative in the topological data analysis community [25]. For two parameters, efficient algorithms for computing this minimal presentation have been proposed in [19, 22, 25]. Beyond two parameters, an algorithm has been suggested in [4], though full details are missing at the moment.

These algorithms require, as input, a free/projective implicit representation of the chain

[1] An interval persistence module is one-dimensional (point-wise) on a convex connected subposet of the indexing poset P and zero elsewhere, with identity maps between comparable elements in the support.

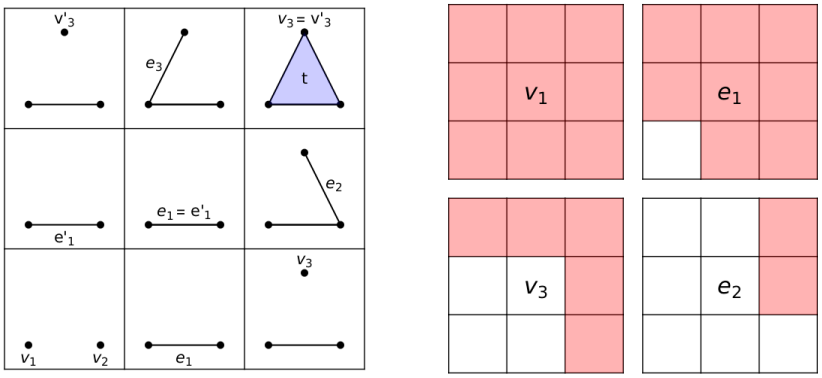


Figure 4 Left: a bifiltered simplicial complex with simplex labels shown at their generating grades. Here v_3 and v_3' represent two copies of the same vertex that get identified in the upper right. Right: The interval modules corresponding to some of the simplices.

complex segment $C_{\ell+1}(K) \xrightarrow{\partial_{\ell+1}} C_\ell(K) \xrightarrow{\partial_\ell} C_{\ell-1}(K)$. It is conceivable that understanding structural and computational aspects of the chain modules $C_\ell(K)$ may become useful for computing minimal presentations of $H_\ell(K)$ more efficiently. The chain modules $C_\ell(K)$ of multi-parameter filtrations are still decomposable into interval modules, with intervals representing the simplices (see Figure 4). But simplices can be generated at different incomparable grades of the filtration and get identified at their joins. Hence, contrary to the case of one-parameter filtrations, $C_\ell(K)$ can have relations and is not necessarily projective. To obtain a valid input for the algorithm of [25], we first have to convert the simplicial input into a free/projective implicit representation by graded matrices. In [9], a method to convert a general simplicial filtration over a d-dimensional grid to a FiRep is discussed. If we consider arbitrary simplicial maps over a grid poset, as in the example in Figure 5, the chain modules $C_\ell(K)$ will not even be interval decomposable anymore. Different simplices can be glued together by collapses, yielding more complicated indecomposables.

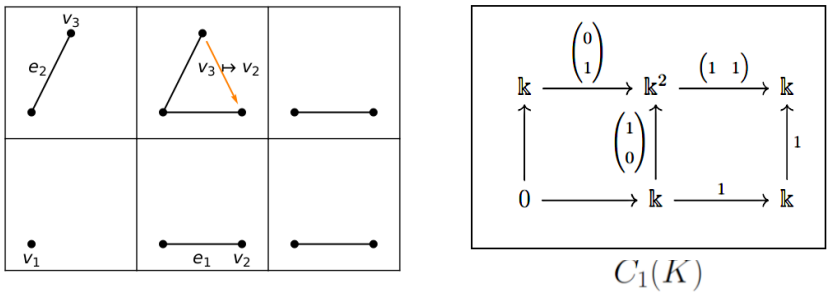


Figure 5 Left: a poset tower over a 2×3-grid. The orange arrow denotes the collapse of the vertex v_3 into the vertex v_2 when moving to the right. Right: The indecomposable persistence module representing $C_1(K)$.

What about going beyond d-parameter filtrations? That leads us to poset towers. In topological data analysis (TDA), these general-purpose frameworks have already started to appear. In [14, 18], zigzag-grids, that is, grids with arrows pointing in both directions along one axis (which gives quasi zigzag persistent homology as described in [14]), are considered.

The authors in [14] argue that the resulting quasi zigzag persistence module is useful to study time-varying data. Also, persistent homology over general posets has become a subject of study [24] in TDA. If we consider arbitrary indexing posets, the chain modules $C_\ell(K)$ will not even be interval decomposable in the case of inclusions. The example in Figure 6 shows a poset tower over a poset that does not have a maximum. In this case, simplices can be identified in some branches and can be different in other branches. In other words, we can no longer speak about "the simplices" in absolute terms, because two simplices can be the same from the perspective of one maximal element but can be different from the perspective of another maximal element. Therefore, except for the special case of one-parameter filtrations, the chain modules $C_\ell(K)$ are, in general, not projective and can have a complicated structure. In this work, we tackle the problem of computing a projective implicit representation of the persistent homology of a general poset tower.

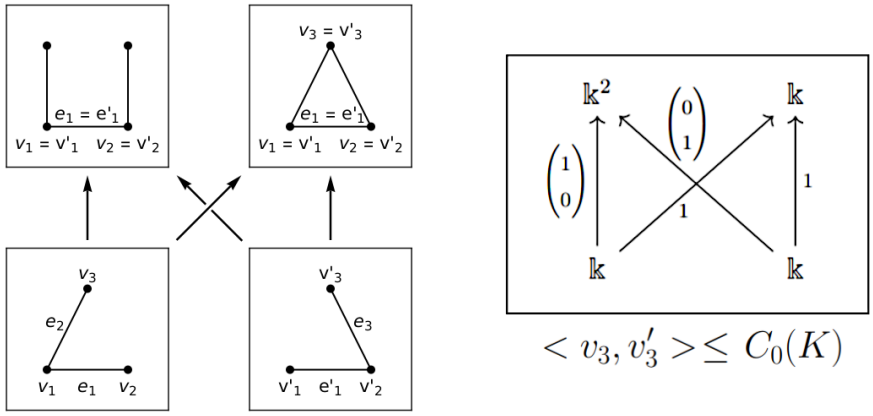


Figure 6 Left: a poset tower over a poset with two maximal elements, where all maps are inclusions. At the left maximal element the two vertices v_3 and v_3' are different. In the right maximal element they are identified. Right: The indecomposable summand $< v_3, v_3' >$ of $C_0(K)$, generated by the two vertices v_3 and v_3'.

Contributions. We develop a general-purpose algorithm (PIREP in Section 4) to compute projective implicit representations of the homology of poset towers $K\colon P \to \mathbf{SCpx}$, where we allow arbitrary simplicial maps and arbitrary (finite) indexing posets. The algorithm PIREP computes degreewise (projective) minimal presentations of the (persistent) simplicial chain complex $C_\bullet(K)$. This means that, for every ℓ, we compute a minimal presentation p_ℓ^1 of the chain module $C_\ell(K)$, as depicted in (1), and a lift f_ℓ^0 of the boundary map ∂_ℓ to the generators.

$$\begin{array}{ccccccc}
C_{\ell+1}(K) & \longleftarrow & G_{\ell+1} & \xleftarrow{p^1_{\ell+1}} & R_{\ell+1} & \xleftarrow{p^2_{\ell+1}} & RR_{\ell+1} \\
\downarrow{\scriptstyle \partial_{\ell+1}} & & \downarrow{\scriptstyle f^0_{\ell+1}} & \swarrow{\scriptstyle \vartheta_{\ell+1}} & \downarrow{\scriptstyle f^1_{\ell+1}} & & \\
C_\ell(K) & \longleftarrow & G_\ell & \xleftarrow{p^1_\ell} & R_\ell & \xleftarrow{p^2_\ell} & RR_\ell \\
\downarrow{\scriptstyle \partial_\ell} & & \downarrow{\scriptstyle f^0_\ell} & \swarrow & \downarrow{\scriptstyle f^1_\ell} & & \\
C_{\ell-1}(K) & \longleftarrow & G_{\ell-1} & \xleftarrow{p^1_{\ell-1}} & R_{\ell-1} & \xleftarrow{p^2_{\ell-1}} & RR_{\ell-1}
\end{array} \tag{1}$$

Our algorithm takes advantage of the special structure of the chain complex $C_\bullet(K)$. We

observe that the presentation matrix p_ℓ^1 (almost) has the structure of the boundary matrix of a P-filtered graph (Definition 39). This allows us to construct a minimal presentation by constructing a graph and avoiding closing (superfluous) cycles. We represent the maps p_ℓ^1 and f_ℓ^0 by P-graded matrices. Together, they faithfully represent the chain complex $C_\bullet(K)$. Given this representation, one could now compute a projective implicit representation of $H_\ell(K)$ by purely algebraic methods. However, we observe that the structure of $C_\bullet(K)$ can be further exploited to efficiently compute an asymptotically minimal second term p_ℓ^2 in a projective resolution of $C_\ell(K)$, as depicted in (1). Here asymptotic minimality means that the number of columns of p_ℓ^2 is asymptotically worst-case optimal. Individual outputs need not be exactly minimal. We compute p_ℓ^2, which can be interpreted as capturing relations among relations, by detecting cycles in a graph associated with p_ℓ^1. We take advantage of the graph structure of p_ℓ^1 a third time to compute the maps f_ℓ^1, lifting the boundary maps ∂_ℓ to the relations, and $\vartheta_{\ell+1}$, which corrects for the failure of the maps f_ℓ^0 to form a chain complex. These maps can be determined by solving linear systems, but the graph structure of the coefficient matrix allows us to solve these systems much more efficiently. Finally, we construct a projective implicit representation of $H_\ell(K)$ by assembling the maps in (1) (see Section 3).

At this point, it is worth mentioning that, in [6], the authors provide an algorithm to compute minimal injective resolutions of sheaves on finite posets. A sheaf of vector spaces on a finite poset is equivalent to a P-persistence module, and an injective resolution of a P-persistence module is equivalent to a projective resolution of a P^{op}-persistence module obtained by dualization. Their algorithm does not address computing lifts f_ℓ, as in (1), but could be applied in our context to compute the degreewise resolutions. For a resolution of $C_\ell(K)$, as depicted in the rows of (1), the complexity of the algorithm in [6] is $O(t_0 c^3)$, where t_0 denotes the number of points in the poset, and c is the maximal size of the graded matrices in the output resolution. This cubic behavior, with an additional factor from the size of the poset, is of course expected by a general algorithm based on matrix reduction. A matrix representing a minimal p_ℓ^2 can have $\Theta(t_1 n)$ columns if n denotes the number of simplex generators and t_1 the number of edges in the Hasse diagram of the poset (see Proposition 48). This would result in a worst-case complexity of $O(t_0 t_1^3 n^3)$, while our specialized algorithm achieves a complexity of $O\big((t_0 + t_1)n^2\big)$.

It is also worth noting that, for the special case of K being a graph, efficient algorithms for computing minimal presentations are known. The authors of [28] give an algorithm to compute a minimal presentation of $H_0(G)$ of a P-filtered graph G in $O(|G|^2)$ time for a general poset P and in $O(|G| \log |G|)$ time for $P = \mathbb{Z}^2$. The algorithm is not applicable to the general case of simplicial filtrations or towers indexed by a (finite) poset.

Outline. In Section 2, we discuss the basic notions of poset towers, persistence modules, projective modules, projective resolutions, and projective implicit representations. In Section 3, we discuss how we can construct a PiRep from partial projective resolutions of $C_\ell(K)$ and lifts of the boundary maps ∂_ℓ, as depicted in (1). In Section 4, we present an algorithm PiRep to compute a projective implicit representation from a poset tower. In Section 5, we prove the correctness of the algorithm. We conclude in Section 6. We provide proofs omitted from the main text in Appendix A and B. In Appendix C, we discuss an efficient algorithm for solving systems of linear equations whose coefficient matrix has the structure of the boundary matrix of a tree, which is used in the algorithm PiRep. For completeness, we discuss in Appendix D how to compute a minimal presentation of $H_\ell(K)$ from a PiRep using an algorithm discussed in [6].

2 Poset towers, PiReps, and algebraic background

In this section, we introduce the basic notions of poset towers, persistence modules, projective modules, projective resolutions, and projective implicit representations.

Notation. In the following, we denote by P a finite partially ordered set (poset) with its points referred to as *grades*. For $x, y \in P$, we use the notation $x \prec y$ if $x < y$ and there is no grade $z \in P$ such that $x < z < y$. We call such an x a predecessor of y. We assume that P is represented by a directed graph (called the *Hasse diagram*), whose nodes are the grades and whose edges are relations $x \prec y$ in P. We denote by $\overline{P} \coloneqq P \cup \{-\infty\}$ the poset obtained from P by adding a grade $-\infty$ such that $-\infty < x$ for all $x \in P$.

We also view P as a category with objects the elements of P and a unique morphism $x \to y$ if $x \leq y$. For a functor $F \colon P \to \mathbf{C}$, we denote by $F(x)$ the image of $x \in P$ under F and by $F(x \leq y)$ the image of $x \to y$ under F for $x \leq y \in P$. Let the category of functors $P \to \mathbf{C}$ and their natural transformations be denoted by $\mathbf{C}^P$. We use the notation **Vec** for the category of finite-dimensional vector spaces over $\mathbb{F}_2$, **SCpx** for the category of finite simplicial complexes and simplicial maps, and $\mathbf{\Delta Cpx}_{\leq 1}$ for the category of finite one-dimensional Δ-complexes (see Chapter 2.1 in [21]). For simplices σ and τ in a simplicial or Δ-complex, we write $\sigma \leq \tau$ if σ is a face of τ, and $\sigma < \tau$ if σ is a proper face of τ.

Coefficients. Throughout this work, we use coefficients in $\mathbb{F}_2$, as is common in topological data analysis. This choice simplifies both the exposition and the algorithmic treatment. Since we allow general simplicial maps over general posets, working over an arbitrary field would require additional orientation data and sign bookkeeping at the chain level. In particular, one would need to keep track of the signs induced by simplicial maps on oriented simplices, which would substantially complicate both the input representation and the resulting algorithms. We do not see an immediate conceptual obstruction to extending our methods to general field coefficients, but the present algorithm and its proof of correctness are developed specifically for $\mathbb{F}_2$. Finally, we note that although we formulate the algebraic results in Sections 2 and 3 over $\mathbb{F}_2$, they remain true over arbitrary fields after the appropriate signs are taken into account.

2.1 Poset towers and persistence modules

Generalizing the notion of simplicial tower [11, 23], we introduce poset towers:

▶ **Definition 1** (Poset tower). *A poset tower is a functor $K \colon P \to \mathbf{SCpx}$; see, e.g., Figure 7.*

We denote by $K_\ell(x)$ the set of all ℓ-simplices in $K(x)$. In particular, $K_0(x)$ is the set of vertices of $K(x)$. We define $\mathrm{Vert}(K) \coloneqq \bigcup_{x \in P} K_0(x)$ as the set of all vertices in K where the union is the ordinary set-theoretic union and vertex sets at different grades may overlap. For each $x \in P$, every simplex of $K(x)$ is uniquely determined by its set of vertices in $\mathrm{Vert}(K)$. The simplicial maps need not preserve these labels.

In most applications, it is not economical to store a poset tower as a simplicial complex for every grade of the poset P. For example, in the case of a multi-filtered simplicial complex, which, in our setting, is a poset tower $K \colon P \to \mathbf{SCpx}$ over a finite grid $P = \{1, \dots, m\}^d$ where each map $K(x \leq y)$ is an inclusion, we only need a list of birth grades for every simplex in the complex $K(\max)$ at the maximum grade $\max = (m, \dots, m) \in P$. We generalize this compact representation to general poset towers by introducing simplex generators and edge events.

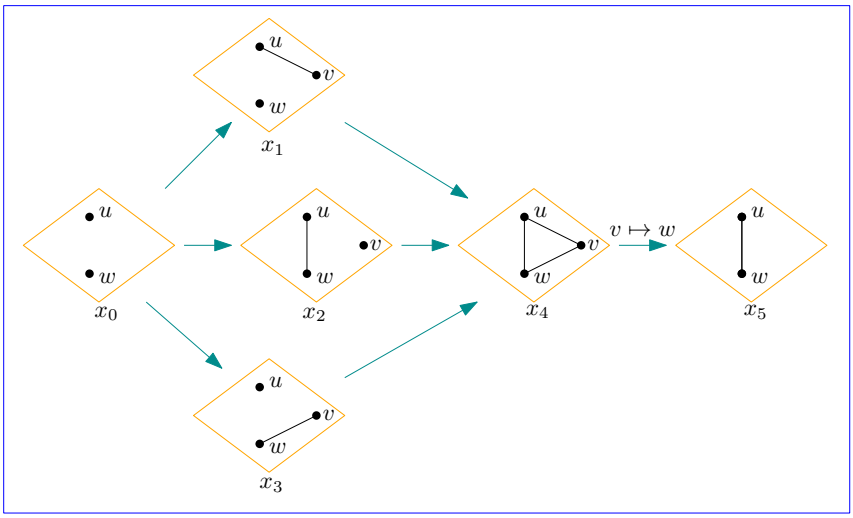


Figure 7 A poset tower on a poset $P = \{x_0, \dots, x_5\}$ with simplicial complexes comprising vertices and edges at the nodes ($\diamond$) and inclusions/collapses on poset edges ($\to$); e.g. vertex v is generated at x_1, x_2, x_3, i.e., it has generators $g_v^{x_1}, g_v^{x_2}, g_v^{x_3}$, which generate two relations $r_1^{x_4} \mapsto g_v^{x_1} + g_v^{x_2}$ and $r_2^{x_4} \mapsto g_v^{x_2} + g_v^{x_3}$ at x_4; there is a vertex collapse $v \mapsto w$ on the poset edge x_4x_5; this collapse generates the relation $r_1^{x_5} \mapsto g_{vw}^{x_3}$, it also generates two copies of uw at x_5 with generators $g_{uv}^{x_1}$ and $g_{uw}^{x_2}$ which get identified with a relation $r_2^{x_5} \mapsto g_{uw}^{x_2} + g_{uv}^{x_1}$ at x_5.

▶ **Definition 2** (Simplex generators). *Let $K\colon P \to \mathbf{SCpx}$ be a poset tower. We call the tuple $g_\sigma^x := (x, \sigma)$ a generator of the simplex σ at $x \in P$ if $\sigma \in K(x)$ and $\sigma \notin \operatorname{im}\big(K(y \prec x)\big)$ for all $y \prec x \in P$. We denote by $\mathcal{S}$ the set of all simplex generators of K. Formally:*

$$\mathcal{S} := \Big\{ g_\sigma^x \,|\, x \in P, \sigma \in K(x) \setminus \bigcup_{y \prec x} \operatorname{im}\big(K(y \prec x)\big) \Big\}.$$

We denote by $\mathcal{S}_\ell$ the set of generators of ℓ-simplices.

Note that, for the special case of a multi-filtered simplicial complex K, the simplex generators g_σ^x simply recover the birth grades of a simplex $\sigma \in K(\max)$ in the complex at the maximum. For general poset towers, the definition of simplex generators provides the flexibility needed to handle situations in which simplices may be collapsed or no maximum grade exists; see Figures 5 and 6.

Next, we introduce edge events as a compact way to represent the simplicial maps in a poset tower. An edge event over an edge $x \prec y$ is a vertex in $K(x)$ that is mapped to a vertex different from itself along the edge. These edge events account for possible collapses and permutations that can happen with arbitrary simplicial maps, which are simulated by vertex identifications. The collapse $v \mapsto w$ on the poset edge x_4x_5 in Figure 7 is an edge event.

▶ **Definition 3** (Edge events). *Let $K\colon P \to \mathbf{SCpx}$ be a poset tower. We call the tuple $c_{v \mapsto w}^{x \prec y} := (x, y, v, w)$ an edge event if $x \prec y \in P$, $v \in K_0(x)$, $w = K(x \prec y)(v) \in K_0(y)$, and $w \neq v$ as elements in $\operatorname{Vert}(K)$. We denote by $\mathcal{C}$ the set of all edge events of K. Formally:*

$$\mathcal{C} := \Big\{ c_{v \mapsto w}^{x \prec y} \,|\, x \prec y \in P, v \in K_0(x), w = K(x \prec y)(v) \in K_0(y), v \neq w \Big\}.$$

Instead of recording all simplicial maps, we just record when they change a simplex. A multi-filtered simplicial complex, for example, has no edge events. The simplex generators and edge events completely determine the poset tower.

▶ **Proposition 4.** *A poset tower $K\colon P \to \mathbf{SCpx}$ is completely described by the list of simplex generators $\mathcal{S}$ and edge events $\mathcal{C}$ induced by K.*

Proof. Let $\mathcal{S}$ and $\mathcal{C}$ be as in Definitions 2 and 3. First, we show that

$$\mathrm{Vert}(K) = \{v | g_v^x \in \mathcal{S}_0\} \cup \{w | c_{v\mapsto w}^{x\prec y} \in \mathcal{C}\}. \tag{2}$$

Let $u \in \mathrm{Vert}(K)$. Then there exists $x \in P$ such that $u \in K_0(x)$. If $u \notin \mathrm{im}\big(K(y \prec x)\big)$ for all $y \prec x \in P$, then $g_u^x \in \mathcal{S}_0$. Otherwise, there exist $y \prec x$ and $v \in K_0(y)$ such that $K(y \prec x)(v) = u$. If $v \neq u$ in $\mathrm{Vert}(K)$, then $c_{v\mapsto u}^{y\prec x} \in \mathcal{C}$. Else $v = u \in K_0(y)$ and we can repeat the same argument for $y < x$. Since P is finite and every vertex at a minimal grade has a generator in $\mathcal{S}_0$, this process terminates, yielding either a generator $g_u^z \in \mathcal{S}_0$ or an edge event whose target is u. Conversely, by definition, $g_v^x \in \mathcal{S}_0$ implies $v \in K_0(x) \subseteq \mathrm{Vert}(K)$ and $c_{v\mapsto w}^{x\prec y} \in \mathcal{C}$ implies $w \in K_0(y) \subseteq \mathrm{Vert}(K)$. This finishes the proof of (2).

We now extend P to a linear order $P_{lin} = \{x_1 <_{lin} \cdots <_{lin} x_n\}$ and construct K' recursively. For the construction we use $\mathrm{Vert}(K)$ which can be obtained directly from $\mathcal{S}$ and $\mathcal{C}$ by (2) and denote by $\Delta_{\mathrm{Vert}(K)}$ the complete simplicial complex on the vertex set $\mathrm{Vert}(K)$. For $i = 1, \ldots, n$, define

$$K'(x_i) \coloneqq \{\sigma | \exists g_\sigma^{x_i} \in \mathcal{S}\} \cup \bigcup_{y \prec x_i} \mathrm{im}\big(\overline{K}'(y \prec x_i)\big).$$

For all $x_i \prec y$, define $\overline{K}'(x_i \prec y)\colon K'(x_i) \to \Delta_{\mathrm{Vert}(K)}$ as the simplicial map defined on vertices $v \in K_0'(x_i)$ by $\overline{K}'(x_i \prec y)(v) \coloneqq w$ if there exists a $c_{v\mapsto w}^{x_i\prec y} \in \mathcal{C}$ and $\overline{K}'(x_i \prec y)(v) \coloneqq v$ otherwise. After processing all x_i in P_{lin}, we define $K'(x \prec y)$ as the restriction of $\overline{K}'(x \prec y)$ to a map $K'(x \prec y)\colon K'(x) \to K'(y)$, for all $x \prec y \in P$, where we use that $\mathrm{im}\,\big(\overline{K}'(x \prec y)\big) \subseteq K'(y)$.

We now show $K = K'$: Suppose $K(x_i) = K'(x_i)$, $v \in K_0'(x_i) = K_0(x_i)$ and $x_i \prec y$. If $K(x_i \prec y)(v) = v$, there is no edge event $c_{v\mapsto w}^{x_i\prec y}$ in $\mathcal{C}$ and, thus, $K'(x_i \prec y)(v) = v$. If $K(x_i \prec y)(v) = w \neq v$, there is an edge event $c_{v\mapsto w}^{x_i\prec y} \in \mathcal{C}$ and, thus, $K'(x_i \prec y)(v) = w$. Since the images of the vertices completely determine a simplicial map, we get $K(x_i \prec y) = K'(x_i \prec y)$.

We show that $K(x_i) = K'(x_i)$ by induction on the grades in the linear order. The first element x_1 must be minimal in P. Hence, by definition, for all $\sigma \in K(x_1)$ we have $g_\sigma^{x_1} \in \mathcal{S}$. Therefore, $K(x_1) = K'(x_1)$. Suppose $K(x_j) = K'(x_j)$ for all $j < i$. Every predecessor $y \prec x_i$ satisfies $y = x_j$ for some $j < i$ in the linear extension. Hence, by the argument in the previous paragraph, we get $K(x_i) = \{\sigma | \exists g_\sigma^{x_i} \in \mathcal{S}\} \cup \bigcup_{y\prec x_i} \mathrm{im}\big(K(y \prec x_i)\big) = \{\sigma | \exists g_\sigma^{x_i} \in \mathcal{S}\} \cup \bigcup_{y\prec x_i} \mathrm{im}\big(K'(y \prec x_i)\big) = K'(x_i)$.

Therefore, $K(x) = K'(x)$ and $K(x \prec y) = K'(x \prec y)$ for all $x \prec y \in P$, which implies that $K = K'$. ◀

We now define the main algebraic objects, the persistence modules, and discuss how we can work with them in a computational framework.

▶ **Definition 5** (P-persistence module)**.** *A P-persistence module is a functor $M\colon P \to \mathbf{Vec}$.*

The P-persistence modules are precisely representations of finite posets. Calling these objects modules is justified by the fact that the category of representations of finite posets is equivalent to the category of modules over the so-called path or incidence algebra of the poset [2, 32]. Persistence modules in our context naturally arise as the homology of poset towers. The ℓ-th homology, with coefficients in $\mathbb{F}_2$, can be viewed as a functor $H_\ell\colon \mathbf{SCpx} \to \mathbf{Vec}$ for every $\ell \geq 0$. If we take the composition of the ℓ-th homology functor with a poset tower $K\colon P \to \mathbf{SCpx}$, we obtain the persistence module $H_\ell(K) \coloneqq H_\ell \circ K\colon P \to \mathbf{Vec}$.

The homology of a simplicial complex is computed from the simplicial chain complex, which can be viewed as an algebraic representation of the complex. For $x \leq y \in P$, we denote by $C_\bullet\big(K(x)\big)$ the simplicial chain complex and by $C_\bullet\big(K(x \leq y)\big)$ the morphism induced by the simplicial map $K(x \leq y)$; see (3). The vector space $C_\ell\big(K(x)\big)$ has a basis consisting of the ℓ-simplices in $K(x)$ and $C_\ell\big(K(x \leq y)\big)$ sends $\sigma \in K_\ell(x)$ to $K(x \leq y)(\sigma)$ if $K(x \leq y)(\sigma) \in K_\ell(y)$ and to zero otherwise.

$$\begin{array}{ccccccccc} \cdots & \longleftarrow & C_{\ell-1}\big(K(x)\big) & \xleftarrow{\partial_\ell} & C_\ell\big(K(x)\big) & \xleftarrow{\partial_{\ell+1}} & C_{\ell+1}\big(K(x)\big) & \longleftarrow & \cdots \\ & & \downarrow{\scriptstyle C_{\ell-1}\left(K(x\leq y)\right)} & & \downarrow{\scriptstyle C_\ell\left(K(x\leq y)\right)} & & \downarrow{\scriptstyle C_{\ell+1}\left(K(x\leq y)\right)} & & \\ \cdots & \longleftarrow & C_{\ell-1}\big(K(y)\big) & \xleftarrow{\partial_\ell} & C_\ell\big(K(y)\big) & \xleftarrow{\partial_{\ell+1}} & C_{\ell+1}\big(K(y)\big) & \longleftarrow & \cdots \end{array} \tag{3}$$

Let $\mathbf{Ch}\big(\mathbf{Vec}\big)$ and $\mathbf{Ch}\big(\mathbf{Vec}^P\big)$ denote the categories of chain complexes over vector spaces and over P-persistence modules, respectively. We can view $C_\bullet(K)\colon P \to \mathbf{Ch}(\mathbf{Vec})$ as a persistent chain complex or, equivalently, as a chain complex of persistence modules in $\mathbf{Ch}\big(\mathbf{Vec}^P\big)$:

$$\cdots \longleftarrow C_{\ell-1}(K) \xleftarrow{\partial_\ell} C_\ell(K) \xleftarrow{\partial_{\ell+1}} C_{\ell+1}(K) \longleftarrow \cdots$$

where each $C_\ell(K)\colon P \to \mathbf{Vec}$ is a persistence module. We will adopt this global perspective in the following.

2.2 Projective modules

To carry out computations with persistence modules, we need to represent them by matrices. As with poset towers, it is not economical to represent a persistence module by a collection of matrices $M(x \leq y)$ for all $x \leq y \in P$. Instead, we represent persistence modules by elementary building blocks called projective modules. Since we work with a specific class of modules, we will directly define the projective modules as in Definition 6. See [2, Theorem III.1.6, Lemma III.2.4] for a reference showing that these are exactly the projective modules according to the general definition [2, Definition I.5.2]. Note that in persistence theory, one often works with graded free modules. For $\mathbb{Z}^d$-graded modules over graded polynomial rings, these coincide with graded projective modules. In the setting of modules over finite posets, however, projective modules are generally different from free modules; see, for example, [2, Chapters I.5, III.2] or [32, Definition 2.6].

▶ **Definition 6** (Projective modules). *We denote by* $\mathrm{Proj}[x]\colon P \to \mathbf{Vec}$ *the elementary projective module at* $x \in P$ *defined for all* $y \leq z \in P$ *by*

$$\mathrm{Proj}[x](y) = \begin{cases} \mathbb{F}_2 & \text{if } x \leq y \\ 0 & \text{otherwise} \end{cases}, \qquad \mathrm{Proj}[x](y \leq z) = \begin{cases} id & \text{if } x \leq y \leq z \\ 0 & \text{otherwise} \end{cases}.$$

A direct sum $\bigoplus_{i=1}^n \mathrm{Proj}[x_i]$ *is called a projective module; see Figure 8 for an example.*

Projective modules are well-suited for computations because morphisms between them are simple and can be described by P-graded matrices.

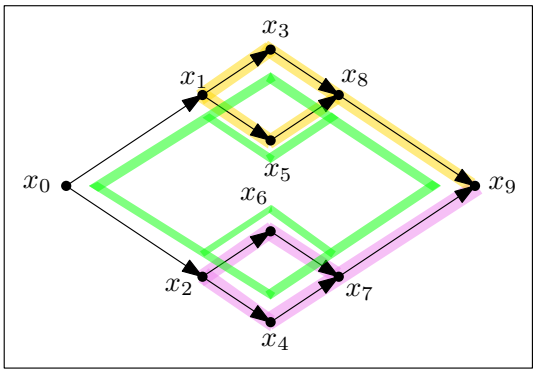


Figure 8 An illustration of a projective module with three elementary projective summands; $\mathrm{Proj}[x_0]$ shaded green, $\mathrm{Proj}[x_1]$ shaded yellow, and $\mathrm{Proj}[x_2]$ shaded violet.

▶ **Proposition 7** (Properties of morphisms between projective modules)**.**

1. *For $x, y \in P$:*

$$\mathrm{Hom}\left(\mathrm{Proj}[x], \mathrm{Proj}[y]\right) \cong \begin{cases} \mathbb{F}_2 & \text{if } y \leq x \\ 0 & \text{otherwise} \end{cases}$$

   as $\mathbb{F}_2$-vector spaces.

2. *For $Q = \bigoplus_{i=1}^{m} \mathrm{Proj}[x_i]$ and $W = \bigoplus_{j=1}^{n} \mathrm{Proj}[y_j]$ projective modules:*

$$\begin{aligned} \mathrm{Hom}\left(Q, W\right) &\cong \mathrm{Hom}\left(\bigoplus_{i=1}^{m} \mathrm{Proj}[x_i], \bigoplus_{j=1}^{n} \mathrm{Proj}[y_j]\right) \\ &\cong \bigoplus_{j=1}^{n} \bigoplus_{i=1}^{m} \mathrm{Hom}\left(\mathrm{Proj}[x_i], \mathrm{Proj}[y_j]\right) \\ &\cong \left\{(\lambda_{ji})_{j,i} \in \mathbb{F}_2^{n \times m} \middle| \lambda_{ji} = 0 \text{ if } y_j \not\leq x_i\right\}. \end{aligned}$$

Proof. See Appendix A. ◀

▶ **Definition 8** (P-graded matrix)**.** *Let $f\colon Q \to W$ be a morphism of projective modules as in Proposition 7(2). Then f is completely determined by a matrix $(f_{ji})_{i=1,j=1}^{m,n} \in \mathbb{F}_2^{n \times m}$ whose columns and rows are identified with the elementary projective modules in Q and W, respectively. We call a matrix representation $(f_{ji})_{i=1,j=1}^{m,n}$ whose columns are labeled with the grades x_i of the summands $\mathrm{Proj}[x_i]$ of Q and whose rows are labeled with the grades y_j of the summands $\mathrm{Proj}[y_j]$ of W a P-graded matrix representation of f.*

Notation for projective generators. To simplify the exposition, we introduce the following notation: We view an elementary projective module $\mathrm{Proj}[x]\colon P \to \mathbf{Vec}$ as a formal span of a projective generator $\langle b^x \rangle$ and a general projective module $\bigoplus_{i=1}^{n} \mathrm{Proj}[x_i]$ as $\langle b^{x_1}, \ldots, b^{x_n} \rangle$. If we pick an element $x \in P$, then the vector space $\left(\bigoplus_{i=1}^{n} \mathrm{Proj}[x_i]\right)(x)$ has a basis $\{b^{x_j} | x_j \leq x\}$ consisting of all formal projective generators born below x in P. Moreover, for $x \leq y \in P$, the linear map $\left(\bigoplus_{i=1}^{n} \mathrm{Proj}[x_i]\right)(x \leq y)$ is induced by the inclusion of basis elements $\{b^{x_j} | x_j \leq x\} \hookrightarrow \{b^{x_k} | x_k \leq y\}$. This allows us to simplify the notation for maps

between projective modules $\mathrm{Proj}[y] \xrightarrow{1} \mathrm{Proj}[x]$ by $b^y \mapsto b^x$ for $x \leq y$ and, more generally,

$$\mathrm{Proj}[y] \xrightarrow{\begin{pmatrix}1\\\vdots\\1\end{pmatrix}} \bigoplus_{i=1}^{n} \mathrm{Proj}[x_i]$$

by $b^y \mapsto b^{x_1} + \cdots + b^{x_n}$. For fixed $x_i \leq y \leq z \in P$ this can again be interpreted as the linear map sending the basis element b^y of $\big(\mathrm{Proj}[y]\big)(z)$ to the sum of basis elements $b^{x_1} + \cdots + b^{x_n}$ of $\big(\bigoplus_{i=1}^{n} \mathrm{Proj}[x_i]\big)(z)$. Given this notion of projective generators, we can also define P-graded matrices to have columns and rows labeled by the projective generators of the corresponding summands.

Equations of morphisms of projective modules. In the following, we will encounter the problem of solving equations $A \circ X = B$ of morphisms of projective modules, as depicted in (4). The special properties of projective modules allow us to solve such equations, which we will call *P-graded linear systems*, via standard linear algebra.

▶ **Definition 9** (P-graded linear system). *For a poset P, projective modules $Q = \bigoplus_{i=1}^{n} \mathrm{Proj}[x_i]$, $U = \bigoplus_{i=1}^{m} \mathrm{Proj}[y_i]$, $R = \bigoplus_{i=1}^{s} \mathrm{Proj}[z_i]$, and morphisms A and B as in* (4),

$$\begin{array}{ccc} U & & \\ {\scriptstyle B}\downarrow & \searrow^{X} & \\ Q & \xleftarrow[A]{} & R \end{array} \tag{4}$$

we call the equation $A \circ X = B$ a P-graded linear system for the unknown morphism X.

By Proposition 7, the vector space $\mathrm{Hom}(U, R)$ can be described as $\{(X_{ij})_{i,j} \in \mathbb{F}_2^{s \times m} | X_{ij} = 0 \text{ if } z_i \not\leq y_j\}$. Using analogous matrix representations $(A_{ij})_{i,j}$ and $(B_{ij})_{i,j}$ of A and B, respectively, we obtain the matrix representation of the composition by matrix multiplication: $(A \circ X)_{ij} = \sum_{k=1}^{s} A_{ik} X_{kj}$. The P-graded linear system $A \circ X = B$ can then equivalently be defined as the following constraint linear system with coefficient matrix A:

$$\begin{aligned} &(A \circ X)_{ij} = \sum_{k=1}^{s} A_{ik} X_{kj} = B_{ij} \ , \ \forall\, 1 \leq i \leq n,\, 1 \leq j \leq m \\ &X_{kj} = 0 \ , \ \forall\, 1 \leq k \leq s,\, 1 \leq j \leq m \text{ such that } z_k \not\leq y_j. \end{aligned} \tag{5}$$

The following proposition shows that a P-graded linear system, as in Definition 9, can be reduced to m unconstrained linear systems.

▶ **Proposition 10.** *Let $AX = (AX_1, \ldots, AX_m) = (B_1, \ldots, B_m) = B$ and $X_{kj} = 0$ for all $z_k \not\leq y_j$ be a P-graded linear system as in Definition 9. Let $A^{\leq y_j}$, $A^{\not\leq y_j}$, $X_j^{\leq y_j}$, and $X_j^{\not\leq y_j}$ be the submatrices of columns and rows indexed by z_k such that $z_k \leq y_j$ and $z_k \not\leq y_j$, respectively. Then $AX = B$ and $X_{kj} = 0$ for all $z_k \not\leq y_j$ if and only if $A^{\leq y_j} X_j^{\leq y_j} = B_j$ and $X_j^{\not\leq y_j} = 0$ for all $1 \leq j \leq m$.*

Proof. See Appendix A. ◀

2.3 Projective resolutions

Every persistence module, and also every morphism between persistence modules can be expressed by projective modules and morphisms between them. We will use this fact to translate complex algebraic objects into P-graded matrices.

▶ **Definition 11** (Projective resolution and presentation).

1. *A projective resolution of a P-persistence module M is a chain complex*

$$0 \xleftarrow{q_{-1}} M \xleftarrow{q_0} Q_0 \xleftarrow{q_1} Q_1 \xleftarrow{q_2} Q_2 \xleftarrow{q_3} \cdots \tag{6}$$

where Q_i is a projective module and $\ker(q_{i-1}) = \operatorname{im}(q_i)$ for all $i \geq 0$.

2. *Let $\phi\colon M \to N$ be a morphism of P-persistence modules and $M \leftarrow Q_\bullet$ and $N \leftarrow W_\bullet$ be projective resolutions of M and N, respectively. A lift of ϕ to these projective resolutions is a family of morphisms $f_\bullet\colon Q_\bullet \to W_\bullet$:*

$$\begin{array}{ccccccccccc} 0 & \xleftarrow{q_{-1}} & M & \xleftarrow{q_0} & Q_0 & \xleftarrow{q_1} & Q_1 & \xleftarrow{q_2} & Q_2 & \xleftarrow{q_3} & \cdots \\ & & \phi\downarrow & & \downarrow f_0 & & \downarrow f_1 & & \downarrow f_2 & & \\ 0 & \xleftarrow{w_{-1}} & N & \xleftarrow{w_0} & W_0 & \xleftarrow{w_1} & W_1 & \xleftarrow{w_2} & W_2 & \xleftarrow{w_3} & \cdots \end{array} \tag{7}$$

such that all squares in (7) commute.

3. *A (projective) presentation of a P-persistence module M is an exact sequence of the form*

$$0 \leftarrow M \xleftarrow{q_0} Q_0 \xleftarrow{q_1} Q_1 \tag{8}$$

where Q_0 and Q_1 are projective modules, q_0 is an epimorphism, and $\ker(q_0) = \operatorname{im}(q_1)$.

Note that while $M \xleftarrow{q_0} Q_0$ in Definition 11 is important for theoretical arguments and proofs, in practice M is fully described by q_1 since $M \cong \operatorname{coker}(q_1)$.

▶ **Proposition 12** (Existence of projective resolutions).

1. *Let M be a P-persistence module. Then there exists a projective resolution, as in (6). The resolution is unique up to homotopy equivalence of chain complexes.*
2. *Let $\phi\colon M \to N$ be a morphism of P-persistence modules and $M \leftarrow Q_\bullet$ and $N \leftarrow W_\bullet$ projective resolutions. Then there exists a lift $f_\bullet\colon Q_\bullet \to W_\bullet$ of ϕ, as in (7). The lift is unique up to chain homotopy. This means that if $g_\bullet\colon Q_\bullet \to W_\bullet$ is another lift of ϕ:*

$$\begin{array}{ccccccccccc} 0 & \xleftarrow{q_{-1}} & M & \xleftarrow{q_0} & Q_0 & \xleftarrow{q_1} & Q_1 & \xleftarrow{q_2} & Q_2 & \xleftarrow{q_3} & \cdots \\ & & \phi\downarrow & & g_0\Downarrow f_0 & \swarrow_{h_0} & g_1\Downarrow f_1 & \swarrow_{h_1} & g_2\Downarrow f_2 & \swarrow_{h_2} & \\ 0 & \xleftarrow{w_{-1}} & N & \xleftarrow{w_0} & W_0 & \xleftarrow{w_1} & W_1 & \xleftarrow{w_2} & W_2 & \xleftarrow{w_3} & \cdots \end{array}$$

then there exists a chain homotopy $h_\bullet\colon Q_\bullet \to W_{\bullet+1}$ such that $f_i + g_i = h_{i-1} \circ q_i + w_{i+1} \circ h_i$ for all $i \geq 1$ and $f_0 + g_0 = w_1 \circ h_0$.

3. *Resolutions and lifts can be constructed iteratively. Any partial resolution or lift satisfying the conditions in Definition 11(1) or (2) up to index $i \geq 0$ can be extended to a full resolution or lift, respectively.*

Proof. See Lemma 2.2.5 and Theorem 2.2.6 in [33]. ◀

Minimal projective resolutions. Since presentations and resolutions are not unique up to isomorphism, they can contain superfluous information. For computational efficiency, we want resolutions that are as small as possible.

▶ **Definition 13** (Minimal projective resolution). *A projective resolution $M \leftarrow Q_\bullet$, as in Definition 11, is called minimal if Q_i minimizes the number of elementary projective summands for all $i \geq 0$. A presentation is called minimal if Q_0 and Q_1 minimize the number of elementary projective summands.*

We will use an equivalent characterization of minimal resolutions based on the notion of the radical and projective cover. This notion of minimality will be crucial for the algorithm PiRep and the proof of its correctness.

▶ **Definition 14** (Radical). *Let M be a P-persistence module. Then* $\mathrm{Rad}(M)\colon P \to \mathbf{Vec}$ *defined for all $x \leq y \in P$ by*

$$\mathrm{Rad}(M)(x) \coloneqq \sum_{u<x} \mathrm{im}\big(M(u<x)\big)$$

$$\mathrm{Rad}(M)(x \leq y) \coloneqq \left[\sum_{u<x} \mathrm{im}\big(M(u<x)\big) \xrightarrow{M(x\leq y)} \sum_{v<y} \mathrm{im}\big(M(v<y)\big)\right]$$

is called the radical of M.

▶ **Definition 15** (Projective cover). *Let M be a P-persistence module. Then a morphism $g\colon Q \to M$ is called a projective cover of M if Q is projective, g is an epimorphism, and* $\ker(g) \subseteq \mathrm{Rad}(Q)$.

▶ **Proposition 16.** *A projective resolution, as in Definition 11, is minimal, according to Definition 13, if and only if $q_i\colon Q_i \to \ker(q_{i\text{-}1})$ is a projective cover for all $i \geq 0$. Similarly, a projective presentation, as in Definition 11, is minimal, according to Definition 13, if and only if $q_0 : Q_0 \to M$ and $q_1\colon Q_1 \to \ker(q_0)$ are projective covers.*

Proof. See Appendix A. ◀

Finally, in our setting, minimal resolutions always exist.

▶ **Proposition 17** (Existence of minimal projective resolutions). *Let M be a P-persistence module over a finite poset valued in finite-dimensional vector spaces. Then M has a minimal projective resolution. The minimal resolution is unique up to isomorphism, consists of finite direct sums of elementary projective modules, and its length is bounded by the longest chain in the poset P. Moreover, every minimal presentation is the restriction of a minimal projective resolution.*

Proof. See Theorem 5.8 and Corollary 5.10 in [2] as well as Proposition 2.2.6 in [31]. ◀

2.4 Projective implicit representations

To compute the persistent homology of a poset tower $K\colon P \to \mathbf{SCpx}$, we need to compute the homology of the chain complex of P-persistence modules $C_\bullet(K)$. For this task, it would be ideal if we had a projective resolution of the whole chain complex $C_\bullet(K)$, which can be defined as a chain complex of projective modules $Q_\bullet$ together with a morphism of chain complexes $f_\bullet\colon Q_\bullet \to C_\bullet(K)$:

$$\begin{array}{ccccccccccc}
0 & \longleftarrow & Q_0 & \xleftarrow{q_1} & Q_1 & \xleftarrow{q_2} & Q_2 & \xleftarrow{q_3} & \cdots \\
 & & \downarrow{\scriptstyle f_0} & & \downarrow{\scriptstyle f_1} & & \downarrow{\scriptstyle f_2} & & \\
0 & \longleftarrow & C_0(K) & \xleftarrow{\partial_1} & C_1(K) & \xleftarrow{\partial_2} & C_2(K) & \xleftarrow{\partial_3} & \cdots
\end{array}$$

such that $H_\ell(f_\bullet)\colon H_\ell(Q_\bullet) \to H_\ell\big(C_\bullet(K)\big)$ is an isomorphism for all $\ell \geq 0$. Computing such a resolution for arbitrary posets is a difficult problem. Fortunately, to compute the homology $H_\ell\big(C_\bullet(K)\big)$, we only need to replace the relevant segment of $C_\bullet(K)$ by projective modules.

▶ **Definition 18** (Projective implicit representation)**.** *Let*

$$C^S\colon \quad C_l \xleftarrow{\partial_c} C_c \xleftarrow{\partial_r} C_r \xleftarrow{0}$$

be a chain complex segment of P-persistence modules. We call a chain complex segment

$$D^S\colon \quad D_l \xleftarrow{d_c} D_c \xleftarrow{d_r} D_r \xleftarrow{0}$$

a projective implicit representation (PiRep) of $H(C^S) \coloneqq \ker(\partial_c)/\operatorname{im}(\partial_r)$ if D_x is projective for $x \in \{l, c, r\}$ and $H(C^S) \cong \ker(d_c)/\operatorname{im}(d_r) =: H(D^S)$.

In the following, we abbreviate projective implicit representation as *PiRep*.

▶ Remark 19. We note that we could also define a PiRep of $H(C^S)$ as a chain complex segment D^S together with maps:

$$\begin{array}{ccccccc} D^S\colon & D_l & \xleftarrow{d_c} & D_c & \xleftarrow{d_r} & D_r \\ {\scriptstyle f^S}\downarrow & \downarrow{\scriptstyle f_l} & & \downarrow{\scriptstyle f_c} & & \downarrow{\scriptstyle f_r} \\ C^S\colon & C_l & \xleftarrow{\partial_c} & C_c & \xleftarrow{\partial_r} & C_r \end{array}$$

such that the squares commute and $H(f^S)\colon H(D^S) \to H(C^S)$ is an isomorphism. This could be viewed as a "local" projective resolution and is more natural from a theoretical perspective. The PiReps we will construct in the following satisfy this stronger property. Nevertheless, we adopt Definition 18 to align with the notion of free implicit representations used in the TDA literature. We also note that the most prevalent construction of free implicit representations [9] does not satisfy this stronger property.

3 Constructing PiReps from projective resolutions

In this section, we establish the theoretical basis for our method to compute a PiRep of $C_{\ell\text{-}1}(K) \xleftarrow{\partial_\ell} C_\ell(K) \xleftarrow{\partial_{\ell+1}} C_{\ell+1}(K)$, as in Definition 18, from a poset tower K. Our method consists of two steps. In the first step, we compute degreewise projective resolutions of $C_\ell(K)$ up to the second term, together with lifts of the boundary maps ∂_ℓ to the resolutions up to the first term, as depicted in (9) below. In the second step, we assemble the partial resolutions and maps in (9), together with an additional correction term $\vartheta_{\ell+1}$, to obtain a PiRep of $H_\ell(K)$. We note that the results presented below hold for arbitrary chain complexes of P-persistence modules, but since we focus on poset towers, we present them in this context.

$$\begin{array}{ccccccccc} 0 & \longleftarrow & C_{\ell+1}(K) & \xleftarrow{\alpha_{\ell+1}} & G_{\ell+1} & \xleftarrow{p^1_{\ell+1}} & R_{\ell+1} & \xleftarrow{p^2_{\ell+1}} & RR_{\ell+1} \\ & & {\scriptstyle \partial_{\ell+1}}\downarrow & & {\scriptstyle f^0_{\ell+1}}\downarrow & \swarrow{\scriptstyle \vartheta_{\ell+1}} & \downarrow{\scriptstyle f^1_{\ell+1}} & & \\ 0 & \longleftarrow & C_\ell(K) & \xleftarrow{\alpha_\ell} & G_\ell & \xleftarrow{p^1_\ell} & R_\ell & \xleftarrow{p^2_\ell} & RR_\ell \\ & & {\scriptstyle \partial_\ell}\downarrow & & {\scriptstyle f^0_\ell}\downarrow & & \downarrow{\scriptstyle f^1_\ell} & & \\ 0 & \longleftarrow & C_{\ell\text{-}1}(K) & \xleftarrow{\alpha_{\ell-1}} & G_{\ell\text{-}1} & \xleftarrow{p^1_{\ell-1}} & R_{\ell\text{-}1} & \xleftarrow{p^2_{\ell-1}} & RR_{\ell\text{-}1} \end{array} \tag{9}$$

We use the following notation for the partial resolutions of $C_\ell(K)$, illustrated in the rows of (9). The zeroth term of the resolution is denoted by G_ℓ. It should be thought of

as the module of generators of $C_\ell(K)$. We will show in Section 5.1 that the set of simplex generators $\mathcal{S}_\ell$ of K gives rise to a minimal set of (projective) generators of $C_\ell(K)$. In the following, we will slightly abuse notation and also denote the projective generators of G_ℓ by g^x_σ. Hence, we obtain $G_\ell = \bigoplus_{g^x_\sigma \in \mathcal{S}_\ell} \mathrm{Proj}[x]$. This means that with the right choice of α_ℓ, as discussed in detail in Section 5.1, $\alpha_\ell \colon G_\ell \to C_\ell(K)$ defines the first step of a minimal resolution (see Proposition 38). The first term of the resolution is denoted by R_ℓ and should be thought of as the module of relations of the generators G_ℓ of $C_\ell(K)$. We will denote its projective generators by r^x and simply call them relations. The second term of the resolution is denoted by RR_ℓ and should be thought of as the relations of relations R_ℓ of $C_\ell(K)$. We denote its projective generators by rr^x and call them relations of relations.

Figure 9 shows an example of a poset tower and (10) shows the partial resolutions of $C_\ell(K)$ together with the lifts of the boundary maps, as in (9).

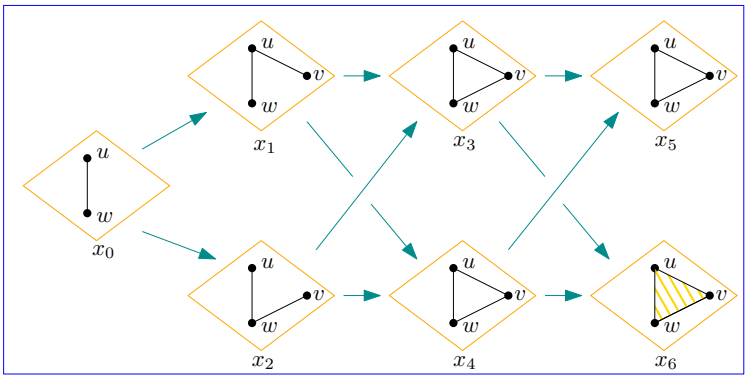


Figure 9 A poset tower over $P = \{x_0, \ldots, x_6\}$ where all the maps are inclusions. Every simplex has a unique generator except v, which appears at x_1 and x_2. The two copies of v get identified at x_3 and x_4 by relations $r^{x_3}, r^{x_4} \mapsto g_v^{x_1} + g_v^{x_2}$. These two relations become the same at x_5 and x_6, leading to relations of relations $rr^{x_5}, rr^{x_6} \mapsto r^{x_3} + r^{x_4}$.

$$
\begin{array}{ccccccc}
 & & g_{uvw}^{x_6} & & & & \\
 & & \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} & & & & \\
 & & g_{uw}^{x_0} \\ & & g_{uv}^{x_1} \\ & & g_{vw}^{x_2} & & & & \\
 & & G_2 & \xleftarrow{p_2^1} & 0 & \xleftarrow{p_2^2} & 0 \\
 & & \downarrow{\scriptstyle f_2^0} & & \downarrow{\scriptstyle f_2^1} & & \\
 & g_{uw}^{x_0} \; g_{uv}^{x_1} \; g_{vw}^{x_2} & G_1 & \xleftarrow{p_1^1} & 0 & \xleftarrow{p_1^2} & 0 \\
g_u^{x_0} & \begin{pmatrix} 1 & 1 & 0 \end{pmatrix} & \downarrow{\scriptstyle f_1^0} & & \downarrow{\scriptstyle f_1^1} & & \\
g_w^{x_0} & \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \\
g_v^{x_1} & \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \\
g_v^{x_2} & \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} & G_0 & \xleftarrow{p_0^1} & R_0 & \xleftarrow{p_0^2} & RR_0 \\
 & & r^{x_3} \; r^{x_4} & & & rr^{x_5} \; rr^{x_6} & \\
g_u^{x_0} & \begin{pmatrix} 0 & 0 \end{pmatrix} & & & r^{x_3} & \begin{pmatrix} 1 & 1 \end{pmatrix} \\
g_w^{x_0} & \begin{pmatrix} 0 & 0 \end{pmatrix} & & & r^{x_4} & \begin{pmatrix} 1 & 1 \end{pmatrix} \\
g_v^{x_1} & \begin{pmatrix} 1 & 1 \end{pmatrix} \\
g_v^{x_2} & \begin{pmatrix} 1 & 1 \end{pmatrix}
\end{array}
\tag{10}
$$

We now need to assemble the projective modules and matrices in (9) in such a way that we obtain a PiRep of $H_\ell(K) = \ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$. We note that this is more or less standard homological algebra, but we are not aware of a good reference explaining this process. We provide proofs of the results in this section, omitted from the main text, in Appendix B.

We start by discussing how to obtain $\ker(\partial_\ell)$ from the resolutions. The running example in Figure 9 and (10) illustrates the intricacies of this process. As one can see, the matrix f_1^0 representing ∂_1 on the generators has no kernel. The image of the generators representing the triangle uvw is $f_1^0\big(g_{uw}^{x_0} + g_{uv}^{x_1} + g_{vw}^{x_2}\big) = g_v^{x_1} + g_v^{x_2}$. This is because at x_1 the only available generator for the boundary vertex v of the edge uv is $g_v^{x_1}$ whereas at x_2 only $g_v^{x_2}$ is available for the boundary of the edge vw. Hence, the kernel of f_1^0 is insufficient to represent the kernel of ∂_1, which is the P-persistence module $\ker(\partial_1)\colon P \to \mathbf{Vec}$ that is grade-wise generated by the cycles $\ker\big(\partial_1(x)\big)$ in Figure 9. To get the correct kernel, we have to recognize that at x_3 and x_4, the two copies of v get identified, which is witnessed by the relations r^{x_3} and r^{x_4}. The boundary of the triangle is zero modulo these relations. To account for this, we define the morphism:

$$\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix} : G_\ell \oplus R_{\ell\text{-}1} \to G_{\ell\text{-}1} \tag{11}$$

whose kernel serves as the generators of $\ker(\partial_\ell)$. In the example, we obtain the matrix:

$$\begin{pmatrix} f_1^0 & p_0^1 \end{pmatrix} = \begin{array}{c} \\ g_u^{x_0} \\ g_w^{x_0} \\ g_v^{x_1} \\ g_v^{x_2} \end{array} \begin{array}{c} \begin{array}{ccccc} g_{uw}^{x_0} & g_{uv}^{x_1} & g_{vw}^{x_2} & r^{x_3} & r^{x_4} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \tag{12}$$

whose kernel is generated by $g_{uw}^{x_0} + g_{uv}^{x_1} + g_{vw}^{x_2} + r^{x_3}$ and $g_{uw}^{x_0} + g_{uv}^{x_1} + g_{vw}^{x_2} + r^{x_4}$. This kernel represents the triangles at x_3 and x_4 in Figure 9. A problem ensues because these two triangles become the same triangle at x_5 and x_6. Thus, the kernel of ∂_1 has relations at x_5 and x_6. Since there are no relations between the edges, these relations have to come from relations on r^{x_3} and r^{x_4}. In general, relations on the kernel can, of course, also come from relations on the edges. We account for the relations in G_ℓ and $R_{\ell\text{-}1}$ by defining the following morphism:

$$\begin{pmatrix} p_\ell^1 & 0 \\ f_\ell^1 & p_{\ell\text{-}1}^2 \end{pmatrix} : R_\ell \oplus RR_{\ell\text{-}1} \to \ker\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix} \tag{13}$$

from the relations of the domain of (11) to the kernel of $\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix}$ where we additionally include the morphism $f_\ell^1\colon R_\ell \to R_{\ell\text{-}1}$ from (9). The morphism f_ℓ^1 is needed to make (13) well-defined. Since

$$\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix} \circ \begin{pmatrix} p_\ell^1 & 0 \\ f_\ell^1 & p_{\ell\text{-}1}^2 \end{pmatrix} = \begin{pmatrix} f_\ell^0 \circ p_\ell^1 + p_{\ell\text{-}1}^1 \circ f_\ell^1 & p_{\ell\text{-}1}^1 \circ p_{\ell\text{-}1}^2 \end{pmatrix} = 0,$$

we obtain a well-defined map to the kernel. In our running example, the matrix of relations on the kernel is just p_0^2, which leads to the identification of the two generators of the kernel at x_5 and x_6.

▶ **Proposition 20.** *The following sequence is exact:*

$$0 \longleftarrow \ker(\partial_\ell) \xleftarrow{\begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}} \ker\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix} \xleftarrow{\begin{pmatrix} p_\ell^1 & 0 \\ f_\ell^1 & p_{\ell\text{-}1}^2 \end{pmatrix}} R_\ell \oplus RR_{\ell\text{-}1}. \tag{14}$$

Proof. See Appendix B. ◀

Proposition 20 shows that $\ker(\partial_\ell)$ is the cokernel of (13), but this is not a presentation since $\ker \begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix}$ is not necessarily projective for general P.

To obtain a representation of the homology $\ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$, we have to construct a morphism representing $\partial_{\ell+1} \colon C_{\ell+1} \to \ker(\partial_\ell)$. The morphism $\partial_{\ell+1}$ is represented by $f_{\ell+1}^0$ on the generators but, as one can see in (10) and (12), for the poset tower in Figure 9, f_2^0 does not map to the kernel of $\begin{pmatrix} f_1^0 & p_0^1 \end{pmatrix}$. The problem is that $f_1^0 \circ f_2^0 \neq 0$ because the boundary of the triangle is only zero modulo relations. In other words, the morphisms $(f_\ell^0)_\ell$ on the generators do not form a chain complex. However, because they lift the differentials $(\partial_\ell)_\ell$, which compose to zero, they compose to zero up to homotopy. By Definition 11(2), $f_\ell^0 \circ f_{\ell+1}^0$ is a lift of $\partial_\ell \circ \partial_{\ell+1} = 0$ to the generators. Proposition 12(3) guarantees that we can extend these partial resolutions and lifts to full resolutions and lifts $f_\ell \circ f_{\ell+1}$. Obviously, the zero map is also a lift of $\partial_\ell \circ \partial_{\ell+1} = 0$ to these resolutions. Proposition 12(2) then implies that $f_\ell \circ f_{\ell+1}$ and 0 are homotopic, which means that there exists a morphism $\vartheta_{\ell+1} \colon G_{\ell+1} \to R_{\ell\text{-}1}$, illustrated by the dashed arrow in (9), such that $p_{\ell\text{-}1}^1 \circ \vartheta_{\ell+1} = f_\ell^0 \circ f_{\ell+1}^0 + 0 = f_\ell^0 \circ f_{\ell+1}^0$. This motivates the definition of the following morphism:

$$\begin{pmatrix} f_{\ell+1}^0 \\ \vartheta_{\ell+1} \end{pmatrix} : G_{\ell+1} \to \ker \begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix} \tag{15}$$

representing $\partial_{\ell+1}$. Again, the additional morphism $\vartheta_{\ell+1}$ is needed to make the map well-defined. Since

$$\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix} \circ \begin{pmatrix} f_{\ell+1}^0 \\ \vartheta_{\ell+1} \end{pmatrix} = f_\ell^0 \circ f_{\ell+1}^0 + p_{\ell\text{-}1}^1 \circ \vartheta_{\ell+1} = 0,$$

we obtain a well-defined map to the kernel. In the running example (Figure 9), this homotopy for $f_1^0 \circ f_2^0$ can be chosen as

$$\vartheta_2 = \begin{matrix} \\ r^{x_3} \\ r^{x_4} \end{matrix} \begin{matrix} g_{uvw}^{x_6} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{matrix}$$

sending the triangle to a relation annihilating its boundary. We now combine (13) and (15) to represent $\ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$. Let $\pi \colon \ker(\partial_\ell) \to \ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$ be the projection to the cokernel.

▶ **Proposition 21.** *The following sequence is exact:*

$$0 \longleftarrow \ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1}) \xleftarrow{\pi \circ \begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}} \ker \begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix} \xleftarrow{\begin{pmatrix} f_{\ell+1}^0 & p_\ell^1 & 0 \\ \vartheta_{\ell+1} & f_\ell^1 & p_{\ell\text{-}1}^2 \end{pmatrix}} G_{\ell+1} \oplus R_\ell \oplus RR_{\ell\text{-}1}.$$

Proof. See Appendix B. ◀

Proposition 21 implies that $\ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$ is isomorphic to the cokernel of the matrix built from (13) and (15). The exact sequence in Proposition 21 is already a presentation of the homology $\ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$ if $\ker \begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix}$ is projective. Again, note that this is not necessarily the case for general posets. In any case, we obtain a PiRep of the homology as stated in the following theorem.

▶ **Theorem 22.** *The following diagram*

$$G_{\ell\text{-}1} \xleftarrow{\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix}} G_\ell \oplus R_{\ell\text{-}1} \xleftarrow{\begin{pmatrix} f_{\ell+1}^0 & p_\ell^1 & 0 \\ \vartheta_{\ell+1} & f_\ell^1 & p_{\ell\text{-}1}^2 \end{pmatrix}} G_{\ell+1} \oplus R_\ell \oplus RR_{\ell\text{-}1} \tag{16}$$

is a projective implicit representation of $\ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$.

Proof. This follows directly from Proposition 21. ◀

In the case where $\ker\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix}$ is not projective, computing a presentation of the homology $\ker(\partial_\ell)/\operatorname{im}(\partial_{\ell+1})$ from a PiRep, as in (16), requires the computation of a projective resolution of $\ker\begin{pmatrix} f_\ell^0 & p_{\ell\text{-}1}^1 \end{pmatrix}$. Over a general poset, this is not trivial but there are available algorithms for this task. For completeness, we will discuss this part in Appendix D.

In our example, we obtain the following PiRep of $H_1(K)$:

$$\begin{array}{c} \\ g_u^{x_0} \\ g_w^{x_0} \\ g_v^{x_1} \\ g_v^{x_2} \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} g_{uw}^{x_0} & g_{uv}^{x_1} & g_{vw}^{x_2} & r^{x_3} & r^{x_4} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ g_{uw}^{x_0} \\ g_{uv}^{x_1} \\ g_{vw}^{x_2} \\ r^{x_3} \\ r^{x_4} \end{array}\!\!\begin{array}{c} \begin{array}{ccc} g_{uvw}^{x_6} & rr^{x_5} & rr^{x_6} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

$$G_0 \longleftarrow G_1 \oplus R_0 \longleftarrow G_2 \oplus RR_0$$

In this case, the kernel of $\begin{pmatrix} f_1^0 & p_0^1 \end{pmatrix}$ is already projective and, after a change of basis, denoted by $\cong$, we obtain the following presentation of the homology:

$$\begin{pmatrix} f_2^0 & p_1^1 & 0 \\ \vartheta_2 & f_1^1 & p_0^2 \end{pmatrix}\Big|_{\ker\begin{pmatrix} f_1^0 & p_0^1 \end{pmatrix}} = \begin{array}{c} \\ r^{x_3} \\ r^{x_4} \end{array}\!\!\begin{array}{c} \begin{array}{ccc} g_{uvw}^{x_6} & rr^{x_5} & rr^{x_6} \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \cong \begin{array}{c} \\ r^{x_3} \\ r^{x_4} \end{array}\!\!\begin{array}{c} \begin{array}{ccc} rr^{x_5} & g_{uvw}^{x_6} & rr^{x_6} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array}.$$

It is generated by the two triangular cycles appearing at x_3 and x_4 in Figure 9 which get identified at x_5 and killed at x_6 by the triangle.

▶ Remark 23. On a high level, the construction of this section is a truncated version of a homotopical method that computes a projective resolution of an entire chain complex, as discussed in Section 2.4, from projective resolutions of its individual terms. Let $C_\bullet$ be a bounded-below chain complex, $C_\ell \leftarrow Q_\bullet^\ell$ be projective resolutions, and $f_\bullet^\ell \colon Q_\bullet^\ell \to Q_\bullet^{\ell\text{-}1}$ be lifts of the boundary maps $\partial_\ell \colon C_\ell \to C_{\ell\text{-}1}$. If consecutive lifts would compose to zero, $Q_\bullet^\bullet$ would form a double complex whose total complex is quasi-isomorphic to $C_\bullet$; see [33] Section 2.7. But for fixed resolutions $Q_\bullet^\bullet$, it may be impossible to choose lifts with this property; see for example (10). Proposition 12 guarantees that compositions of these lifts are homotopic to zero. Thus, the resolutions and lifts form a chain complex in the homotopy category.

However, to construct a total complex, these homotopies must be retained explicitly as additional data. Moreover, these homotopies are also only compatible up to a choice of higher homotopy. The appropriate structure is therefore a homotopy-coherent analogue of a double complex that records all the higher homotopical structure. Its homotopy-coherent total complex includes the resolutions and the lifted boundary maps as well as all contributions from the chosen homotopies and higher homotopies and is quasi-isomorphic to $C_\bullet$. This homotopy-coherent total complex construction is discussed in [30] in a different setting. The existence of this homotopy coherent structure in our setting follows from the fact that the homotopies guaranteed by Proposition 12 are again unique up to higher homotopy and the analogous uniqueness holds at every higher level.

This homotopy coherent approach is different from the more well-known Cartan-Eilenberg method; see [33] Section 5.7. That construction uses projective resolutions of the boundary and homology modules of $C_\bullet$ to obtain compatible resolutions that directly form a double complex. But this is not suitable for our purpose because it requires resolving the homology

modules first. Our aim is to compute a PiRep to facilitate an efficient global homology computation.

Because we only focus on a segment of $C_\bullet$, we only need to compute degreewise resolutions up to the second term and we only have to incorporate one component of the degree one homotopies. Truncating the chain complex allows us to ignore the higher order structure which would account for longer range interactions.

4 PiRep algorithm

4.1 Input, output, and data structures

In this section, we describe an efficient algorithm to compute a PiRep of $H_\ell(K)$, as in Theorem 22, from a poset tower $K\colon P \to \mathbf{SCpx}$. The PiRep consists of the two block matrices in (16), whose entries come from projective resolutions of $C_\ell(K)$ up to the second term, lifts of the boundary maps ∂_ℓ up to the first term, and a correction map $\vartheta_{\ell+1}$, as depicted in (9). Therefore, our algorithm has to compute matrix representations of the maps p^1_ℓ, p^2_ℓ representing the resolutions of $C_\ell(K)$, f^0_ℓ, f^1_ℓ representing the lifts of ∂_ℓ, and $\vartheta_{\ell+1}$. Instead of performing an algebraic brute-force computation, we make essential use of the special structure of the modules $C_\ell(K)$ and morphisms ∂_ℓ to compute all these maps in a combinatorial fashion from the following representation of the poset tower K.

Input: A poset tower $K\colon P \to \mathbf{SCpx}$ represented by:
- The Hasse diagram of P as a directed (transitively reduced) graph.
- A list $\mathcal{S} = \{g^{x_1}_{\sigma_1}, \dots, g^{x_{n'}}_{\sigma_{n'}}\}$ of simplex generators.
- A list $\mathcal{C} = \{c^{y_1 \prec z_1}_{v_{i_1} \mapsto v_{j_1}}, \dots, c^{y_{n''} \prec z_{n''}}_{v_{i_{n''}} \mapsto v_{j_{n''}}}\}$ of edge events.

The complexity of the proposed algorithm is measured with respect to the input size, which includes (i) $t = t_0 + t_1$, the number of vertices (t_0) and edges (t_1) of the graph representing P, (ii) $n = n' + n''$, the number of simplex generators in $\mathcal{S}$ plus the number of edge events in $\mathcal{C}$. Additionally, we assume that the maximal simplex dimension is constant.

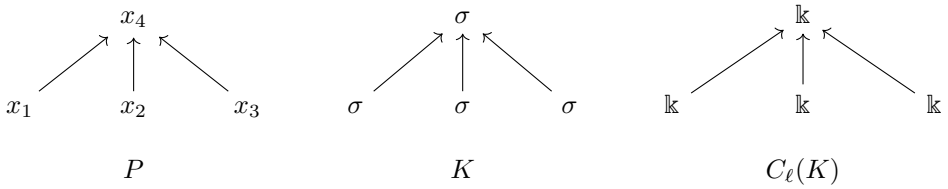
Output: P-graded sparse matrix representations (containing only the non-zero entries) of

$$\begin{pmatrix} f^0_\ell & p^1_{\ell-1} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} f^0_{\ell+1} & p^1_\ell & 0 \\ \vartheta_{\ell+1} & f^1_\ell & p^2_{\ell-1} \end{pmatrix}.$$

As already discussed in Section 3, the set of simplex generators $\mathcal{S}_\ell$ gives rise to a minimal set of generators of $C_\ell(K)$ allowing us to define $G_\ell \coloneqq \bigoplus_{g^x_\sigma \in \mathcal{S}_\ell} \mathrm{Proj}[x]$ (see Proposition 38). Given this set of generators, we have to determine how they are related. A presentation of $C_\ell(K)$ based on G_ℓ can be built using the following three kinds of relations. The fact that these three types of relations are sufficient to form a complete presentation is a consequence of the proof of correctness of our algorithm in Section 5.1, which constructs the presentation by adding exactly these relations.

1. Two generators g^x_σ and g^y_σ of the same simplex σ born at different grades can join at a grade $b > x, y$ where they must be identified by a relation $r^b \mapsto g^x_\sigma + g^y_\sigma$.
2. An ℓ-simplex $\sigma \in K(a)$ represented by a generator g^x_σ can be collapsed into an ℓ-simplex $\tau \in K(b)$ represented by a generator g^y_τ via a simplicial map such that $K(a < b)(\sigma) = \tau$. In this case, they also have to be identified by a relation $r^b \mapsto g^x_\sigma + g^y_\tau$.
3. An ℓ-simplex $\sigma \in K(a)$ represented by a generator g^x_σ can be collapsed into a simplex $\tau \in K(b)$ of dimension $< \ell$ via a simplicial map such that $K(a < b)(\sigma) = \tau$. In this case, the generator has to be killed by a relation $r^b \mapsto g^x_\sigma$.

To provide some intuition why generators in a poset tower are always related in pairs, we look at the example in (17), where three copies of the same simplex σ join at x_4.



(17)

These three copies cannot be related by a single relation at x_4 because the presentation matrix in (18) on the right would induce a two-dimensional cokernel at x_4. Instead, they have to be related in pairs as shown in (18) on the left.

$$p_\ell^1 = \begin{matrix} \\ g_\sigma^{x_1} \\ g_\sigma^{x_2} \\ g_\sigma^{x_3} \end{matrix} \begin{matrix} \begin{matrix} r^{x_4} & r^{x_4} \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \end{matrix} \quad , \qquad p_\ell^1 \neq \begin{matrix} \\ g_\sigma^{x_1} \\ g_\sigma^{x_2} \\ g_\sigma^{x_3} \end{matrix} \begin{matrix} \begin{matrix} r^{x_4} \end{matrix} \\ \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \end{matrix} \tag{18}$$

The complexity of finding these relations arises because they depend on each other. Two generators g_σ^x and g_σ^y should not be identified at a common join if one of them is already killed by a relation at this point. In this case, they are different simplices. However, we also cannot just treat them as different simplices because they might have to be identified at some other join. Moreover, many generators can meet multiple times at multiple points along the poset tower. Therefore, we need an efficient method to keep track of which generators have to be related, which simplices are already related, and which simplices are already dead.

Graph structure. Our key observation is that we can model the relations by a $\overline{P}$-filtered graph, where $\overline{P} = P \cup \{-\infty\}$ such that $-\infty < x$ for all $x \in P$. The generators in a poset tower always get related in pairs (Cases 1 and 2) except in the case of simplices getting totally collapsed (Case 3). But we can model the third case by a generator that gets related to a distinguished "graveyard" vertex Ω at grade $-\infty$. Together with this distinguished vertex Ω, we obtain a $\overline{P}$-filtered graph $\mathcal{G}\colon \overline{P} \to \mathbf{\Delta Cpx}_{\leq 1}$ (Definition 39) whose other vertices are the generators $g_\sigma^x \in \mathcal{S}$, born at grade x, and whose edges are the relations $r^b \mapsto g_\sigma^x + g_\tau^y$ or $r^b \mapsto g_\sigma^x + \Omega$, born at grade b. In particular, each vertex and edge of the $\overline{P}$-filtered graph $\mathcal{G}$ is born at a unique grade in $\overline{P}$. Therefore, we can directly represent the boundary matrix of $\mathcal{G}$ by a $\overline{P}$-graded matrix. We formally define $\mathcal{G}$ as a one-dimensional $\overline{P}$-filtered Δ-complex, because there can be multiple edges between two vertices. The following theorem is a direct consequence of the proof of correctness, in Section 5, of the upcoming algorithm PiRep (see Definition 39 and Theorem 45).

▶ **Theorem 24.** *Let $K\colon P \to \mathbf{SCpx}$ be a poset tower. Then there exists a $\overline{P}$-filtered graph $\mathcal{G}\colon \overline{P} \to \mathbf{\Delta Cpx}_{\leq 1}$, where each vertex and edge is born at a unique grade $x \in \overline{P}$, with $\overline{P}$-graded boundary matrix $\overline{p}_\ell^1$ such that the matrix p_ℓ^1, obtained from $\overline{p}_\ell^1$ by removing the row corresponding to a distinguished vertex Ω at $-\infty$, is a minimal presentation of $C_\ell(K)$.*

At every grade $x \in P$, the connected component of Ω in $\mathcal{G}(x)$ represents the simplices that already died. Each remaining connected component represents a simplex in $K(x)$. Moreover, the cycles in $\mathcal{G}$ represent the kernel of the presentation matrix p_ℓ^1 which corresponds to the relations of the relations (see Definition 11(1)). In the following algorithm PiRep, we successively build the graph $\mathcal{G}$ by adding generators and relations between them in the form

of vertices and edges. If a new relation is added that forms a cycle in $\mathcal{G}(x)$, then this new relation is a linear combination of other existing relations, which should not be added to the minimal presentation (see Definition 15). Nevertheless, a cycle that is generated by the inclusion of edges from predecessors implies a relation of relations.

The special structure of p^1_ℓ is also crucial for the efficient computation of f^1_ℓ and $\vartheta_{\ell+1}$. These morphisms are determined by the equations $p^1_{\ell\text{-}1} \circ f^1_\ell = f^0_\ell \circ p^1_\ell$ and $p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1} = f^0_\ell \circ f^0_{\ell+1}$, i.e., by P-graded linear systems as in Definition 9. By Proposition 10, for a specific column of f^1_ℓ, corresponding to a relation r^x, or a specific column of $\vartheta_{\ell+1}$, corresponding to a generator g^x_σ, these equations can be solved via a standard system of linear equations, where the coefficient matrix is the boundary matrix of $\mathcal{G}(x)$. Using this graph structure, we can solve these systems in linear time.

Matrices. Our algorithm computes matrix representations of p^1_ℓ, p^2_ℓ, f^0_ℓ, f^1_ℓ, and $\vartheta_{\ell+1}$. All matrices are implemented with a sparse representation that contains only non-zero entries. As a result, the space and time complexities for computing and representing them may be less than their actual size given by the product of their row and column dimensions. While analyzing the time complexities of the algorithms, we assume that the matrices are represented sparsely.

The rows of p^1_ℓ and the rows and columns of f^0_ℓ are associated to generators g^x_σ, i.e., we maintain a bijective map $I \to \{g^x_\sigma\}$ for the index set I for rows. Similarly, the columns of p^1_ℓ, the rows and columns of f^1_ℓ, and the rows of p^2_ℓ are associated to relations r^x and are indexed by a graded identifier. This identifier is the column index paired with the grade x if the column is added at the grade x. In other words, denoting this identifier as r^x, we maintain a bijective map $J \to \{\mathsf{r}^x\}$ for the column index set J. Therefore, we write $p^1_\ell[g^x_\sigma, \mathsf{r}^y]$ for $p^1_\ell[i,j]$ if $i \mapsto g^x_\sigma$ and $j \mapsto \mathsf{r}^y$. We use this notation to address entries of a matrix in our algorithm to simplify presentation even though the matrix is implemented with a sparse representation. The columns of the matrix p^2_ℓ are associated to relations of relations and are indexed by a graded global counter rr. A column gets the identifier rr^x if the column is generated at grade x. We write $p^2_\ell[\mathsf{r}^x, \mathsf{rr}^y]$ in place of $p^2_\ell[i,j]$ if $i \mapsto \mathsf{r}^x$ and $j \mapsto \mathsf{rr}^y$.

In what follows, the algorithms handle all degrees $\ell \geq 0$ together and thus maintain single matrices p^1, p^2, f^0, f^1, and ϑ accounting for all degrees.

The main routine PiRep computes a linear extension $x_0, \ldots, x_{t_0\text{-}1}$ of the poset P. Then it iterates over each grade x_i maintaining the invariant that, for every degree $\ell \geq 0$, it has correctly computed matrix representations of the maps p^1_ℓ, p^2_ℓ, f^0_ℓ, f^1_ℓ, and $\vartheta_{\ell+1}$ restricted to the subposet induced by grades $x_0, \ldots, x_i$. We note that the choice of linear extension is arbitrary, and different choices may yield different output matrices. However, since the matrix representations of (even minimal) presentations and projective resolutions of modules are generally not unique, this behavior is expected. In Section 5, we prove that any choice of linear extension produces a correct PiRep.

The algorithm maintains a set $L^{x_i}_{act}$ of active generators, i.e., a set of pairs (σ, g^y_τ) such that the generator g^y_τ represents the simplex σ at the grade x_i, and a forest $\mathcal{F}^{x_i}_{act}$ of active relations. The set $L^{x_i}_{act}$ helps to determine the relations R_ℓ (columns of p^1_ℓ) that may arise due to vertex collapses handled by the subroutine Collapse or by identification of generators handled by the subroutine Generator. The forest $\mathcal{F}^{x_i}_{act}$, initialized with a special vertex Ω, helps to determine superfluous relations in R_ℓ and thus helps guarantee minimality of p^1_ℓ and also to determine relations of relations RR_ℓ registered in the matrix p^2_ℓ by the subroutine RelRel. The special vertex Ω plays the role of a "graveyard" vertex, which gets attached to any edge in the relation graph generated by a "vanishing" simplex due to collapse.

The subroutines take the arguments in the parentheses as input and update the global

matrices p^1, p^2, f^0, f^1, and ϑ along with other global structures such as an active list $L^\bullet_{act}$ and graphs $\mathcal{G}^\bullet_{act}$ and $\mathcal{F}^\bullet_{act}$ as part of their output.

4.2 Algorithm

Algorithm PiRep $(P, \mathcal{C}, \mathcal{S})$

- Compute a linear extension $x_0, \ldots, x_{t_0\text{-}1}$ of P;
- Initialize empty matrices p^1, p^2, f^0, f^1, ϑ, and counters $\mathsf{r} = 0$, $\mathsf{rr} = 0$;
- For $i := 0$ to $t_0 - 1$ do
  1. Call Collapse$(P, \mathcal{C}, x_i)$;
  2. Call Generator$(P, \mathcal{S}, x_i)$;
  3. Call Lift(x_i);
  4. Call RelRel(x_i)

In the routine Collapse, for every poset edge $y \prec x$, we handle collapsing a vertex v to a vertex w. In step 1, we initialize an empty list of active generators L^x_{act} which we keep associated with the grade x for subsequent use. In step 2, we compute the union of forests $\cup_{y\prec x}\mathcal{F}^y_{act}$ resulting in a graph $\mathcal{G}^x_{act}$ that may not be a forest. This graph (relation graph) is used and updated in the successive steps and routines (Figure 10). In step 3(a), every simplex $\sigma \in K(y)$ gets its vertices updated either remaining the same $(v \mapsto v)$ or renamed to w $(v \mapsto w)$; e.g. in Figure 7, the edges uv, uw and vw at x_4 become uw, uw and ww at x_5, respectively.

If a simplex σ in $K(y)$ after vertex renaming contains none or only one of v and w, it is included in L^x_{act} with the appropriate generator (step 3(b.i)); e.g. uv from x_4 becoming uw at x_5 is added to $L^{x_5}_{act}$ with generator $g^{x_1}_{uv}$. Similarly, the edge uw from x_4 remaining uw at x_5 is added with the generator $g^{x_2}_{uw}$. Now, $L^{x_5}_{act}$ has two copies of uw with two different generators. They are identified by the routine Generator.

Otherwise, in step 3(b.ii), the vanishing simplex σ, such as the edge vw becoming ww, is not included in L^x_{act}; instead, a relation is generated for p^1 unless the addition of this relation to the graph $\mathcal{G}^{x_i}_{act}$ generates a cycle. For the vanishing edge vw, we add the edge $e^{x_5}_{(\Omega, g^{x_3}_{vw})}$ to the graph $\mathcal{G}^{x_5}_{act}$ which does not create a cycle (Figure 10 (bottom right)). Therefore, we create a column indexed by r^{x_5} for the relation $r^{x_5}_1 \mapsto g^{x_3}_{vw}$ setting $p^1[g^{x_3}_{vw}, \mathsf{r}^{x_5}] = 1$.

Algorithm Collapse $(P, \mathcal{C}, x)$

1. Initialize $L^x_{act} := \emptyset$ and keep it associated with grade x;
2. If x is not minimal in P, compute $\mathcal{G}^x_{act} := \cup_{y\prec x}\mathcal{F}^y_{act}$, else initialize $\mathcal{G}^x_{act}$ with vertex Ω;
3. For all $y \prec x$ do
   - **a.** Compute $K(y \prec x)$ as $v \mapsto w$ if $c^{y\prec x}_{v\mapsto w} \in \mathcal{C}$ and as $v \mapsto v$ else;
   - **b.** For all $(\sigma, g^z_\rho) \in L^y_{act}$ do
     - **i.** If $K(y \prec x)(\sigma)$ has no repeated vertices
       - Add $\big(K(y \prec x)(\sigma), g^z_\rho\big)$ to L^x_{act};
     - **ii.** Otherwise, if adding $e^x_{(\Omega, g^z_\rho)}$ to $\mathcal{G}^x_{act}$ does not make a cycle
       - Add edge $e^x_{(\Omega, g^z_\rho)}$ to $\mathcal{G}^x_{act}$;
       - Add a column with index r^x in p^1 with $p^1[g^z_\rho, \mathsf{r}^x] = 1$; $\mathsf{r{+}{+}}$

In the routine Generator, we add a pair (σ, g^x_σ) to the active set L^x_{act} if σ has a generator at the current grade x. After adding a new generator g^x_σ, we record its boundary map in f^0 by calling the routine Boundary (step 1). In Boundary, we find active generators for the boundary of σ and map the new generator of σ to these boundary generators. Because we add simplices in order of increasing simplex dimension and because each simplex is either

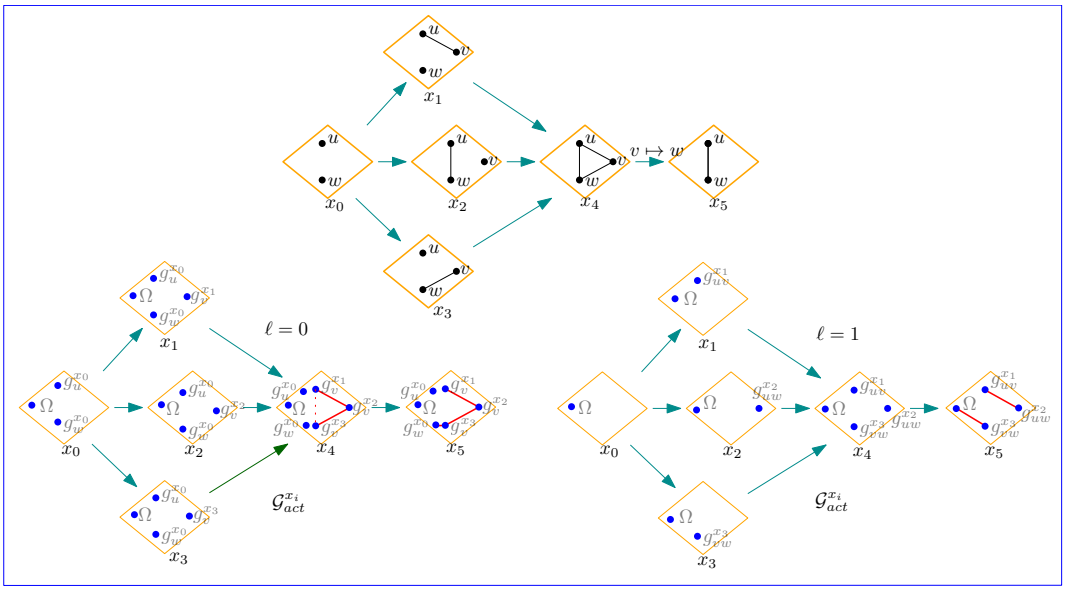


Figure 10 (Figure 7 example) Relation graphs $\mathcal{G}_{act}^{x_i}$ for vertices ($\ell = 0$) on left and for edges ($\ell = 1$) on right.

generated at grade x or is the image of a simplex in a predecessor, at this point, each boundary simplex has at least one active generator. If the active set contains a simplex σ with two different generators, they are identified by a relation. If this relation does not create a cycle in the relation graph $\mathcal{G}_{act}^{x}$, we add it to both $\mathcal{G}_{act}^{x}$ and p^1. Otherwise, we ignore it as a superfluous relation.

Again, consider the example in Figure 7. First, consider the case for edges ($\ell = 1$). At x_5, we have $(uw, g_{uv}^{x_1})$ and $(uw, g_{uw}^{x_2})$ in $L_{act}^{x_5}$ from the routine Collapse. We identify the two copies of uw, and add the edge $e_{(g_{uv}^{x_1}, g_{uw}^{x_2})}^{x_5}$ to the graph $\mathcal{G}_{act}^{x_5}$ whose addition does not create a cycle; see Figure 10 (bottom right). Therefore, we generate a relation $r_2^{x_5} \mapsto g_{uv}^{x_1} + g_{uw}^{x_2}$ (step 2(a)), and delete, say, $(uw, g_{uw}^{x_2})$ (step 2(b)). A column for $r_2^{x_5}$ is added to p^1.

Next, consider the case for vertices ($\ell = 0$). At x_4, three generators $g_v^{x_1}$, $g_v^{x_2}$, and $g_v^{x_3}$ of v generate the pairs $(v, g_v^{x_1}), (v, g_v^{x_2})$, and $(v, g_v^{x_3})$ in $L_{act}^{x_4}$ after the Collapse call. Step 2 potentially creates three relations $r_1^{x_4} \mapsto g_v^{x_1} + g_v^{x_2}$, $r_2^{x_4} \mapsto g_v^{x_2} + g_v^{x_3}$, and $r_3^{x_4} \mapsto g_v^{x_1} + g_v^{x_3}$. Assuming that edges for relations $r_1^{x_4}$ and $r_2^{x_4}$ are added first in $\mathcal{G}_{act}^{x_4}$ (Figure 10 (bottom left)), the edge for $r_3^{x_4}$ (shown dotted) creates a cycle. So, it is not added to $\mathcal{G}_{act}^{x_4}$, and equivalently only columns for relations $r_1^{x_4}$ and $r_2^{x_4}$ are added to p^1.

At the end (step 3), we compute a spanning forest $\mathcal{F}_{act}^{x}$ of $\mathcal{G}_{act}^{x}$ which we keep associated with the grade x for subsequent use. In Figure 10 (bottom left), $\mathcal{F}_{act}^{x}$ happens to be the same as $\mathcal{G}_{act}^{x}$ for every x. However, it is possible that, at a grade x, relation forests from predecessors come together to form a cycle, see e.g. grade x_4 in Figure 11. In this case, the relation forest and the relation graph would be different and the edges $(e_{(g_\sigma^{y_1}, g_\sigma^{y_2})}, e_{(g_\sigma^{y_2}, \Omega)})$ not included in the forest would give rise to relations of relations, adding two columns to p^2.

Algorithm Generator $(P, \mathcal{S}, x)$

1. For all $g_\sigma^x \in \mathcal{S}$ (in increasing order of simplex dimension) do
   - Add (σ, g_σ^x) to L_{act}^x;
   - Add vertex g_σ^x to $\mathcal{G}_{act}^x$;
   - Call Boundary(x, σ);

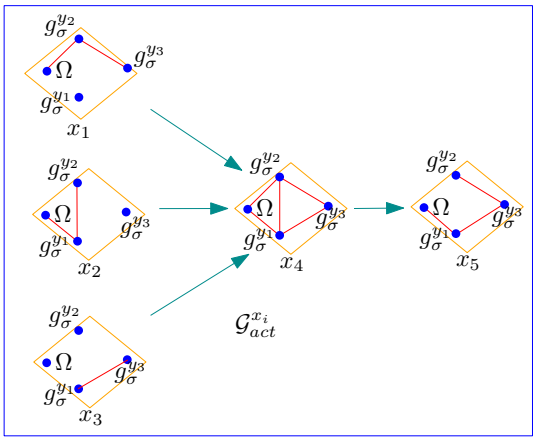


Figure 11 Relation graphs $\mathcal{G}_{act}^{x_i}$; poset tower not shown; the relation forest $\mathcal{F}_{act}^{x_4}$, which is passed to x_5, is a proper subgraph of $\mathcal{G}_{act}^{x_4}$.

2. For every $((\sigma, g_\tau^y), (\sigma, g_\rho^z) \in L_{act}^x)$ and $(g_\tau^y \neq g_\rho^z)$ do
   a. If adding edge $e^x_{(g_\tau^y, g_\rho^z)}$ to $\mathcal{G}_{act}^x$ does not create a cycle
      - Add edge $e^x_{(g_\tau^y, g_\rho^z)}$ to $\mathcal{G}_{act}^x$;
      - Add column r^x in p^1 with $p^1[g_\tau^y, \mathsf{r}^x] = 1$ and $p^1[g_\rho^z, \mathsf{r}^x] = 1$; r++;
   b. Delete (σ, g_ρ^z) from L_{act}^x;
3. Compute a spanning forest $\mathcal{F}_{act}^x$ of $\mathcal{G}_{act}^x$ and keep it associated with grade x

Algorithm BOUNDARY (x, σ)
1. Find $(\tau_1, g_{\kappa_1}^{y_1}), \ldots, (\tau_k, g_{\kappa_k}^{y_k})$ in L_{act}^x s.t. $\sum_{i=1}^k \tau_i = \partial\sigma$;
2. Add column g_σ^x in f^0 with $f^0[g_{\kappa_j}^{y_j}, g_\sigma^x] = 1 \; \forall j \in \{1, \ldots, k\}$

In the routine LIFT, we compute the matrices f^1 and ϑ, by lifting the maps $f^0 \circ p^1$ and $f^0 \circ f^0$ along p^1. Let $A[:, \mathrm{ind}]$ denote the column of a matrix A with the column index ind. Then the lifting is carried out by solving the equations $p^1 \circ f^1[:, \mathsf{r}^x] = f^0 \circ p^1[:, \mathsf{r}^x]$ and $p^1 \circ \vartheta[:, g_\sigma^x] = f^0 \circ f^0[:, g_\sigma^x]$ at the current grade x, for all unknown columns $f^1[:, \mathsf{r}^x]$ and $\vartheta[:, g_\sigma^x]$, indexed by r^x and g_σ^x, respectively.

The columns of the coefficient matrix p^1 and the rows of the right-hand side $f^0 \circ p^1$ and $f^0 \circ f^0$ correspond to edges and vertices in the total relation graph $\mathcal{G}$ associated to p^1, respectively (see Theorem 24). By Proposition 10, to solve the equations, we only have to consider those columns and rows that are indexed by a grade less than or equal to x, which correspond to the edges and vertices of $\mathcal{G}(x)$. Moreover, it is enough to consider only the columns corresponding to the edges in a spanning forest of $\mathcal{G}(x)$, since these columns already yield the full column rank.

The algorithm maintains such a spanning forest $\mathcal{F}_{act}^x$ (see Lemma 41). For each equation $p^1 \circ f^1[:, \mathsf{r}^x] = f^0 \circ p^1[:, \mathsf{r}^x]$, it first computes the right-hand side $b[\mathcal{F}_{act}^x \setminus \Omega] := f^0 \circ p^1[:, \mathsf{r}^x]$ and $b[\Omega] := \overline{1} \circ f^0 \circ p^1[:, \mathsf{r}^x]$, where $b[\mathcal{F}_{act}^x \setminus \Omega]$ is a vector indexed as the rows of $f^0 \circ p^1$, $\overline{1} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}$, and $b[\Omega]$ is an additional row that accounts for the additional vertex Ω in the relation graph. For a connected component C of $\mathcal{F}_{act}^x$, let $V(C)$ and $E(C)$ denote its vertices and edges. Then for each such component, the algorithm calls the routine TREESOLVE (see details in Appendix C) to solve the equation $\overline{p}^1[V(C), E(C)] \circ f^1[E(C), \mathsf{r}^x] = b[V(C)]$ in linear time. Here $\overline{p}^1[V(C), E(C)]$ is the boundary matrix of C, which is p^1 with an additional row accounting for the vertex Ω (see Theorem 24) restricted to the vertices and edges in the component C. The computation for ϑ is analogous.

Algorithm LIFT (x)

1. For each r^x added at grade x do
   a. Compute $b[\mathcal{F}^x_{act} \setminus \Omega] \coloneqq f^0 \circ p^1[:, \mathsf{r}^x]$ and $b[\Omega] \coloneqq \overline{1} \circ f^0 \circ p^1[:, \mathsf{r}^x]$;
   b. For each connected component C of $\mathcal{F}^x_{act}$ do
      - Set $f^1[E(C), \mathsf{r}^x] \coloneqq$TREESOLVE$(C, b[V(C)])$;
2. For each g^x_σ added at grade x do
   a. Compute $b[\mathcal{F}^x_{act} \setminus \Omega] \coloneqq f^0 \circ f^0[:, g^x_\sigma]$ and $b[\Omega] \coloneqq \overline{1} \circ f^0 \circ f^0[:, g^x_\sigma]$;
   b. For each connected component C of $\mathcal{F}^x_{act}$ do
      - Set $\vartheta[E(C), g^x_\sigma] \coloneqq$TREESOLVE$(C, b[V(C)])$

The routine RELREL creates relations of relations adding columns to the matrix p^2. It uses the spanning forest $\mathcal{F}^x_{act}$ of the relation graph $\mathcal{G}^x_{act}$ available at this stage. For every edge in $\mathcal{G}^x_{act}$ that is not in $\mathcal{F}^x_{act}$, it computes a cycle, say γ, and registers a relation of relations using the edges on γ. It is registered as a column in p^2. For example, in Figure 11, the two edges $e_{(g^{y_1}_\sigma, g^{y_2}_\sigma)}, e_{(g^{y_2}_\sigma, \Omega)}$ add two columns $\mathsf{rr}^{x_4}_1$ and $\mathsf{rr}^{x_4}_2$ to p^2. Finally, there is an option to call REDUCE which minimizes p^2.

Algorithm RELREL (x)

- For every edge $e^{z_0}_* \in \mathcal{G}^x_{act} \setminus \mathcal{F}^x_{act}$ do
  1. Find the path of edges $e^{z_1}_*, \ldots, e^{z_s}_*$ in $\mathcal{F}^x_{act}$ connecting the vertices of $e^{z_0}_*$;
  2. Add a relation of relations rr^x in p^2 where $p^2[\mathsf{r}^{z_j}, \mathsf{rr}^x] = 1 \; \forall j \in \{0, \ldots, s\}$; rr++;
- (Optional) Call REDUCE (x)

The routine REDUCE removes redundant relations of relations by left-to-right column reduction of an appropriate submatrix of p^2. This makes p^2 minimal as we prove in Proposition 50.

Algorithm REDUCE (x)

1. Compute the submatrix $A^{\leq x}$ of p^2 consisting of columns with indices rr^y such that $y \leq x$ and the column order respects the chosen linear extension of grades;
2. Reduce $A^{\leq x}$ (with left-to-right column additions as done in standard persistent homology) such that each column either has a unique row as a pivot or is completely zeroed out;
3. Remove a column with index rr^x from p^2 if the column is reduced to a zero column in the reduced matrix $A^{\leq x}$

4.3 Time complexity analysis

We analyze the time complexity of the algorithm PIREP. Let t_0 and t_1 be the total number of vertices (grades) and edges, respectively, in the directed acyclic graph (Hasse diagram) representing the poset P and let $t = t_0 + t_1$. Let n denote the number $|\mathcal{S}|$ of generators plus the number $|\mathcal{C}|$ of edge events input to the algorithm. Thus, the maximum number of simplices in the complex $K(x)$ is $O(n)$ for any x. The maximum dimension d of a simplex is assumed to be constant for this analysis. For this analysis, we denote by w_x the number of y in P such that $y \prec x$ and note that $\sum_{x \in P} w_x = t_1$.

We first analyze the maximal size of the data structures L^x_{act}, $\mathcal{F}^x_{act}$, and G^x_{act} at a given grade x. At a minimal element in P, the size of L^x_{act} is clearly $O(n)$ because we add at most $O(n)$ simplex generators to it in GENERATOR. For each $y \prec x$, L^y_{act} contains at most one pair (σ, g^y_τ) for each $\sigma \in K(y)$ at the end of GENERATOR because we remove all but one such pair. Hence, at the end of GENERATOR L^y_{act} is of size $O(n)$. This implies that COLLAPSE adds at

most $O(w_x n)$ elements and Generator adds at most $O(n)$ elements to L^x_{act}. Therefore, the maximal size of L^x_{act} at any point in the algorithm is $O\big((w_x+1)n\big)$.

The set of vertices of $\mathcal{F}^x_{act}$ always corresponds to a subset of the set of simplex generators $\mathcal{S}$, so it has at most $O(n)$ vertices. Thus, the forest $\mathcal{F}^x_{act}$ is of size $O(n)$ at any point of the algorithm. The graph $\mathcal{G}^x_{act}$ is first initialized in Collapse as the union of all $\mathcal{F}^y_{act}$ with $y \prec x$. Thus, it has size $O(w_x n)$ after initialization. Also, as there are at most $O(n)$ vertices overall, $\mathcal{G}^x_{act}$ contains at most $O(n)$ vertices throughout. In particular, $\mathcal{G}^x_{act}$ has at most $O(n)$ components. Because in Collapse and Generator we only add edges if they do not create a cycle, each added edge must connect two components. This implies that we can add at most $O(n)$ edges at each grade x and thus the graph $\mathcal{G}^x_{act}$ has size $O\big((w_x+1)n\big)$ at any point of the algorithm.

Analysis of Collapse**.** First assume that $w_x = 0$, i.e., the grade x has no predecessors. In this case, we only initialize $\mathcal{G}^x_{act}$ with the vertex Ω, which can be done in constant time. Over all grades $x \in P$ such that $w_x = 0$ this takes at most $O(t_0)$ time. Notice that the poset $\overline{P} = P \cup \{-\infty\}$ has a single minimum at $\{-\infty\}$ where the vertex Ω is born, but our algorithm directly operates on P. Thus, we have to initialize a vertex Ω at each minimal element of P, which are the only grades where $w_x = 0$.

Now assume that $w_x > 0$. Step 1 clearly takes constant time. In step 2, we construct $\mathcal{G}^x_{act}$ as a union of w_x forests $\mathcal{F}^y_{act}$ of size $O(n)$. Since we have a fixed set of vertices, corresponding to the simplex generators, this can be done in $O(w_x n)$ time.

For a given $y \prec x$, let $n_y \leq n$ denote the number of simplices in $K(y)$ and $n_{y\prec x}$ denote the number of edge events on the poset edge $y \prec x$. Step 3(a) can be done using a dictionary (implemented with a simple array of vertex ids) for $y \prec x$ initialized as $v \mapsto v$ for all vertices in L^y_{act}, which has length at most n_y. Then we go over at most $n_{y\prec x}$ edge events and set $v \mapsto w$ for $c^{y\prec x}_{v\mapsto w}$. In step 3(b), we update at most n_y simplices using this dictionary, which can be done in constant time because we assume the maximum dimension d of a simplex to be a constant. The same is true for checking whether there is a repeated vertex in step 3(b.i).

For step 3(b.ii), we can use a union-find data structure [10] to check efficiently whether an added edge creates a cycle in the graph $\mathcal{G}^x_{act}$. Such a union-find data structure for the components of $\mathcal{G}^x_{act}$ can be initialized in $O\big(nw_x\alpha(n)\big)$ time (once per grade x), where $\alpha(\cdot)$ is the extremely slowly growing inverse Ackermann function. Over all grades this adds a total cost of $O\big(t_1 n\alpha(n)\big)$. Note that if there are no predecessors we do not need the union-find structure. With this union-find data structure, we can check if an edge creates a cycle in $O\big(\alpha(n)\big)$ time. An edge creates a cycle if and only if the two endpoints are already in the same component. If they are in different components, the edge does not create a cycle and the two components are merged. Therefore, step (3) takes at most $O\big(n_y + n_{y\prec x} + n_y\alpha(n)\big) = O\big(n\alpha(n)\big)$ time for each $y \prec x$. Over all grades $x \in P$ such that $w_x > 0$ and all w_x predecessors $y \prec x$ this results in a worst case complexity of $\sum_x O\big(w_x n\alpha(n)\big) = O\big(nt_1\alpha(n)\big)$. Hence, overall Collapse has a worst-case complexity of $O\big(nt_1\alpha(n) + t_0\big)$.

Analysis of Generator**.** Step 1 in the routine Generator accesses the generators in $\mathcal{S}$ and their boundaries. Finding the simplex generators born at grade x and adding them to L^x_{act} takes at most $O(n)$ time and we add at most $O(n)$ generators. When adding these new generators, we also update the union-find data structure for the components of $\mathcal{G}^x_{act}$, initialized in Collapse, which can be done in constant time, since each generator added at this point forms a new component. For each added generator, finding active generators of its boundary can be done in $O(d)$ time, assuming that simplices and their boundaries are maintained with an appropriate pointer data structure. Assuming d to be a constant, we get a total complexity of this step as $O(n)$ per grade and $O(nt_0)$ over all grades.

In step 2, we identify all pairs (σ, g^y_τ) and (σ, g^z_ρ) with the same simplex `id`. This can be implemented efficiently as follows: We maintain L^x_{act} with an array of lists containing all pairs (σ, g^y_τ) indexed by the unique `id` of the simplex σ. Then for each simplex σ that appears in that array, we keep the first list entry (σ, g^y_τ) indexed by σ and remove all following (σ, g^z_ρ). For each removed (σ, g^z_ρ), we have to check if adding the edge $e^x_{(g^y_\tau, g^z_\rho)}$ creates a cycle in $\mathcal{G}^x_{act}$. As discussed in the analysis of Collapse, this can be done in $O\big(\alpha(n)\big)$ time using the union-find data structure on the components of $\mathcal{G}^x_{act}$. We have already analyzed that L^x_{act} is of size at most $O\big((w_x+1)n\big)$. Therefore, we have to check for at most $O\big((w_x+1)n\big)$ many (σ, g^z_ρ). Hence, step (2) takes $O\big((w_x+1)n\alpha(n)\big)$ time per grade and $O\big(nt\alpha(n)\big)$ time after summing over all grades.

In step (3), we compute a spanning forest $\mathcal{F}^x_{act}$ of $\mathcal{G}^x_{act}$. As we have analyzed above, the graph $\mathcal{G}^x_{act}$ is of size $O\big((w_x+1)n\big)$. Computing a spanning forest can be done in $O(|\text{Vertices}| + |\text{Edges}|)$ using breadth-first or depth-first search. Thus, step (3) costs $O\big((w_x+1)n\big)$ per grade and $O(nt)$ over all grades. Therefore, overall the routine Generator has a worst-case time complexity of $O\big(nt\alpha(n)\big)$.

Note that the matrix p^1 gets at most $O(n)$ columns at every grade x because they correspond to the edges of the forest $\mathcal{F}^x_{act}$ (we only add edges that connect two components). Therefore, overall we add at most $O(nt_0)$ columns over all grades and p^1 has at most $O(nt_0)$ columns. Each column of p^1 has at most two entries in sparse matrix representation. Thus, the overall time (and memory) cost of creating a representation of p^1 is $O(nt_0)$.

Analysis of Lift. In step $1(a)$, we first compute $f^0 \circ p^1[:, \mathsf{r}^x]$. Since each column of p^1 has at most two entries and each column of f^0 has at most $d+1$ entries, computing this product is equivalent to an XOR of two columns of f^0 in sparse matrix format and takes constant time (d is assumed to be constant). Moreover, $f^0 \circ p^1[:, \mathsf{r}^x]$ has at most $2(d+1)$ entries, which implies that computing $\overline{1} \circ f^0 \circ p^1[:, \mathsf{r}^x]$, i.e., summing up all entries, can also be done in constant time. Let n_C denote the size of the connected component C of $\mathcal{F}^x_{act}$. Since $\mathcal{F}^x_{act}$ is of size $O(n)$, we have $O(\sum_C n_C) = O(n)$. By Theorem 57 in Appendix C, the call of TreeSolve on C costs $O(n_C)$ time. Thus, overall step $1(b)$ takes time $O(n)$. We execute step 1 exactly once for each r^x added at grade x. Since there are at most $O(nt_0)$ relations, the total cost of step 1 over all grades $x \in P$ is $O(n^2 t_0)$. For step $2(a)$, we again note that f^0 has at most $d+1$ entries. Thus, step $2(a)$ can also be done in constant time. The complexity of step $2(b)$ is $O(n)$ by the same argument as for step $1(b)$. We execute step 2 exactly once for every generator g^x_σ added by the algorithm at grade x. Since there are at most $O(n)$ generators, the total cost of step 2 over all grades $x \in P$ is $O(n^2)$. Thus, in total Lift has a worst-case time complexity of $O(n^2 t_0)$.

Analysis of RelRel. Steps 1 and 2 take at most $O(n)$ time per edge $e^{z_0}_*$ since $\mathcal{F}^x_{act}$ has size $O(n)$. We have that $\mathcal{G}^x_{act} \setminus \mathcal{F}^x_{act}$ is of size $O(w_x n)$ because after initialization $\mathcal{G}^x_{act}$ is of size $O(nw_x)$ and every edge added afterwards to $\mathcal{G}^x_{act}$ is in $\mathcal{F}^x_{act}$. Counting over all edges of $\mathcal{G}^x_{act} \setminus \mathcal{F}^x_{act}$, we get a complexity of $O(n^2 w_x)$. Then, over all grades, we get a bound of $O(\sum_x n^2 w_x) = O(n^2 t_1)$. Notice that, at every grade x, we add at most $O(w_x n)$ columns to p^2. Therefore, p^2 has at most $O(nt_1)$ columns. Because each column of p^2 corresponds to the relations/edges on a path in the forest $\mathcal{F}^x_{act}$ plus one additional edge closing the cycle, it has at most $O(n)$ entries. Thus, the overall time (and memory) cost of creating a representation of p^2 is $O(n^2 t_1)$.

Analysis of Reduce. The most costly step in Reduce is the matrix reduction. This can be done in $O(s^\omega)$ time if the matrix has $O(s)$ columns and rows, where $\omega < 2.373$ is the exponent for the matrix multiplication time complexity [1]. By the analysis of RelRel, the matrix $A^{\leq x}$ being a submatrix of p^2, cannot have more than $O(nt_1)$ columns. Moreover,

since the entries in the columns of p^2 correspond to relations (columns of p^1), in non-sparse representation p^2 has at most $O(nt_0)$ rows (see analysis of Generator). Then, Reduce takes $O\big(n^\omega t^\omega\big)$ time per grade and $O\big(n^\omega t^{\omega+1}\big)$ time for all grades.

The proof of correctness in Section 5 together with Theorem 22 establishes that the algorithm PiRep computes a projective implicit representation of $H_\ell(K)$. It also establishes that p^1_ℓ is a minimal presentation (Theorem 45), that p^2_ℓ without the call of Reduce is asymptotically minimal (Proposition 48), and that p^2_ℓ with the call of Reduce is minimal (Theorem 51). We summarize this in our main theorem:

▶ **Theorem 25.** *Let $K\colon P \to \mathbf{SCpx}$ be a poset tower represented by the Hasse diagram of P with t_0 nodes and t_1 edges, together with the lists $\mathcal{S}$ and $\mathcal{C}$ of simplex generators and edge events. Set $t = t_0 + t_1$ and $n = |\mathcal{S}| + |\mathcal{C}|$. Then the following statements hold for all ℓ, where (1) uses only* Collapse *and* Generator*, (2) additionally uses* RelRel*, (3) additionally uses* Reduce*, and (4) refers to the full algorithm without* Reduce*.*

1. PiRep *computes a minimal presentation matrix p^1_ℓ for $C_\ell(K)$ with $O(nt_0)$ columns in $O\big(nt\alpha(n)\big)$ time.*
2. PiRep *computes matrices p^1_ℓ and p^2_ℓ, representing a projective resolution of $C_\ell(K)$ up to the second term, in $O(n^2t)$ time. The computed matrix p^2_ℓ has $O(nt_1)$ columns, which is asymptotically minimal because a minimal p^2_ℓ can have $\Theta(nt_1)$ columns.*
3. PiRep *computes a minimal p^2_ℓ in $O\big(n^\omega t^{\omega+1}\big)$ time.*
4. PiRep *computes a projective implicit representation of $H_\ell(K)$ in $O(n^2t)$ time.*

5 Correctness of the algorithm PiRep

In this section, we prove the correctness of the PiRep algorithm, hereafter referred to as "the algorithm" for brevity. The overall proof strategy is to show that the algorithm correctly computes matrix representations of the maps p^1_ℓ, p^2_ℓ, f^0_ℓ, f^1_ℓ, and $\vartheta_{\ell+1}$, shown in the diagram (9). This means that p^1_ℓ and p^2_ℓ form a partial resolution of $C_\ell(K)$ up to the second term, f^0_ℓ and f^1_ℓ form a partial lift of ∂_ℓ up to the first term, and $\vartheta_{\ell+1}$ satisfies $p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1} = f^0_\ell \circ f^0_{\ell+1}$. Assembling these maps into the block matrices in (16) then yields a PiRep of $H_\ell(K)$ by Theorem 22. In Section 5.1, we show that p^1_ℓ, as constructed by PiRep, is a valid presentation and that f^0_ℓ is a valid lift of ∂_ℓ to the generators. In Section 5.2, we show that the constructed p^1_ℓ is a minimal presentation. In Section 5.3, we show that p^1_ℓ and p^2_ℓ together form a partial resolution up to the second term. Moreover, we show that p^2_ℓ is asymptotically minimal and even minimal if Reduce is called in PiRep. In Section 5.4, we show that the maps f^1_ℓ and $\vartheta_{\ell+1}$ satisfy the required commutativity relations.

5.1 Correctness of computing p^1_ℓ and f^0_ℓ

In this section, we prove that p^1_ℓ, as constructed in the algorithm PiRep, for a poset tower $K\colon P \to \mathbf{SCpx}$, is a presentation of $C_\ell(K)$. While the algorithm processes all degrees ℓ simultaneously, for the proof we will w.l.o.g. fix a degree ℓ. Also, we abbreviate $C_\ell(K)$ by C_ℓ. We note that the algorithm implicitly constructs a set of generators g^x_σ and relations r^y as the indices of the rows and columns of the matrices p^1_ℓ, respectively. Hence, using the notation established in Section 2.2, the domain and codomain of the maps of projective modules $p^1_\ell\colon R_\ell \to G_\ell$, represented by the matrices, are given by $R_\ell = \langle r^y | r^y \in \text{ column indices of } p^1_\ell\rangle$ and $G_\ell = \langle g^x_\sigma | g^x_\sigma \in \text{ row indices of } p^1_\ell\rangle$. We start by showing that G_ℓ generates C_ℓ.

▶ **Proposition 26.** *For $K\colon P \to \mathbf{SCpx}$, let $G_\ell = \langle g_\sigma^x | g_\sigma^x \in$ row indices of $p_\ell^1 \rangle$ be the codomain of p_ℓ^1 constructed by the algorithm. Then*

$$G_\ell = \bigoplus_{g_\sigma^x \in \mathcal{S}_\ell} \mathrm{Proj}[x].$$

Proof. The algorithm adds every generator $g_\sigma^x \in \mathcal{S}_\ell$ at the grade $x \in P$ in the routine GENERATOR. ◀

At every grade $x \in P$, the vector space $G_\ell(x)$ has a basis $\{g_\sigma^y \in \mathcal{S}_\ell | y \leq x\}$ where g_σ^y is the contribution of the summand $\mathrm{Proj}[y]$. The vector space $C_\ell(x)$ has a basis generated by the ℓ-simplices $K_\ell(x)$. We denote the basis elements of $C_\ell(x)$ simply by σ.

▶ **Definition 27.** *Define the morphism $\alpha_\ell\colon G_\ell \to C_\ell$ in the following way:*

$$\alpha_\ell(x)(g_\sigma^y) = \begin{cases} K(y \leq x)(\sigma) & \text{if } K(y \leq x)(\sigma) \in K_\ell(x) \\ 0 & \text{otherwise} \end{cases}.$$

In the following, we will use the convention $K(y \leq x)(\sigma) = 0$, for $\sigma \in K_\ell(y)$, if $K(y \leq x)(\sigma) \notin K_\ell(x)$. Also note that $g_\sigma^y \in \mathcal{S}_\ell$ implies $\sigma \in K_\ell(y)$.

▶ **Proposition 28.** *The morphism α_ℓ is well-defined.*

Proof. Let $x \leq y \in P$ and $g_\sigma^z \in \mathcal{S}_\ell$ such that $z \leq x$. Then

$$\begin{aligned} C_\ell(x \leq y) \circ \alpha_\ell(x)(g_\sigma^z) &= C_\ell(x \leq y)\big(K(z \leq x)(\sigma)\big) \\ &= K(x \leq y) \circ K(z \leq x)(\sigma) \\ &= K(z \leq y)(\sigma). \end{aligned}$$

On the other hand,

$$\alpha_\ell(y) \circ G_\ell(x \leq y)(g_\sigma^z) = \alpha_\ell(y)(g_\sigma^z) = K(z \leq y)(\sigma).$$

Hence, the diagram

$$\begin{array}{ccc} C_\ell(y) & \xleftarrow{\alpha_\ell(y)} & G_\ell(y) \\ {\scriptstyle C_\ell(x\leq y)}\uparrow & & \uparrow{\scriptstyle G_\ell(x\leq y)} \\ C_\ell(x) & \xleftarrow{\alpha_\ell(x)} & G_\ell(x) \end{array}$$

commutes and α_ℓ is well-defined. ◀

▶ **Proposition 29.** *The morphism α_ℓ is an epimorphism.*

Proof. Let $\sigma \in K_\ell(x)$. If $\sigma \notin \bigcup_{y \prec x} \mathrm{im}\big(K(y \prec x)\big)$, then $g_\sigma^x \in \mathcal{S}_\ell$ and $\alpha_\ell(x)(g_\sigma^x) = \sigma$. Otherwise, $\sigma = K(y \prec x)(\tau)$ for some $\tau \in K_\ell(y)$. If τ is generated at y, there is g_τ^y in $\mathcal{S}_\ell$, otherwise we can iterate this argument until we find $\rho \in K_\ell(z)$ such that $g_\rho^z \in \mathcal{S}_\ell$ and $K(z \leq x)(\rho) = \sigma$. Note that this works because we reach a minimal element $a \in P$ after a finite number of steps and, by definition, every $\pi \in K_\ell(a)$ is a generator $g_\pi^a \in \mathcal{S}_\ell$. This implies

$$\begin{aligned} \alpha_\ell(x)(g_\rho^z) &= \alpha_\ell(x) \circ G_\ell(z \leq x)(g_\rho^z) \\ &= C_\ell(z \leq x) \circ \alpha_\ell(z)(g_\rho^z) \\ &= C_\ell(z \leq x)(\rho) \\ &= K(z \leq x)(\rho) = \sigma. \end{aligned}$$

Since $C_\ell(x)$ has a basis $K_\ell(x)$, $\alpha_\ell(x)$ is an epimorphism for all $x \in P$ and, thus, α_ℓ is an epimorphism. ◀

Proposition 29 shows that G_ℓ does indeed generate C_ℓ. The following propositions establish that p^1_ℓ is a presentation of C_ℓ, a result we summarize in Theorem 37. To prove them, we make use of the following lemmas.

▶ **Lemma 30.** *At every grade x_i in the algorithm* PiRep*, after the call of* Generator*, $L^{x_i}_{act} \cong K(x_i)$ as sets. In particular, for each $\sigma \in K(x_i)$, there exists an active generator $(\sigma, g^y_\tau) \in L^{x_i}_{act}$.*

Proof. We proceed by induction on the linear extension of the grades in P. The grade x_0 is minimal in P as any predecessor would have to come before x_0 in the linear extension. At a minimal element $x_0 \in P$, for every $\sigma \in K(x_0)$ there is a $g^{x_0}_\sigma \in \mathcal{S}$. Therefore, the identification $(\sigma, g^{x_0}_\sigma) \sim \sigma$ yields $L^{x_0}_{act} \cong K(x_0)$.

Assume that $L^{x_j}_{act} \cong K(x_j)$ for all $0 \leq j < i$. For every k-simplex $\sigma \in \bigcup_{y \prec x_i} \operatorname{im}\big(K(y \prec x_i)\big)$, there exists a $y \prec x_i$ and an l-simplex $\tau \in K(y)$, with $l \geq k$, such that $K(y \prec x_i)(\tau) = \sigma$. By dropping superfluous vertices from τ, we can always find a k-simplex $\tau' < \tau \in K(y)$ such that $K(y \prec x_i)(\tau') = \sigma$. Thus, w.l.o.g. we can assume $l = k$ and no vertex of τ gets collapsed by $K(y \prec x_i)$. By assumption, there exists (τ, g^z_ρ) in L^y_{act}. By calling Collapse, the algorithm adds $\big(K(y \prec x_i)(\tau), g^z_\rho\big) = (\sigma, g^z_\rho)$ to $L^{x_i}_{act}$. For every $\sigma \in K(x_i) \setminus \bigcup_{y \prec x_i} \operatorname{im}\big(K(y \prec x_i)\big)$ there exists $g^{x_i}_\sigma \in \mathcal{S}$ which is added as $(\sigma, g^{x_i}_\sigma)$ to $L^{x_i}_{act}$ in step 1 of Generator. Thus, every $\sigma \in K(x_i)$ has at least one representative in $L^{x_i}_{act}$. In step 2 of Generator, (σ, g^y_τ) is deleted if there is another element (σ, g^z_ρ) in $L^{x_i}_{act}$. Therefore, after this step, there is exactly one representative (σ, g^z_ρ) for every $\sigma \in K(x_i)$ and $L^{x_i}_{act} \cong K(x_i)$. ◀

▶ **Lemma 31.** *Suppose that after the call of the routines* Collapse *and* Generator *in* PiRep*, $(\tau, g^y_\sigma) \in L^x_{act}$ is active at $x \in P$. Then $K(y \leq x)(\sigma) = \tau$.*

Proof. Since $g^y_\sigma \in \mathcal{S}$, we have that (σ, g^y_σ) is added to L^y_{act} by Generator at $y \in P$. Since active simplices are added from $\mathcal{S}$ by Generator or inherited from predecessors by Collapse, (τ, g^y_σ) can only be active at x if $y \leq x$ and if there is a path $y = z_0 \prec \cdots \prec z_k = x$ in P along which g^y_σ is active as (ρ_i, g^y_σ) at z_i with $\rho_0 = \sigma$ and $\rho_k = \tau$. By the update procedure of the Collapse routine in PiRep, $\rho_i = K(z_{i\text{-}1} \prec z_i)(\rho_{i\text{-}1})$. Therefore, $K(y \leq x)(\sigma) = K(z_{k\text{-}1} \prec z_k) \circ \cdots \circ K(z_0 \prec z_1)(\rho_0) = \rho_k = \tau$. ◀

Let $r^{y_1}, \dots, r^{y_n}$ be the relations added by the algorithm PiRep for the ℓ-simplices. Then we have $R_\ell = \bigoplus_{i=1}^n \operatorname{Proj}[y_i]$. As for the generators G_ℓ, at each grade $x \in P$ the vector space $R_\ell(x)$ has a basis $\{r^y | y \leq x\}$.

▶ **Proposition 32.** *Let $G_\ell \xleftarrow{p^1_\ell} R_\ell$ be the matrix for the ℓ-simplices constructed by the* PiRep *algorithm. Then $\alpha_\ell \circ p^1_\ell = 0$.*

Proof. First, let $r^y \in R_\ell$ be a relation, added by the Collapse routine, such that $p^1_\ell(r^y) = g^z_\sigma$. The algorithm adds the relation at $y \in P$ because there exists $a \prec y \in P$ such that (τ, g^z_σ) is active at a but $K(a \prec y)(\tau) = 0$. For $y \leq x \in P$ we obtain:

$$\alpha_\ell(x) \circ p^1_\ell(x)(r^y) = \alpha_\ell(x)(g^z_\sigma) = K(z \leq x)(\sigma).$$

Since (τ, g^z_σ) is active at a, Lemma 31 implies that $z \leq a$ and

$$\begin{aligned} K(z \leq x)(\sigma) &= K(y \leq x) \circ K(a \leq y) \circ K(z \leq a)(\sigma) \\ &= K(y \leq x) \circ K(a \leq y)(\tau) = 0. \end{aligned}$$

Thus, $\alpha_\ell(x) \circ p^1_\ell(x)(r^y) = 0$.

Now, let $r^y \in R_\ell$ be a relation, added by the algorithm, such that $p^1_\ell(r^y) = g^z_\sigma + g^w_\tau$. Since a simplex generator g^y_ξ is added only when ξ is not in the image of any predecessor of y, no relation r^y created by GENERATOR at grade y can involve a generator added at grade y. Indeed, any other active representative of the same simplex ξ would have to be inherited from a predecessor of y, contradicting the defining property of g^y_ξ. Hence, the algorithm adds the relation r^y in the GENERATOR routine at $y \in P$ because there exist $a \prec y$ and $b \prec y$ such that (ρ, g^z_σ) is active at a, (κ, g^w_τ) is active at b and $\pi \coloneqq K(a \prec y)(\rho) = K(b \prec y)(\kappa)$. For $y \leq x$, we get

$$\begin{aligned}\alpha_\ell(x) \circ p^1_\ell(x)(r^y) &= \alpha_\ell(x)(g^z_\sigma + g^w_\tau)\\ &= \alpha_\ell(x)(g^z_\sigma) + \alpha_\ell(x)(g^w_\tau)\\ &= K(z \leq x)(\sigma) + K(w \leq x)(\tau).\end{aligned}$$

By Lemma 31, we further obtain

$$\begin{aligned}K(z \leq x)(\sigma) &= K(y \leq x) \circ K(a \leq y) \circ K(z \leq a)(\sigma)\\ &= K(y \leq x) \circ K(a \leq y)(\rho)\\ &= K(y \leq x)(\pi)\end{aligned}$$

and analogously

$$\begin{aligned}K(w \leq x)(\tau) &= K(y \leq x) \circ K(b \leq y) \circ K(w \leq b)(\tau)\\ &= K(y \leq x) \circ K(b \leq y)(\kappa)\\ &= K(y \leq x)(\pi).\end{aligned}$$

Therefore, $\alpha_\ell(x) \circ p^1_\ell(x)(r^y) = 0$.

We have shown that for all $x \in P$ and all relations r^y of R_ℓ such that $y \leq x$ we have $\alpha_\ell(x) \circ p^1_\ell(x) = 0$. This implies $\alpha_\ell \circ p^1_\ell = 0$ and $\mathrm{im}(p^1_\ell) \subseteq \ker(\alpha_\ell)$. ◀

▶ **Lemma 33.** *Let $g^x_\sigma \in \mathcal{S}_\ell$ be such that $\tau \coloneqq K(x \leq y)(\sigma) \in K_\ell(y)$. Then after the call of* COLLAPSE *and* GENERATOR *in the algorithm, there exists $(\tau, g^z_\rho) \in L^y_{act}$ and $r \in R_\ell$ such that $g^x_\sigma + g^z_\rho = p^1_\ell(r)$.*

Proof. Let $x = z_0 \prec \cdots \prec z_m = y$ be a path from x to y in P. Moreover, let $\pi_i = K(z_0 \leq z_i)(\sigma)$ for $0 \leq i \leq m$. In particular, $\pi_0 = \sigma$, $\pi_m = K(x \leq y)(\sigma) = \tau$ and, since $\tau \in K_\ell(y)$, we have $\pi_i \in K_\ell(z_i)$. By Lemma 30, each π_i has an active generator $(\pi_i, g^{a_i}_{\kappa_i})$ at z_i. At $x = z_0$, the generator $(\sigma, g^x_\sigma) = (\pi_0, g^{a_0}_{\kappa_0})$ is active.

In the routine COLLAPSE at grade z_i, we add $\big(K(z_{i-1} \prec z_i)(\pi_{i-1}), g^{a_{i-1}}_{\kappa_{i-1}}\big) = \big(\pi_i, g^{a_{i-1}}_{\kappa_{i-1}}\big)$ to $L^{z_i}_{act}$. If $(\pi_i, g^{a_{i-1}}_{\kappa_{i-1}}) \neq (\pi_i, g^{a_i}_{\kappa_i})$, in the routine GENERATOR, the algorithm adds the relation $r^{z_i} \mapsto g^{a_{i-1}}_{\kappa_{i-1}} + g^{a_i}_{\kappa_i}$ if the edge $e^{z_i}_{(g^{a_{i-1}}_{\kappa_{i-1}}, g^{a_i}_{\kappa_i})}$ does not close a cycle. If the edge $e^{z_i}_{(g^{a_{i-1}}_{\kappa_{i-1}}, g^{a_i}_{\kappa_i})}$ closes a cycle, then there is a path from $g^{a_{i-1}}_{\kappa_{i-1}}$ to $g^{a_i}_{\kappa_i}$ in $\mathcal{G}^{z_i}_{act}$. The edges on this path correspond to relations $r^{w_1}, \ldots, r^{w_q} \in R_\ell$ with $w_j \leq z_i$ such that $p^1_\ell(r^{w_1} + \cdots + r^{w_q}) = g^{a_{i-1}}_{\kappa_{i-1}} + g^{a_i}_{\kappa_i}$. Therefore, in any case, there exists $r_i \in R_\ell$ such that $p^1_\ell(r_i) = g^{a_{i-1}}_{\kappa_{i-1}} + g^{a_i}_{\kappa_i}$. If $(\pi_i, g^{a_{i-1}}_{\kappa_{i-1}}) = (\pi_i, g^{a_i}_{\kappa_i})$ we set $r_i = 0$.

We now define $(\tau, g^z_\rho) \coloneqq (\pi_m, g^{a_m}_{\kappa_m})$ and $r = r_1 + \cdots + r_m$ and obtain $p^1_\ell(r) = p^1_\ell(r_1 + \cdots + r_m) = g^{a_0}_{\kappa_0} + g^{a_m}_{\kappa_m} = g^x_\sigma + g^z_\rho$. ◀

▶ **Lemma 34.** *Let $g^x_\sigma \in \mathcal{S}_\ell$ and $x \leq y \in P$ such that $K(x \leq y)(\sigma) \notin K_\ell(y)$. Then after the call of* COLLAPSE *in the algorithm, there exists $r \in R_\ell$ such that $p^1_\ell(r) = g^x_\sigma$.*

Proof. Let $x = z_0 \prec \cdots \prec z_m = y$ be a path from x to y in P. Moreover, let $\pi_i = K(z_0 \leq z_i)(\sigma)$ for $0 \leq i \leq m$. Since $K(x \leq y)(\sigma) = K(z_{m-1} \prec z_m) \circ \cdots \circ K(z_0 \prec z_1)(\sigma) \notin K_\ell(y)$, there exists a minimal $0 < i \leq m$ such that $\pi_i \notin K_\ell(z_i)$. By Lemma 33, there exists an active generator (π_{i-1}, g_ρ^z) at z_{i-1} representing π_{i-1} and $r' \in R_\ell$ such that $g_\sigma^x + g_\rho^z = p_\ell^1(r')$. Since $K(z_{i-1} \prec z_i)(\pi_{i-1}) \notin K_\ell(z_i)$ the algorithm adds the relation $r^{z_i} \mapsto g_\rho^z$ in the Collapse routine if the edge $e^{z_i}_{(\Omega, g_\rho^z)}$ does not close a cycle in $\mathcal{G}_{act}^{z_i}$. If the edge $e^{z_i}_{(\Omega, g_\rho^z)}$ does close a cycle, then there is a path from Ω to g_ρ^z in $\mathcal{G}_{act}^{z_i}$. This path corresponds to relations $r^{w_1}, \dots, r^{w_q}$ such that $p_\ell^1(r^{w_1} + \cdots + r^{w_q}) = g_\rho^z$. Hence, in any case, there exists a relation $r'' \in R_\ell$ such that $p_\ell^1(r'') = g_\rho^z$. Therefore, $p_\ell^1(r' + r'') = g_\sigma^x + g_\rho^z + g_\rho^z = g_\sigma^x$. ◀

▶ **Proposition 35.** *Let $G_\ell \xleftarrow{p_\ell^1} R_\ell$ be the matrix for the ℓ-simplices constructed by the* PiRep *algorithm. Then* $\operatorname{im}(p_\ell^1) = \ker(\alpha_\ell)$.

Proof. Let $x \in P$ and $g_{\sigma_1}^{y_1}, \dots, g_{\sigma_m}^{y_m} \in \mathcal{S}_\ell$ such that $y_i \leq x$ and $g_{\sigma_i}^{y_i} \neq g_{\sigma_j}^{y_j}$ for all $1 \leq i < j \leq m$ and assume $\alpha_\ell(x)(g_{\sigma_1}^{y_1} + \cdots + g_{\sigma_m}^{y_m}) = \sum_{i=1}^m K(y_i \leq x)(\sigma_i) = 0$. Suppose $\tau \in K_\ell(x)$ appears in this sum and $K(y_i \leq x)(\sigma_i) = \tau$ for $1 \leq i \leq q$ and $K(y_i \leq x)(\sigma_i) \neq \tau$ for $q+1 \leq i \leq m$. This implies that q is even. Let $(\tau, g_\rho^z) \in L_{act}^x$ be the active generator at x representing τ. By Lemma 33, there are relations $r_i \in R_\ell$ such that $p_\ell^1(x)(r_i) = g_\rho^z + g_{\sigma_i}^{y_i}$ for $1 \leq i \leq q$. Therefore, $p_\ell^1(x)(r_1 + \cdots + r_q) = q \cdot g_\rho^z + g_{\sigma_1}^{y_1} + \cdots + g_{\sigma_q}^{y_q} = g_{\sigma_1}^{y_1} + \cdots + g_{\sigma_q}^{y_q}$. For every other simplex $\tau' \in K_\ell(x)$ appearing in this sum we can argue analogously. Now assume $K(y_i \leq x)(\sigma_i) = 0$, i.e. $K(y_i \leq x)(\sigma_i) \notin K_\ell(x)$. By Lemma 34, there exists $r_i' \in R_\ell$ such that $p_\ell^1(x)(r_i') = g_{\sigma_i}^{y_i}$. We conclude that $g_{\sigma_1}^{y_1} + \cdots + g_{\sigma_m}^{y_m} \in \operatorname{im}\big(p_\ell^1(x)\big)$. Therefore, $\ker\big(\alpha_\ell(x)\big) \subseteq \operatorname{im}\big(p_\ell^1(x)\big)$ for all $x \in P$ which together with Proposition 32 implies that $\ker(\alpha_\ell) = \operatorname{im}(p_\ell^1)$. ◀

Finally, we show that the matrices f_ℓ^0 represent the boundary maps ∂_ℓ on the generators.

▶ **Proposition 36.** *The morphism $f_\ell^0 \colon G_\ell \to G_{\ell-1}$, constructed by the* PiRep *algorithm, lifts the morphism $\partial_\ell \colon C_\ell \to C_{\ell-1}$, i.e. the following square commutes:*

$$\begin{array}{ccc} C_\ell & \xleftarrow{\alpha_\ell} & G_\ell \\ {\scriptstyle \partial_\ell}\downarrow & & \downarrow{\scriptstyle f_\ell^0} \\ C_{\ell-1} & \xleftarrow{\alpha_{\ell-1}} & G_{\ell-1} \end{array} . \tag{19}$$

Proof. Let $x \in P$ and $g_\rho^y \in \mathcal{S}_\ell$ such that $y \leq x$. When (ρ, g_ρ^y) is added to L_{act}^y by Generator at $y \in P$, the Boundary routine of the algorithm finds active generators $(\pi_1, g_{\kappa_1}^{y_1}), \dots, (\pi_k, g_{\kappa_k}^{y_k})$, which exist by Lemma 30, such that $y_i \leq y$ and $\partial\rho = \sum_{i=1}^k \pi_i$ and defines $f_\ell^0(g_\rho^y) = \sum_{i=1}^k g_{\kappa_i}^{y_i}$. By Lemma 31, $K(y_i \leq y)(\kappa_i) = \pi_i$ for all $1 \leq i \leq k$. First, assume $K(y \leq x)(\rho) = \sigma \in K_\ell(x)$. Then

$$\partial_\ell(x) \circ \alpha_\ell(x)(g_\rho^y) = \partial_\ell(x)\big(K(y \leq x)(\rho)\big) = \partial_\ell(x)(\sigma)$$

and

$$\begin{aligned}
\alpha_{\ell\text{-}1}(x) \circ f_\ell^0(x)(g_\rho^y) &= \alpha_{\ell\text{-}1}(x)(\sum_{i=1}^{k} g_{\kappa_i}^{y_i}) = \sum_{i=1}^{k} \alpha_{\ell\text{-}1}(x)(g_{\kappa_i}^{y_i}) = \sum_{i=1}^{k} K(y_i \le x)(\kappa_i) \\
&= \sum_{i=1}^{k} K(y \le x) \circ K(y_i \le y)(\kappa_i) = \sum_{i=1}^{k} K(y \le x)(\pi_i) \\
&= \sum_{i=1}^{k} C_{\ell\text{-}1}(y \le x)(\pi_i) = C_{\ell\text{-}1}(y \le x)(\sum_{i=1}^{k} \pi_i) \\
&= C_{\ell\text{-}1}(y \le x) \circ \partial_\ell(y)(\rho) = \partial_\ell(x) \circ C_\ell(y \le x)(\rho) \\
&= \partial_\ell(x)\big(K(y \le x)(\rho)\big) = \partial_\ell(x)(\sigma)
\end{aligned}$$

where we use that

$$\begin{array}{ccc}
C_\ell(y) & \xrightarrow{C_\ell(y \le x)} & C_\ell(x) \\
{\scriptstyle \partial_\ell(y)}\downarrow & & \downarrow{\scriptstyle \partial_\ell(x)} \\
C_{\ell\text{-}1}(y) & \xrightarrow{C_{\ell\text{-}1}(y \le x)} & C_{\ell\text{-}1}(x)
\end{array}$$

commutes. Now assume $K(y \le x)(\rho) \notin K_\ell(x)$. Then $\partial_\ell(x) \circ \alpha_\ell(x)(g_\rho^y) = 0$ and

$$\alpha_{\ell\text{-}1}(x) \circ f_\ell^0(x)(g_\rho^y) = \partial_\ell(x)\big(K(y \le x)(\rho)\big) = \partial_\ell(x)\big(0\big) = 0.$$

Therefore, (19) commutes and f_ℓ^0 is a lift of ∂_ℓ to the generators G_ℓ and $G_{\ell\text{-}1}$. ◀

▶ **Theorem 37.** *The matrices p_ℓ^1 and f_ℓ^0 constructed by the algorithm* PiRep*:*

$$\begin{array}{ccccccc}
 & & \vdots & & \vdots & & \\
 & & \downarrow & & \downarrow & & \\
0 & \longleftarrow & C_2 & \xleftarrow{\alpha_2} & G_2 & \xleftarrow{p_2^1} & R_2 \\
 & & {\scriptstyle \partial_2}\downarrow & & \downarrow{\scriptstyle f_2^0} & & \\
0 & \longleftarrow & C_1 & \xleftarrow{\alpha_1} & G_1 & \xleftarrow{p_1^1} & R_1 \\
 & & {\scriptstyle \partial_1}\downarrow & & \downarrow{\scriptstyle f_1^0} & & \\
0 & \longleftarrow & C_0 & \xleftarrow{\alpha_0} & G_0 & \xleftarrow{p_0^1} & R_0
\end{array}$$

form degreewise presentations of $C_\bullet(K)$ and lifts of $\partial_\bullet$ to the generators.

Proof. By Proposition 28 and 29, α_ℓ is a well-defined epimorphism. By Proposition 35, $\ker(\alpha_\ell) = \mathrm{im}(p_\ell^1)$. Hence, p_ℓ^1 is a presentation of C_ℓ. By Proposition 36, f_ℓ^0 is a lift of ∂_ℓ to the generators. ◀

5.2 Minimality of the presentation p_ℓ^1

In this section, we show that the matrix p_ℓ^1, as constructed by the algorithm PiRep, is a minimal presentation of C_ℓ. We use the notion of projective cover in Definition 15 together with Proposition 16.

▶ **Proposition 38.** *The morphism $\alpha_\ell \colon G_\ell \to C_\ell$ is a projective cover.*

Proof. Let $\mathrm{Rad}(G_\ell)$ denote the radical of G_ℓ as defined in Definition 14. Suppose $\alpha_\ell(x)(g_{\sigma_1}^{y_1} + \cdots + g_{\sigma_m}^{y_m}) = 0$ for $g_{\sigma_i}^{y_i} \in \mathcal{S}_\ell$ such that $y_i \leq x$ and $g_{\sigma_i}^{y_i} \neq g_{\sigma_j}^{y_j}$ for $i \neq j$. If $y_j = x$, then $\sigma_j \in K_\ell(x)$ and $\sigma_j \notin \bigcup_{z \prec x} \mathrm{im}\big(K(z \prec x)\big)$. For all $i \neq j$, if $y_i = x$, then $\sigma_j \neq \sigma_i$. For all $y_i < x$, $K(y_i \leq x)(\sigma_i) \neq \sigma_j$ since any path from y_i to x contains some predecessor $z \prec x$ which would imply $K(z \prec x) \circ K(y_i \leq z)(\sigma_i) = \sigma_j$, contradicting $\sigma_j \notin \bigcup_{z \prec x} \mathrm{im}\big(K(z \prec x)\big)$. Therefore,

$$\alpha_\ell(x)(g_{\sigma_1}^{y_1} + \cdots + g_{\sigma_m}^{y_m}) = \sum_{i=1}^{m} K(y_i \leq x)(\sigma_i) = \sigma_j + \sum_{i \neq j} K(y_i \leq x)(\sigma_i).$$

But this sum cannot be zero since $K(y_i \leq x)(\sigma_i) \neq \sigma_j$ for all $y_i < x$ and $\sigma_i \neq \sigma_j$ for all $y_i = x$ with $i \neq j$. This contradiction implies $y_i < x$ for all $1 \leq i \leq m$. Therefore, $g_{\sigma_i}^{y_i} \in \mathrm{im}\big(G_\ell(z \prec x)\big)$ for some $z \prec x$ and $g_{\sigma_1}^{y_1} + \cdots + g_{\sigma_m}^{y_m} \in \mathrm{Rad}(G_\ell)(x)$. We conclude that $\ker(\alpha_\ell) \subseteq \mathrm{Rad}(G_\ell)$, which implies that α_ℓ is a projective cover of C_ℓ. ◀

Next, we show that the presentation matrix p_ℓ^1, constructed by the PIREP algorithm, almost has the structure of a $\overline{P}$-filtered graph. We will use this structure to show that p_ℓ^1 is a minimal presentation.

▶ **Definition 39.** *Suppose p_ℓ^1 is the presentation matrix generated by the algorithm. Recall that $\overline{P} := P \cup \{-\infty\}$ with $-\infty < x$ for all $x \in P$. Define the morphism of $\overline{P}$-persistence modules $\overline{p}_\ell^1 \colon R_\ell \to \overline{G}_\ell$, where $\overline{G}_\ell = G_\ell \oplus \langle \Omega \rangle$ with an additional generator Ω of grade $-\infty$. The matrix $\overline{p}_\ell^1$ is obtained from p_ℓ^1 by adding one row labeled by Ω and replacing each column of the form $r^y \mapsto g_\sigma^x$ by the column $r^y \mapsto g_\sigma^x + \Omega$. Since every column of $\overline{p}_\ell^1$ has exactly two non-zero entries, it is the boundary matrix of a $\overline{P}$-filtered graph $\mathcal{G} \colon \overline{P} \to \Delta\mathbf{Cpx}_{\leq 1}$. The graph $\mathcal{G}$ is defined by the vertices Ω born at $-\infty$ and $g_\sigma^x \in G_\ell$ born at $x \in P$ and by the edges $r^y \in R_\ell$ born at $y \in P$. All morphisms $\mathcal{G}(x < y)$ are inclusions.*

▶ Remark 40. The edges of the graph $\mathcal{G}(x)$ are exactly the edges added by the algorithm to the graphs $\mathcal{G}_{act}^y$ with $y \leq x$. The relations $r^y \mapsto g_\sigma^w + g_\tau^z$ correspond to the edges $e^y_{(g_\sigma^w, g_\tau^z)}$ added in step 2(a) of GENERATOR. The relations $r^y \mapsto g_\sigma^w$ that become $r^y \mapsto g_\sigma^w + \Omega$ in $\overline{p}_\ell^1$ correspond to the edges $e^y_{(g_\sigma^w, \Omega)}$ added in step 3(b) of COLLAPSE. Therefore, $\mathcal{G}_{act}^x$ is a subgraph of $\mathcal{G}(x)$ for every $x \in P$.

▶ **Lemma 41.** *At any grade x_i in the algorithm, after execution of* GENERATOR, *$\mathcal{F}_{act}^{x_i}$ is a spanning forest of $\mathcal{G}(x_i)$.*

Proof. By definition, it is clear that $\mathcal{F}_{act}^{x_i}$ is always a subgraph of $\mathcal{G}(x_i)$. We proceed by induction on the linear extension of the grades in P. At a minimal grade $x \in P$, the statement is obviously true as there are no edges in $\mathcal{F}_{act}^x$ or $\mathcal{G}(x)$. Assume that the statement is true for all $0 \leq j < i$. At the grade x_i, we have that $\mathcal{G}(x_i)$ is the union of all $\mathcal{G}(y)$ with $y \prec x_i$ plus the generators $g_\sigma^{x_i}$ and relations r^{x_i} born at x_i. Define $\mathcal{G}' := \bigcup_{y \prec x_i} \mathcal{G}(y)$. By inductive assumption $\mathcal{F}_{act}^y$ is a spanning forest of $\mathcal{G}(y)$. Hence, after initialization, $\mathcal{G}_{act}^{x_i} = \bigcup_{y \prec x_i} \mathcal{F}_{act}^y$ contains a spanning forest of $\mathcal{G}'$. In COLLAPSE, the algorithm adds the edge $e^{x_i}_{(\Omega, g_\rho^z)}$ to $\mathcal{G}_{act}^{x_i}$ exactly when it adds the relation $r^{x_i} \mapsto g_\rho^z$. Similarly, in GENERATOR, the algorithm adds the edge $e^{x_i}_{(g_\tau^y, g_\rho^z)}$ exactly when it adds the relation $r^{x_i} \mapsto g_\tau^y + g_\rho^z$. A relation is only added to $\mathcal{G}_{act}^{x_i}$ if it does not close a cycle. This implies that, at the grade x_i where the edge is added, it is in the spanning forest $\mathcal{F}_{act}^{x_i}$. Hence, every edge added at x_i is added to $\mathcal{G}'$ and also to $\mathcal{F}_{act}^{x_i}$. Before adding edges at x_i, a spanning forest $\mathcal{F}_{act}^{x_i}$ of $\mathcal{G}_{act}^{x_i}$ is also a spanning forest of $\mathcal{G}'$. Since all edges at x_i are added to both $\mathcal{G}'$ and $\mathcal{F}_{act}^{x_i}$, this property is preserved. The remaining vertices $g_\sigma^{x_i}$ of $\mathcal{G}(x_i)$ are added to $\mathcal{G}_{act}^{x_i}$ by GENERATOR. Since the existence of $g_\sigma^{x_i} \in \mathcal{S}$ implies

that $\sigma \notin \bigcup_{y \prec x_i} \operatorname{im}\big(K(y \prec x_i)\big)$, $g_\sigma^{x_i}$ is an isolated vertex in $\mathcal{G}_{act}^{x_i}$ and $\mathcal{G}(x_i)$ and hence also in $\mathcal{F}_{act}^{x_i}$. Therefore, the spanning forest $\mathcal{F}_{act}^{x_i}$ of $\mathcal{G}_{act}^{x_i}$, constructed in GENERATOR, is also a spanning forest of $\mathcal{G}(x_i)$. ◀

▶ Remark 42. Since $\overline{p}_\ell^1(x)$ is the boundary matrix of $\mathcal{G}(x)$, $\operatorname{coker}\big(\overline{p}_\ell^1(x)\big)$ is the space of connected components and $\ker\big(\overline{p}_\ell^1(x)\big)$ is the space of cycles in $\mathcal{G}(x)$.

▶ **Lemma 43.** *For all $x \in P$, we have $\ker\big(\overline{p}_\ell^1(x)\big) = \ker\big(p_\ell^1(x)\big)$ and $\operatorname{coker}\big(\overline{p}_\ell^1(x)\big) \cong \operatorname{coker}\big(p_\ell^1(x)\big) \oplus [\Omega]$ where $[\Omega]$ denotes the connected component of Ω.*

Proof. The matrix $p_\ell^1(x)$ is the restriction of p_ℓ^1 to all generators g_σ^y and relations r^z such that $y, z \le x$. Suppose $p_\ell^1(x)(r^{y_1} + \cdots + r^{y_k}) = (g_{\sigma_1}^{a_1} + g_{\tau_1}^{b_1}) + \cdots + (g_{\sigma_p}^{a_p} + g_{\tau_p}^{b_p}) + g_{\rho_1}^{c_1} + \cdots + g_{\rho_q}^{c_q} = 0$. Because all generators g_*^* are basis elements of $G_\ell(x)$ and a sum of basis elements over $\mathbb{F}_2$ can only be zero if every basis element appears an even number of times in the sum, the number q of single-generator relations has to be even. Also, this implies $\overline{p}_\ell^1(x)(r^{y_1} + \cdots + r^{y_k}) = (g_{\sigma_1}^{a_1} + g_{\tau_1}^{b_1}) + \cdots + (g_{\sigma_p}^{a_p} + g_{\tau_p}^{b_p}) + g_{\rho_1}^{c_1} + \cdots + g_{\rho_q}^{c_q} + q\Omega = 0$. Conversely, if $\overline{p}_\ell^1(x)(r^{y_1} + \cdots + r^{y_k}) = (g_{\sigma_1}^{a_1} + g_{\tau_1}^{b_1}) + \cdots + (g_{\sigma_p}^{a_p} + g_{\tau_p}^{b_p}) + (g_{\rho_1}^{c_1} + \Omega) + \cdots + (g_{\rho_q}^{c_q} + \Omega) = (g_{\sigma_1}^{a_1} + g_{\tau_1}^{b_1}) + \cdots + (g_{\sigma_p}^{a_p} + g_{\tau_p}^{b_p}) + g_{\rho_1}^{c_1} + \cdots + g_{\rho_q}^{c_q} + q\Omega = 0$. We get that q is even and therefore also $p_\ell^1(x)(r^{y_1} + \cdots + r^{y_k}) = (g_{\sigma_1}^{a_1} + g_{\tau_1}^{b_1}) + \cdots + (g_{\sigma_p}^{a_p} + g_{\tau_p}^{b_p}) + g_{\rho_1}^{c_1} + \cdots + g_{\rho_q}^{c_q} = 0$. Hence, $\ker\big(p_\ell^1(x)\big) = \ker\big(\overline{p}_\ell^1(x)\big)$ for all $x \in P$.

Since $\overline{G}_\ell(x) = G_\ell(x) \oplus \Omega$ we can define a projection map $\pi\colon \overline{G}_\ell(x) \to G_\ell(x)$ that forgets Ω. We obtain

$$\begin{array}{ccccc}
\operatorname{coker}\big(\overline{p}_\ell^1(x)\big) & \xleftarrow{\phi} & \overline{G}_\ell(x) & \xleftarrow{\overline{p}(x)} & R_\ell(x) \\
{\scriptstyle \operatorname{coker}\pi}\downarrow & & \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle =} \\
\operatorname{coker}\big(p_\ell^1(x)\big) & \xleftarrow{\psi} & G_\ell(x) & \xleftarrow{p_\ell^1(x)} & R_\ell(x)
\end{array}$$

Since $\overline{p}_\ell^1(x)$ is the boundary matrix of a graph, the cokernel has a basis consisting of the connected components of this graph. If $[\Omega]$ denotes the equivalence class of Ω in $\operatorname{coker}\big(\overline{p}_\ell^1(x)\big)$, then $(\operatorname{coker}\pi)([\Omega]) = [\pi(\Omega)] = 0$. Since $\psi \circ \pi = (\operatorname{coker}\pi) \circ \phi$ is an epimorphism, also $\operatorname{coker}\pi$ is an epimorphism and $\operatorname{coker}\big(\overline{p}_\ell^1(x)\big) \cong \operatorname{im}(\operatorname{coker}\pi) \oplus \ker(\operatorname{coker}\pi) \cong \operatorname{coker}\big(p_\ell^1(x)\big) \oplus \ker(\operatorname{coker}\pi)$. Since $\dim\ker\big(p_\ell^1(x)\big) = \dim\ker\big(\overline{p}_\ell^1(x)\big)$, we also get $\dim\operatorname{im}\big(p_\ell^1(x)\big) = \dim\operatorname{im}\big(\overline{p}_\ell^1(x)\big)$ and, moreover, $\dim\operatorname{coker}\big(p_\ell^1(x)\big) = \dim G_\ell(x) - \dim\operatorname{im}\big(p_\ell^1(x)\big) = \dim\overline{G}_\ell(x) - 1 - \dim\operatorname{im}\big(\overline{p}_\ell^1(x)\big) = \dim\operatorname{coker}\big(\overline{p}_\ell^1(x)\big) - 1$. Therefore, $\ker(\operatorname{coker}\pi)$ is one-dimensional, generated by $[\Omega]$, and $\operatorname{coker}\big(\overline{p}_\ell^1(x)\big) \cong \operatorname{coker}\big(p_\ell^1(x)\big) \oplus [\Omega]$. ◀

▶ **Proposition 44.** *The morphism $p_\ell^1\colon R_\ell \to \ker(\alpha_\ell)$ is a projective cover.*

Proof. By construction R_ℓ is projective. By Proposition 35, $\ker(\alpha_\ell) = \operatorname{im}(p_\ell^1)$. Thus, $p_\ell^1\colon R_\ell \to \ker(\alpha_\ell)$ is an epimorphism.

Let $x \in P$ and $r^{y_1}, \ldots, r^{y_k} \in R_\ell$ such that $y_i \le x$ and $p_\ell^1(x)(r^{y_1} + \cdots + r^{y_k}) = 0$. W.l.o.g. assume that the sum $r^{y_1} + \cdots + r^{y_k}$ is reduced, i.e., even numbers of copies of the same relation (same formal basis element of R_ℓ) are canceled. Note that different formal basis elements r^y of R_ℓ can have the same grade y. By Lemma 43, we obtain that $\overline{p}_\ell^1(x)(r^{y_1} + \cdots + r^{y_k}) = 0$ and, thus, $e^{y_1}, \ldots, e^{y_k}$ is a cycle in the graph $\mathcal{G}(x)$. For any edge e^{y_i} such that $y_i = x$ the corresponding relation r^x is added at the grade x during the algorithm. Since this only happens when the edge e^x is added to $\mathcal{G}_{act}^x$, any edge e^x on the cycle is in $\mathcal{G}_{act}^x$. If $y_i < x$ then there exists a $z \prec x$ such that $e^{y_i} \in \mathcal{G}(z)$. By Lemma 41, $\mathcal{F}_{act}^z$ is a spanning forest of $\mathcal{G}(z)$. If $e^{y_i} \notin \mathcal{F}_{act}^z$, then there exists a path $\gamma_1^i, \ldots, \gamma_{l_i}^i$ in $\mathcal{F}_{act}^z$ connecting the endpoints of e^{y_i}. So,

by construction of $\mathcal{G}^x_{act}$ in Collapse, in any case, there exists a path in $\mathcal{G}^x_{act}$ connecting the endpoints of the edge e^{y_i}. Hence, we can replace any edge on the cycle $e^{y_1}, \dots, e^{y_k}$ that is not in $\mathcal{G}^x_{act}$ by a path in $\mathcal{G}^x_{act}$. This yields another cycle that lies completely in $\mathcal{G}^x_{act}$. If there are edges e^x on the initial cycle, they will not be replaced and are also on the cycle in $\mathcal{G}^x_{act}$. But this implies that there is an edge e^x, added at the grade x, closing the cycle. This is a contradiction, since the algorithm does not add edges that close cycles in $\mathcal{G}^x_{act}$. Therefore, $y_i < x$ for all $1 \leq i \leq k$. Recalling Definition 14, we get that $r^{y_1} + \dots + r^{y_k} \in \mathrm{Rad}(R_\ell)(x)$ and, moreover, $\ker\big(p^1_\ell(x)\big) \subseteq \mathrm{Rad}(R_\ell)(x)$. This implies that $\ker(p^1_\ell) \subseteq \mathrm{Rad}(R_\ell)$ and, thus, p^1_ℓ is a projective cover of $\ker(\alpha_\ell)$. ◀

▶ **Theorem 45.** *The matrix p^1_ℓ, computed by the algorithm* PiRep*, is a minimal presentation of C_ℓ.*

Proof. By Theorem 37, p^1_ℓ is a presentation. By Proposition 38 and 44, $\alpha_\ell \colon G_\ell \to C_\ell$ is a projective cover of C_ℓ and $p^1_\ell \colon R_\ell \to \ker(\alpha_\ell)$ is a projective cover of $\ker(\alpha_\ell)$. Therefore, by Proposition 16, p^1_ℓ is a minimal presentation. ◀

5.3 Correctness of computing p^2_ℓ and minimality

In this section, we show that the matrix p^2_ℓ, as constructed by the PiRep algorithm, yields an asymptotically minimal partial projective resolution of C_ℓ up to the second term. With the additional call of Reduce, we will show that it is a minimal partial projective resolution of C_ℓ up to the second term.

▶ **Proposition 46.** $\mathrm{im}(p^2_\ell) \subseteq \ker(p^1_\ell)$.

Proof. Suppose rr^x is a relation of relations in RR_ℓ such that $p^2_\ell(rr^x) = r^{z_0} + \dots + r^{z_s}$. The relation of relations is added by the algorithm if and only if the corresponding edges $e^{z_0}, \dots, e^{z_s}$ form a cycle in the graph $\mathcal{G}^x_{act}$ at the grade x. Since $\mathcal{G}^x_{act}$ is a subgraph of $\mathcal{G}(x)$, the edges $e^{z_0}, \dots, e^{z_s}$ also form a cycle in $\mathcal{G}(x)$. By Definition 39, we get that $\overline{p}^1_\ell(y)(r^{z_0} + \dots + r^{z_s}) = 0$ and, by Lemma 43, also $p^1_\ell(y)(r^{z_0} + \dots + r^{z_s}) = 0$ for all $y \geq x$. Hence, $p^1_\ell \circ p^2_\ell(rr^x) = 0$ and $\mathrm{im}(p^2_\ell) \subseteq \ker(p^1_\ell)$. ◀

▶ **Proposition 47.** $\ker(p^1_\ell) \subseteq \mathrm{im}(p^2_\ell)$.

Proof. We proceed by induction on the grades in the linear extension of P. Assume that $\ker\big(p^1_\ell(z)\big) \subseteq \mathrm{im}\big(p^2_\ell(z)\big)$ for all $z < x$. This is obviously true for the minimal elements in P. Let $r^{y_1}, \dots, r^{y_n} \in R_\ell$ so that $y_i \leq x$ for all $1 \leq i \leq n$ and $p^1_\ell(x)(r^{y_1} + \dots + r^{y_n}) = 0$. By Lemma 43, $e^{y_1}, \dots, e^{y_n}$ form a cycle in $\mathcal{G}(x)$. By the proof of Proposition 44, $\ker\big(p^1_\ell(x)\big) \subseteq \mathrm{Rad}(R_\ell)(x)$. Thus, for each $1 \leq i \leq n$, we have $y_i < x$ and $e^{y_i} \in \mathcal{G}(y)$ for some $y \prec x$. Suppose $e^{y_i} \notin \mathcal{G}^x_{act}$. Then $e^{y_i} \notin \mathcal{F}^y_{act}$. Since, by Lemma 41, $\mathcal{F}^y_{act}$ is a spanning forest of $\mathcal{G}(y)$, there exists a path $e^{z^i_1}, \dots, e^{z^i_{k_i}}$ in $\mathcal{F}^y_{act}$ that connects the endpoints of e^{y_i}. Hence, this path together with e^{y_i} forms a cycle in $\mathcal{G}(y)$. By Lemma 43, $p^1_\ell(y)(r^{y_i} + r^{z^i_1} + \dots + r^{z^i_{k_i}}) = 0$. Thus, by induction, there exists $rr_i \in RR_\ell(y)$ such that $p^2_\ell(y)(rr_i) = r^{y_i} + r^{z^i_1} + \dots + r^{z^i_{k_i}}$. We now replace the edge e^{y_i} in the cycle $e^{y_1}, \dots, e^{y_n}$ by the path $e^{z^i_1}, \dots, e^{z^i_{k_i}}$ which lies in $\mathcal{G}^x_{act}$. If we replace

each edge $e^{y_i} \notin \mathcal{G}^x_{act}$ by such a path, we obtain a cycle $e^{z_1}, \dots, e^{z_l}$ in $\mathcal{G}^x_{act}$ since

$$\begin{aligned} p^1_\ell(x)\big(r^{z_1} + \cdots + r^{z_l}\big) &= p^1_\ell(x)\Big(\sum_{r^{y_i} \in \mathcal{G}^x_{act}} r^{y_i} + \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} r^{z^i_1} + \cdots + r^{z^i_{k_i}}\Big) \\ &= p^1_\ell(x)\Big(\sum_{r^{y_i} \in \mathcal{G}^x_{act}} r^{y_i}\Big) + \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} p^1_\ell(x)\big(r^{z^i_1} + \cdots + r^{z^i_{k_i}}\big) \\ &= p^1_\ell(x)\Big(\sum_{r^{y_i} \in \mathcal{G}^x_{act}} r^{y_i}\Big) + p^1_\ell(x)\Big(\sum_{r^{y_i} \notin \mathcal{G}^x_{act}} r^{y_i}\Big) \\ &= p^1_\ell(x)(r^{y_1} + \cdots + r^{y_n}) = 0. \end{aligned}$$

The algorithm RelRel computes a cycle basis for $\mathcal{G}^x_{act}$ and adds the computed cycles to p^2_ℓ. Hence, there exists $rr \in RR_\ell(x)$ such that $p^2_\ell(x)(rr) = r^{z_1} + \cdots + r^{z_l}$. Combining this cycle with the cycles for the replaced edges yields

$$\begin{aligned} p^2_\ell(x)(rr + \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} rr_i) &= p^2_\ell(x)(rr) + \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} p^2_\ell(x)(rr_i) \\ &= r^{z_1} + \cdots + r^{z_l} + \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} r^{y_i} + r^{z^i_1} + \cdots + r^{z^i_{k_i}} \\ &= \sum_{r^{y_i} \in \mathcal{G}^x_{act}} r^{y_i} + \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} r^{z^i_1} + \cdots + r^{z^i_{k_i}} \\ &+ \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} r^{y_i} + \sum_{r^{y_i} \notin \mathcal{G}^x_{act}} r^{z^i_1} + \cdots + r^{z^i_{k_i}} \\ &= r^{y_1} + \cdots + r^{y_n}. \end{aligned}$$

Therefore, $r^{y_1} + \cdots + r^{y_n} \in \operatorname{im}\big(p^2_\ell(x)\big)$ and, moreover, $\ker(p^1_\ell) \subseteq \operatorname{im}(p^2_\ell)$. ◀

▶ **Proposition 48.** *There exists a family of poset towers $K^m \colon P^m \to \mathbf{SCpx}$, $m \geq 2$, with $n = \Theta(m)$ simplex generators and $t_1 = \Theta(m^2)$ edges in the Hasse diagram of P^m, such that the second term RR_0 in a minimal projective resolution of $C_0(K^m)$ has $\Theta(nt_1)$ elementary projective summands. In particular, the bound $O(nt_1)$ on the number of columns of p^2_ℓ computed by* PiRep *without* Reduce *is asymptotically optimal.*

Proof. Let $A = \{a_1, \dots, a_m\}$, $B = \{b_1, \dots, b_m\}$, $C = \{c_1, \dots, c_m\}$ and let $P^m = A \cup B \cup C$ be the poset with relations $a_i \prec b_j \prec c_k$ for all $1 \leq i, j, k \leq m$. Thus P^m has $t_0 = 3m$ grades and $t_1 = 2m^2$ edges in its Hasse diagram. We define a poset tower $K^m \colon P^m \to \mathbf{SCpx}$ consisting of a single vertex v at every grade and identity maps on this vertex along every edge. Equivalently, $K^m(x) = \{v\}$ for all $x \in P^m$, and $K^m(x \leq y)(v) = v$ whenever $x \leq y$. Then v has exactly one simplex generator $g^{a_i}_v$ at each grade $a_i \in A$ and no simplex generators at the grades in $B \cup C$. Hence $\mathcal{S}_0 = \{g^{a_1}_v, \dots, g^{a_m}_v\}$, $\mathcal{C} = \emptyset$ and the input size is $n = |\mathcal{S}| + |\mathcal{C}| = m$.

At a grade b_j, the vector space $G_0(b_j)$ has basis $g^{a_1}_v, \dots, g^{a_m}_v$ whereas $C_0(K_m)(b_j)$ is one-dimensional. The map $\alpha_0(b_j) \colon G_0(b_j) \to C_0(K_m)(b_j)$ sends every basis element $g^{a_i}_v$ to v. Therefore, $\ker\big(\alpha_0(b_j)\big)$ has dimension $m - 1$. Since $\ker\big(\alpha_0(a_i)\big) = 0$ for all $1 \leq i \leq m$, there must be $m - 1$ relations born at b_j. Since we proved in Theorem 45 that PiRep computes a minimal presentation this can also be seen by applying the algorithm, which has to identify the n copies $(v, g^{a_i}_v)$ at each grade b_j. The algorithm adds edges until we obtain a spanning tree T_j on the vertices $g^{a_1}_v, \dots, g^{a_m}_v$, which requires exactly $n - 1$ edges.

Now consider a grade c_k. The relation graph at c_k contains the union of the m spanning trees $T_1 \cup \cdots \cup T_m$ on the same set of m vertices. The edges are indexed by the relations

born at the different grades b_j, so they are distinct columns of p_0^1, even if some of them have the same endpoints. The graph $T_1 \cup \cdots \cup T_m$ is connected and has m vertices and $m(m-1)$ edges. Therefore, its cycle space has dimension $m(m-1) - m + 1 = (m-1)^2$. By Lemma 43, this cycle space is equal to $\ker\big(p_0^1(c_k)\big)$. Hence $\dim \ker\big(p_0^1(c_k)\big) = (m-1)^2$.

Moreover, for every $x < c_k$ we have $\ker\big(p_0^1(x)\big) = 0$: this is clear at the grades a_i, where no relations are present, and at the grades b_j, where the relation graph is a tree. It follows that any projective cover of $\ker(p_0^1)$ must contain at least $(m-1)^2$ elementary projective summands born at c_k. Since the grades $c_1, \ldots, c_m$ are pairwise incomparable, summands born at one of them cannot contribute to any other one. Thus, the second term RR_0 in a minimal projective resolution has $m(m-1)^2$ elementary projective summands.

Finally, since $n = m$ and $t_1 = 2m^2$, we obtain $m(m-1)^2 = \Theta(m^3) = \Theta(nt_1)$. This proves that the $O(nt_1)$ bound on the number of columns of p_ℓ^2 is asymptotically optimal. ◀

▶ Remark 49. In the following Proposition 50, we show that, with the additional call of the routine Reduce, the algorithm computes a minimal resolution up to the second term. In other words, the algorithm computes a projective cover p_ℓ^2 of $\ker(p_\ell^1)$. This step can also be interpreted geometrically in terms of the $\overline{P}$-filtered graph $\mathcal{G}$ represented by $\overline{p}_\ell^1$. The kernel of $\overline{p}_\ell^1$ is the cycle space of $\mathcal{G}$ and equal to the kernel of p_ℓ^1 by Lemma 43. Thus, the task of computing a projective cover of $\ker(p_\ell^1)\colon P \to \mathbf{Vec}$ is equivalent to computing a minimal set of generators of $H_1(\mathcal{G})\colon P \to \mathbf{Vec}$. Each column of p_ℓ^2, added by the algorithm, can be viewed as a 2-cell filling a cycle in $\mathcal{G}$. The set of 2-cells generated in this way is minimal if a 2-cell added at grade x never closes a void, i.e., never creates a class in H_2. This is exactly a reformulation of the projective cover condition. The kernel of p_ℓ^2 corresponds to H_2 classes or voids, and if we close a void at grade x by adding a 2-cell, then this 2-cell is a linear combination of already existing 2-cells and thus redundant. Reducing the matrix $A^{\leq x}$ in the Reduce routine and checking if columns added at grade x can be reduced to zero is exactly checking if the 2-cell closes a void. Because this is essentially a two-dimensional problem, it cannot be solved as efficiently as the other tasks. It is thus remarkable that we can still compute an asymptotically optimal p_ℓ^2 without checking this condition.

▶ **Proposition 50.** *If* Reduce *is called, p_ℓ^2 is a projective cover of* $\ker(p_\ell^1)$.

Proof. Let $x \in P$ and $rr^{y_1}, \ldots, rr^{y_m} \in RR_\ell$ such that $y_i \leq x$ and $p_\ell^2(x)(rr^{y_1} + \cdots + rr^{y_m}) = 0$. At the grade x in the algorithm, at the start of Reduce, all rr^{y_i} are already in RR_ℓ. The matrix $A^{\leq x}$ representing $p_\ell^2(x)$ consists of all columns rr^z of p_ℓ^2, such that $z \leq x$, in an order respecting the linear extension of P. Thus, the columns rr^z such that $z = x$ come last in $A^{\leq x}$. At the end of Reduce, the matrix $A^{\leq x}$ is column reduced from left to right (with respect to the order of the columns in the matrix).

We now prove by contradiction that $y_i < x$ for all i. Suppose rr^{y_i} is the rightmost column among $rr^{y_1}, \ldots, rr^{y_m}$ in $A^{\leq x}$ such that $y_i = x$. The assumption $p_\ell^2(x)(rr^{y_1} + \cdots + rr^{y_m}) = 0$ implies that the column rr^{y_i} is a linear combination of the columns $\{rr^{y_1}, \ldots, rr^{y_m}\} \setminus rr^{y_i}$. But, since all columns $\{rr^{y_1}, \ldots, rr^{y_m}\} \setminus rr^{y_i}$ come before rr^{y_i} in $A^{\leq x}$, the column rr^{y_i} must have been reduced to zero in Reduce. Therefore, at the end of Reduce it is deleted from p_ℓ^2 contradicting $rr^{y_i} \in RR_\ell$.

We conclude that $y_i < x$ for all $1 \leq i \leq m$ and $rr^{y_1} + \cdots + rr^{y_m} \in \mathrm{Rad}(RR_\ell)(x)$. Therefore, $\ker(p_\ell^2) \subseteq \mathrm{Rad}(RR_\ell)$ and p_ℓ^2 is a projective cover of $\ker(p_\ell^1)$. ◀

▶ **Theorem 51.** *Let p_ℓ^1 and p_ℓ^2 be the matrices computed by the algorithm* PiRep. *Then the sequence*

$$0 \longleftarrow C_\ell(K) \xleftarrow{\alpha_\ell} G_\ell \xleftarrow{p_\ell^1} R_\ell \xleftarrow{p_\ell^2} RR_\ell$$

is an asymptotically minimal projective resolution of $C_\ell(K)$ up to the second term. If Reduce *is called in* RelRel*, then it is a minimal projective resolution of $C_\ell(K)$ up to the second term.*

Proof. By construction G_ℓ, R_ℓ and RR_ℓ are projective. By Proposition 29, 35, 46 and 47 the sequence is exact. Hence, it is a projective resolution up to the second term. By Theorem 45, p^1_ℓ is a minimal presentation of $C_\ell(K)$ and, by Proposition 48, p^2_ℓ is asymptotically minimal. If Reduce is called in RelRel, then, by Proposition 50, p^2_ℓ is a projective cover of $\ker(p^1_\ell)$. Therefore, by Proposition 16, together with p^1_ℓ it forms a minimal projective resolution up to the second term. ◀

5.4 Correctness of computing f^1_ℓ and $\vartheta_{\ell+1}$

After proving the correctness of the computation of p^1_ℓ, p^2_ℓ, and f^0_ℓ by the PiRep algorithm, the only missing ingredients for a PiRep of $H_\ell(K)$ are the matrices for f^1_ℓ and $\vartheta_{\ell+1}$ in the block matrices in (16). The morphisms f^1_ℓ and $\vartheta_{\ell+1}$ need to satisfy $p^1_{\ell\text{-}1} \circ f^1_\ell = f^0_\ell \circ p^1_\ell$ and $p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1} = f^0_\ell \circ f^0_{\ell+1}$, respectively. The routine Lift efficiently solves for the unknown maps f^1_ℓ and $\vartheta_{\ell+1}$ in these equations by exploiting the special structure of $p^1_{\ell\text{-}1}$. It solves a set of linear systems of equations with coefficient matrices induced by the edges of certain spanning trees. In this section, we prove that we can reduce the problem of solving these equations of morphisms of projective modules to such linear systems, which will then establish the correctness of the routine Lift.

Both the construction of f^1_ℓ and $\vartheta_{\ell+1}$ can be described as finding a solution of a P-graded linear system $A \circ X = B$, as defined in Definition 9. For the unknown $X = f^1_\ell$, we set $A = p^1_{\ell\text{-}1}$ and $B = f^0_\ell \circ p^1_\ell$. For the unknown $X = \vartheta_{\ell+1}$, we set $A = p^1_{\ell\text{-}1}$ and $B = f^0_\ell \circ f^0_{\ell+1}$. As discussed in Section 3, Proposition 12(2) and (3) imply that these systems have solutions. By Proposition 10, we can solve a P-graded linear system by solving m standard linear systems $A^{\leq y_j} X_j^{\leq y_j} = B_j$ for submatrices $A^{\leq y_j}$ of A. In general, solving the m linear systems in Proposition 10 would cost cubic time per system, but, in our case, we can exploit the special structure of the coefficient matrix $A = p^1_{\ell\text{-}1}$ to solve them in linear time. By Theorem 24, we obtain $p^1_{\ell\text{-}1}$ from a matrix $\overline{p}^1_{\ell\text{-}1}$ by removing a distinguished row Ω where $\overline{p}^1_{\ell\text{-}1}$ is the boundary matrix of a $\overline{P}$-filtered graph $\mathcal{G}\colon \overline{P} \to \mathbf{\Delta Cpx}_{\leq 1}$. This means that our coefficient matrix A has either one or two non-zero entries per column and its columns are labeled by the edges in the $\overline{P}$-filtered graph. First, we show that we can directly work with the coefficient matrix induced by $\overline{p}^1_{\ell\text{-}1}$.

▶ **Proposition 52.** *Let $A = \begin{pmatrix} T & S \end{pmatrix}$ be a block matrix, where T is a block of columns with two non-zero entries and S is a block of columns with one non-zero entry. Let $\overline{A} \coloneqq \begin{pmatrix} T & S \\ \overline{0} & \overline{1} \end{pmatrix}$ be a block matrix, where $\overline{0} = \begin{pmatrix} 0 & \cdots & 0 \end{pmatrix}$ and $\overline{1} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}$. Moreover, let $\overline{B} \coloneqq \begin{pmatrix} B \\ \overline{1}B \end{pmatrix}$. Then $AX = B$ if and only if $\overline{A}X = \overline{B}$.*

Proof. Since A and B are obtained by removing the last row of $\overline{A}$ and $\overline{B}$, respectively, it is clear that a solution X of $\overline{A}X = \overline{B}$ is also a solution of $AX = B$. Conversely, let $X = \begin{pmatrix} X_T \\ X_S \end{pmatrix}$ be a solution of $AX = \begin{pmatrix} T & S \end{pmatrix} \begin{pmatrix} X_T \\ X_S \end{pmatrix} = TX_T + SX_S = B$. Because T has exactly two non-zero entries per column and S has exactly one non-zero entry per column, we obtain $\overline{1}B = \overline{1}TX_T + \overline{1}SX_S = \overline{0}X_T + \overline{1}X_S = \overline{1}X_S$. Therefore, $\overline{A}X = \begin{pmatrix} T & S \\ \overline{0} & \overline{1} \end{pmatrix} \begin{pmatrix} X_T \\ X_S \end{pmatrix} =$

$$\begin{pmatrix} TX_T + SX_S \\ \overline{1}X_S \end{pmatrix} = \begin{pmatrix} B \\ \overline{1}B \end{pmatrix} = \overline{B}.$$ ◀

By Definition 39, if $A = p^1_{\ell\text{-}1}$, we obtain $\overline{A} = \overline{p}^1_{\ell\text{-}1}$ for the matrix in Proposition 52. So we can consider the system $\overline{A}X = \overline{B}$, where $\overline{A} = \overline{p}^1_{\ell\text{-}1}$ and $\overline{B}$ is constructed from $B = f^0_\ell \circ p^1_\ell$ or $B = f^0_\ell \circ f^0_{\ell+1}$ as in Proposition 52. For $\overline{A} = \overline{p}^1_{\ell\text{-}1}$, the submatrices $\overline{A}^{\leq y_j}$ of Proposition 10 are exactly the boundary matrices of $\mathcal{G}(y_j)$. The following proposition shows that it is enough to solve the system $\overline{A}^{\leq y_j} X_j^{\leq y_j} = \overline{B}_j$ for a submatrix of $\overline{A}^{\leq y_j}$ and a subvector of $X_j^{\leq y_j}$ corresponding to the edges of a spanning forest.

▶ **Proposition 53.** *Let $Ax = b$ be a system of linear equations such that A is the boundary matrix of a graph G (without self-loops) and $b \in \operatorname{im}(A)$. Let $F = \bigcup_{i=1}^n F^i$ be a spanning forest of G with connected components $F^1, \dots, F^n$. Moreover, let A^i be the submatrix of A corresponding to the edges and vertices in F^i, x^i the subvector of variables x corresponding to the edges in F^i, x^r the subvector of x corresponding to edges in $G \setminus F$, and b^i the subvector of b corresponding to the vertices in F^i. Then $b^i \in \operatorname{im}(A^i)$ and $A^i x^i = b^i$ and $x^r = 0$ implies $Ax = b$.*

Proof. Let $G^1, \dots, G^n$ be the connected components of the graph G. Let $\hat{A}^i$ be the submatrix of A corresponding to the vertices and edges in G^i, $\hat{x}^i$ the subvector of variables x corresponding to the edges in G^i, and $\hat{b}^i$ the subvector of b corresponding to the vertices in G^i. After reordering the columns and rows of A, x and b we can write the system as

$$Ax = \begin{pmatrix} \hat{A}^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{A}^n \end{pmatrix} \begin{pmatrix} \hat{x}^1 \\ \vdots \\ \hat{x}^n \end{pmatrix} = \begin{pmatrix} \hat{A}^1\hat{x}^1 \\ \vdots \\ \hat{A}^n\hat{x}^n \end{pmatrix} = \begin{pmatrix} \hat{b}^1 \\ \vdots \\ \hat{b}^n \end{pmatrix} = b,$$

where the coefficient matrix is in block-diagonal form because the connected components $G^1, \dots, G^n$ of G partition the vertices and edges into a disjoint union. Therefore, $Ax = b$ has a solution if and only if $\hat{A}^i\hat{x}^i = \hat{b}^i$ has a solution for all $1 \leq i \leq n$. In other words, $b \in \operatorname{im}(A)$ if and only if $\hat{b}^i \in \operatorname{im}(\hat{A}^i)$.

Since F is a spanning forest of G, each connected component F^i of F must be a spanning tree of some connected component of G. W.l.o.g. assume F^i is a spanning tree of the component G^i. Hence, A^i is the submatrix of $\hat{A}^i$ corresponding to the edges in F^i. If C^i is the submatrix of $\hat{A}^i$ corresponding to the edges in $G^i \setminus F^i$, then we can write $\hat{A}^i = \begin{pmatrix} A^i & C^i \end{pmatrix}$. Every edge in $G^i \setminus F^i$ closes a cycle in G^i with the unique path connecting its endpoints in the spanning tree F^i. Thus, every column of C^i is a linear combination of columns in A^i, given by the columns corresponding to the edges on this path. This implies that $\operatorname{im}(A^i) = \operatorname{im}(\hat{A}^i)$ and, since G^i and F^i have the same vertices, $b^i = \hat{b}^i \in \operatorname{im}(\hat{A}^i) = \operatorname{im}(A^i)$.

After reordering the columns and rows of A, x and b again, we can write the system as

$$Ax = \begin{pmatrix} A^1 & \cdots & 0 & C^1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & A^n & 0 & \cdots & C^n \end{pmatrix} \begin{pmatrix} x^1 \\ \vdots \\ x^n \\ x^r \end{pmatrix} = \begin{pmatrix} b^1 \\ \vdots \\ b^n \end{pmatrix} = b,$$

where $C^1, \dots, C^n$ are the submatrices and x^r is the subvector corresponding to all edges in $G \setminus F$. Since the columns of $A^1, \dots, A^n$ correspond to edges in the connected components of the spanning forest F, the block of A containing them is block-diagonal. Therefore, $A^1x^1 = b^1, \dots, A^nx^n = b^n$ and $x^r = 0$ implies $Ax = b$. ◀

By Proposition 53, we can reduce the problem of solving $\overline{A}^{\leq y_j} X_{\overline{j}}^{\leq y_j} = \overline{B}_j$ to the problem of solving equations $F^k x = b^k$ where the F^k correspond to the connected components of a spanning forest of $\mathcal{G}(y_j)$. In Appendix C, we discuss the routine TreeSolve, which can solve such a linear system $F^k x = b^k$ in linear time. The routine Lift computes solutions of $p^1_{\ell\text{-}1} \circ f^1_\ell = f^0_\ell \circ p^1_\ell$ and $p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1} = f^0_\ell \circ f^0_{\ell+1}$ by solving the systems $F^k x = b^k$ using the spanning forests $\mathcal{F}^{y_j}_{act}$ of $\mathcal{G}(y_j)$ maintained by the PiRep algorithm and the routine TreeSolve.

▶ **Theorem 54.** *The routine* Lift *computes morphisms* f^1_ℓ *and* $\vartheta_{\ell+1}$ *such that* $p^1_{\ell\text{-}1} \circ f^1_\ell = f^0_\ell \circ p^1_\ell$ *and* $p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1} = f^0_\ell \circ f^0_{\ell+1}$.

Proof. By Lemma 41, for each $x \in P$, after the call of Generator, $\mathcal{F}^x_{act}$ is a spanning forest of $\mathcal{G}(x)$ and all updates of p^1 and f^0 at grade x have already been computed. By Proposition 52, we can replace the coefficient matrix $p^1_{\ell\text{-}1}$ in $p^1_{\ell\text{-}1} \circ f^1_\ell = f^0_\ell \circ p^1_\ell$ and $p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1} = f^0_\ell \circ f^0_{\ell+1}$ by $\overline{p}^1_{\ell\text{-}1}$ if we also add an additional row $\overline{1} \circ f^0_\ell \circ p^1_\ell$ and $\overline{1} \circ f^0_\ell \circ f^0_{\ell+1}$, corresponding to the row Ω in $\overline{p}^1_{\ell\text{-}1}$, to the right-hand sides.

A matrix representation of the morphism $f^1_\ell \colon R_\ell \to R_{\ell\text{-}1}$ has columns indexed by the relations r^y spanning R_ℓ and rows indexed by the relations spanning $R_{\ell\text{-}1}$. Let $r^{y_1}, \dots, r^{y_s}$ be the relations spanning R_ℓ. Then we can rewrite the equation $\overline{p}^1_{\ell\text{-}1} \circ f^1_\ell = \begin{pmatrix} f^0_\ell \circ p^1_\ell \\ \overline{1} \circ f^0_\ell \circ p^1_\ell \end{pmatrix}$ as $\overline{p}^1_{\ell\text{-}1} \circ f^1_\ell[:, r^{y_j}] = \begin{pmatrix} f^0_\ell \circ p^1_\ell[:, r^{y_j}] \\ \overline{1} \circ f^0_\ell \circ p^1_\ell[:, r^{y_j}] \end{pmatrix}$ for all $1 \leq j \leq s$. By Proposition 10, we can drop all columns of $\overline{p}^1_{\ell\text{-}1}$ and rows of $f^1_\ell[:, r^{y_j}]$ which are indexed by $r^z \in R_{\ell\text{-}1}$ such that $z \not\leq y_j$. The remaining columns of $\overline{p}^1_{\ell\text{-}1}$ correspond exactly to the edges in $\mathcal{G}(y_j)$. The equation becomes an ordinary system of linear equations where the coefficient matrix is the boundary matrix of $\mathcal{G}(y_j)$, the variables correspond to the edges in $\mathcal{G}(y_j)$ and the right-hand side corresponds to the vertices of $\mathcal{G}(y_j)$. At the grade $x = y_j$ in the algorithm, where we add r^{y_j}, all r^z in $R_{\ell\text{-}1}$ with $z \leq y_j$ are already present. Therefore, all ingredients of the linear system at y_j are available. By Proposition 53, we can solve such a system by restricting the coefficient matrix to the submatrix induced by a spanning forest and instead solve the system for each connected component of the forest.

In step $1(a)$, we compute the right-hand side with the additional row corresponding to Ω. Note that all non-zero entries in the vector $f^0_\ell \circ p^1_\ell[:, r^{y_j}]$ correspond to g^z_σ with $z \leq y_j$, i.e., to vertices in $\mathcal{F}^{y_j}_{act}$. In step $1(b)$, we call the routine TreeSolve (see Appendix C) for each connected component C of the spanning forest $\mathcal{F}^{y_j}_{act}$. By Theorem 57, this yields a solution of $\overline{p}^1_{\ell\text{-}1} \circ f^1_\ell[:, r^{y_j}] = \begin{pmatrix} f^0_\ell \circ p^1_\ell[:, r^{y_j}] \\ \overline{1} \circ f^0_\ell \circ p^1_\ell[:, r^{y_j}] \end{pmatrix}$ and, thus, also of $p^1_{\ell\text{-}1} \circ f^1_\ell[:, r^{y_j}] = f^0_\ell \circ p^1_\ell[:, r^{y_j}]$. After processing all grades of P, we obtain a solution of $p^1_{\ell\text{-}1} \circ f^1_\ell = f^0_\ell \circ p^1_\ell$. The proof for $p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1} = f^0_\ell \circ f^0_{\ell+1}$ is analogous. ◀

6 Conclusion

We have developed an algorithm that converts the simplicial data of a poset tower into a projective implicit representation of its persistent homology. The main difficulty is that the chain modules of a poset tower are not necessarily projective, so the boundary maps cannot, in general, be read directly as graded matrices. Our construction overcomes this by first computing partial projective resolutions of the chain modules, then lifting the boundary maps to these resolutions, and finally assembling the resulting data into a PiRep using an additional correction term. In particular, the construction provides efficient algorithms for

computing minimal presentations of the chain modules $C_\ell(K)$ arising from poset towers, as well as asymptotically minimal projective resolutions of these modules up to the second term.

The central observation behind the algorithm is that the relations in the chain modules are not arbitrary. They arise from identifications of simplex generators and can be organized in relation graphs. This graph structure is used repeatedly: to compute minimal presentations by avoiding redundant cycle-closing relations, to compute relations among relations from cycles, and to solve the linear systems needed for the lifted boundary maps and the correction term. Thus, rather than applying general-purpose algebraic reduction to the chain modules, the algorithm exploits the combinatorial structure inherited from the poset tower, leading to a significant improvement in efficiency.

Several directions remain open. Although the algorithm is general for finite posets, indexing posets arising in applications might have additional structure. We expect that the general algorithm developed here can serve as a starting point for more specialized algorithms and for practical implementations of persistent homology over more general indexing posets.

Moreover, one may ask whether the local construction carried out here for a fixed homological degree can be extended to a more global replacement of the persistent chain complex $C_\bullet(K)$ by a chain complex of projective modules. Such a construction is unlikely to admit an efficient general-purpose solution for arbitrary finite posets. Nevertheless, for special classes of indexing posets, it would be interesting to investigate whether the combinatorial structure of poset towers can be exploited to construct projective resolutions, or partial projective resolutions, of the persistent chain complex in a systematic way.

References

1 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. *TheoretiCS*, Volume 3, 2024. doi:10.46298/theoretics.24.21.

2 Ibrahim Assem, Daniel Simson, and Andrzej Skowroński. *Elements of the Representation Theory of Associative Algebras: Volume 1: Techniques of Representation Theory*, volume 65 of *London Mathematical Society Student Texts*. Cambridge University Press, 2006.

3 Ulrich Bauer, Fabian Lenzen, and Michael Lesnick. Efficient two-parameter persistence computation via cohomology. In *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *LIPIcs*, pages 15:1–15:17, 2023. doi:10.4230/LIPICS.SOCG.2023.15.

4 Matías Bender, Oliver Gäfvert, and Michael Lesnick. Computing minimal presentations of multi-parameter persistent homology. https://www.tugraz.at/projekte/cpw/abstracts.

5 Magnus Bakke Botnan and Michael Lesnick. An introduction to multiparameter persistence. In *Representations of Algebras and Related Structures*, EMS Series of Congress Reports, pages 77–150. EMS Press, 2023. doi:10.4171/ECR/19/4.

6 Adam Brown and Ondřej Draganov. Discrete microlocal Morse theory. *Journal of Pure and Applied Algebra*, 229(10):108068, 2025. doi:10.1016/j.jpaa.2025.108068.

7 Gunnar Carlsson and Vin De Silva. Zigzag persistence. *Foundations of computational mathematics*, 10:367–405, 2010. doi:10.1007/s10208-010-9066-0.

8 Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. In *Proceedings 23rd Annual Symposium on Computational Geometry*, pages 184–193, 2007. doi:10.1145/1247069.1247105.

9 Wojciech Chachólski, Martina Scolamiero, and Francesco Vaccarino. Combinatorial presentation of multidimensional persistent homology. *Journal of Pure and Applied Algebra*, 221(5):1055–1075, 2017. doi:10.1016/j.jpaa.2016.09.001.

10 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009. URL: https://books.google.at/books?id=i-bUBQAAQBAJ.

11 Tamal K Dey, Fengtao Fan, and Yusu Wang. Computing topological persistence for simplicial maps. In *Proceedings 30th Annual Symposium on Computational Geometry*, pages 345–354, 2014. doi:10.1145/2582112.2582165.

12 Tamal K. Dey and Tao Hou. Fast computation of zigzag persistence. In *Proceedings 30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *LIPIcs*, pages 43:1–43:15, 2022. doi:10.4230/LIPIcs.ESA.2022.43.

13 Tamal K. Dey, Florian Russold, and Shreyas N. Samaga. Efficient algorithms for complexes of persistence modules with applications. In *Proceedings 40th International Symposium on Computational Geometry (SoCG 2024)*, volume 293 of *LIPIcs*, pages 51:1–51:18, 2024. doi:10.4230/LIPIcs.SoCG.2024.51.

14 Tamal K. Dey and Shreyas N. Samaga. Quasi zigzag persistence: A topological framework for analyzing time-varying data. *Spotlight paper in TAG-DS*, 2025. arXiv:2502.16049.

15 Tamal K. Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022. URL: https://books.google.at/books?id=PWtYEAAAQBAJ.

16 H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010. URL: https://books.google.at/books?id=MDXa6gFRZuIC.

17 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002. doi:10.1007/s00454-002-2885-2.

18 Martina Flammer. Spatiotemporal persistence landscapes. *Young Researchers Forum (CG:YRF) 2024, Booklet of Abstracts*, 2024. URL: https://socg24.athenarc.gr/proceedings/YRF24%20booklet.pdf.

19 Ulderico Fugacci and Michael Kerber. Chunk reduction for multi-parameter persistent homology. In *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *LIPIcs*, pages 37:1–37:14, 2019. doi:10.4230/LIPICS.SOCG.2019.37.

20 Barbara Giunti, Jānis Lazovskis, and Bastian Rieck. DONUT: Database of Original & Non-Theoretical Uses of Topology, 2022. https://donut.topology.rocks.

21 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, 2002.

22 Michael Kerber and Alexander Rolle. Fast minimal presentations of bi-graded persistence modules. In *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX*, pages 207–220. SIAM, 2021. doi:10.1137/1.9781611976472.16.

23 Michael Kerber and Hannah Schreiber. Barcodes of towers and a streaming algorithm for persistent homology. *Discrete & Computational Geometry*, 61:852–879, 2019. doi:10.1007/s00454-018-0030-0.

24 Woojin Kim and Facundo Mémoli. Generalized persistence diagrams for persistence modules over posets. *Journal of Applied and Computational Topology*, 5(4):533–581, 2021. doi:10.1007/s41468-021-00075-1.

25 Michael Lesnick and Matthew Wright. Computing minimal presentations and bigraded Betti numbers of 2-parameter persistent homology. *SIAM Journal on Applied Algebra and Geometry*, 6(2):267–298, 2022. doi:10.1137/20M1388425.

26 Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 1998.

27 Ezra Miller. Homological algebra of modules over posets. *SIAM Journal on Applied Algebra and Geometry*, 9(3):483–524, 2025. doi:10.1137/22M1516361.

28 Dmitriy Morozov and Luis Scoccola. Computing Betti Tables and Minimal Presentations of Zero-Dimensional Persistent Homology. In *41st International Symposium on Computational Geometry (SoCG 2025)*, volume 332 of *LIPIcs*, pages 69:1–69:15, 2025. doi:10.4230/LIPIcs.SoCG.2025.69.

29 Andreas Nickel. Non-commutative Fitting invariants and annihilation of class groups. *Journal of Algebra*, 323(10):2756–2778, 2010.

30 Jeremy Rickard. Morita theory for derived categories. *Journal of the London Mathematical Society*, s2-39(3):436–456, 1989. URL: `https://londmathsoc.onlinelibrary.wiley.com/doi/10.1112/jlms/s2-39.3.436`, `doi:10.1112/jlms/s2-39.3.436`.

31 Baptiste Rognerud. The bounded derived categories of the Tamari lattices are fractionally Calabi-Yau. *Advances in Mathematics*, 389:107885, 2021. `doi:10.1016/j.aim.2021.107885`.

32 R. Schiffler. *Quiver Representations*. CMS Books in Mathematics. Springer International Publishing, 2014. URL: `https://books.google.at/books?id=0ctuBAAAQBAJ`.

33 Charles A. Weibel. *An Introduction to Homological Algebra*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1994.

34 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. In *Proceedings 20th Annual Symposium on Computational Geometry*, pages 347–356, 2004. `doi:10.1145/997817.997870`.

A Missing proofs from Section 2

Proof of Proposition 7. For the first claim, let $\mathrm{Proj}[x] \xrightarrow{\phi} \mathrm{Proj}[y]$ be a morphism of P-persistence modules.

$$\begin{array}{ccc} \mathrm{Proj}[x](x) & \xrightarrow{\phi(x)} & \mathrm{Proj}[y](x) \\ {\scriptstyle \mathrm{Proj}[x](x\leq z)}\downarrow & & \downarrow{\scriptstyle \mathrm{Proj}[y](x\leq z)} \\ \mathrm{Proj}[x](z) & \xrightarrow{\phi(z)} & \mathrm{Proj}[y](z) \end{array} \tag{20}$$

By Definition 6, if $\phi(x)$ in (20) is non-zero, then $\mathrm{Proj}[y](x) \cong \mathbb{F}_2$ and, thus, $y \leq x$. Moreover, by commutativity, we get that $\phi(z)$ is non-zero for all $z \geq x$. Conversely, if $\phi(x)$ is zero, then, again by commutativity, $\phi(z)$ has to be zero for all $z \geq x$. Hence, the morphism ϕ is completely determined by $\phi(x)$ and $\phi(x)$ can only be non-zero if $y \leq x$.

The second claim is a direct consequence of the first claim and the fact that in abelian categories Hom commutes with finite direct sums in both arguments ([26]). ◀

Proof of Proposition 10. Writing

$$AX = A(X_1, \dots, X_m) = (AX_1, \dots, AX_m) = (B_1, \dots, B_m) = B,$$

the problem reduces to solving m constrained linear systems of equations of the form $AX_j = B_j$ and $X_{kj} = 0$ if $z_k \not\leq y_j$. For each $1 \leq j \leq m$, after reordering, we can write $A = \begin{pmatrix} A^{\leq y_j} & A^{\not\leq y_j} \end{pmatrix}$ and $X_j = \begin{pmatrix} X_j^{\leq y_j} \\ X_j^{\not\leq y_j} \end{pmatrix}$. Then the system becomes

$$AX_j = \begin{pmatrix} A^{\leq y_j} & A^{\not\leq y_j} \end{pmatrix} \begin{pmatrix} X_j^{\leq y_j} \\ X_j^{\not\leq y_j} \end{pmatrix} = A^{\leq y_j} X_j^{\leq y_j} + A^{\not\leq y_j} X_j^{\not\leq y_j} = B_j \quad \text{and} \quad X_j^{\not\leq y_j} = 0.$$

Therefore, we can equivalently solve the unconstrained system $A^{\leq y_j} X_j^{\leq y_j} = B_j$ for all $1 \leq j \leq m$. ◀

Proof of Proposition 16. Let $Q_\bullet \to M$ be a projective resolution such that $q_i \colon Q_i \to \ker(q_{i-1})$ is a projective cover for all $i \geq 0$. By Corollary 5.10 in [2], for a P-persistence module, which corresponds to a module over the finite-dimensional incidence algebra of P, such a projective resolution always exists and is unique up to isomorphism. Moreover, by Proposition 2.1 in [29], it is a direct summand of every projective resolution of M. Thus,

if $U_\bullet \to M$ is any projective resolution, then $U_\bullet \cong Q_\bullet \oplus W_\bullet$ for some complex $W_\bullet$. In particular, for each $i \geq 0$, we have $U_i \cong Q_i \oplus W_i$. Since projective modules decompose into finite direct sums of indecomposable projectives, it follows that the multiplicity of each indecomposable projective in Q_i is bounded above by its multiplicity in U_i. Hence, Q_i has minimal indecomposable-projective multiplicities among all projective resolutions of M. Conversely, if $U_\bullet$ has minimal indecomposable-projective multiplicities in each degree, then from $U_i \cong Q_i \oplus W_i$ we get $W_i = 0$ for all $i \geq 0$. Therefore, $U_\bullet \cong Q_\bullet$.

Any projective presentation can be extended to a projective resolution and, by Corollary 5.10 in [2], a projective presentation satisfying the projective cover condition can be extended to a projective resolution satisfying the projective cover condition. Therefore, the result for presentations directly follows from the above argument. ◀

B Missing proofs from Section 3

The following proofs are based on diagram chasing. Note that this is legitimate as persistence modules can be viewed as actual modules over the incidence algebra of the poset.

Proof of Proposition 20. Suppose $x \in \ker(\partial_\ell)$. Since α_ℓ is an epimorphism, there exists $y \in G_\ell$ such that $\alpha_\ell(y) = x$. By commutativity, we get $\alpha_{\ell\text{-}1} \circ f^0_\ell(y) = \partial_\ell \circ \alpha_\ell(y) = 0$ and by exactness of the rows $f^0_\ell(y) \in \ker(\alpha_{\ell\text{-}1}) = \operatorname{im}(p^1_{\ell\text{-}1})$. Hence, there is a $z \in R_{\ell\text{-}1}$ such that $f^0_\ell(y) = p^1_{\ell\text{-}1}(z)$. Phrased differently, $(y, z) \in \ker \begin{pmatrix} f^0_\ell & p^1_{\ell\text{-}1} \end{pmatrix}$ and $\begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}(y, z) = \alpha_\ell(y) = x$. Hence, $\begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}$ is an epimorphism.

We have $\begin{pmatrix} \alpha_\ell & 0 \end{pmatrix} \circ \begin{pmatrix} p^1_\ell & 0 \\ f^1_\ell & p^2_{\ell\text{-}1} \end{pmatrix} = \begin{pmatrix} \alpha_\ell \circ p^1_\ell & 0 \end{pmatrix} = 0$. So $\operatorname{im} \begin{pmatrix} p^1_\ell & 0 \\ f^1_\ell & p^2_{\ell\text{-}1} \end{pmatrix} \subseteq \ker \begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}$.

Let $(x, y) \in \ker \begin{pmatrix} f^0_\ell & p^1_{\ell\text{-}1} \end{pmatrix}$ such that $\begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}(x, y) = \alpha_\ell(x) = 0$. Then $x \in \ker(\alpha_\ell) = \operatorname{im}(p^1_\ell)$ and there exists $a \in R_\ell$ such that $p^1_\ell(a) = x$. Moreover, we have

$$\begin{aligned} \begin{pmatrix} f^0_\ell & p^1_{\ell\text{-}1} \end{pmatrix}(x, y) &= f^0_\ell(x) + p^1_{\ell\text{-}1}(y) = f^0_\ell \circ p^1_\ell(a) + p^1_{\ell\text{-}1}(y) \\ &= p^1_{\ell\text{-}1} \circ f^1_\ell(a) + p^1_{\ell\text{-}1}(y) = p^1_{\ell\text{-}1}\big(f^1_\ell(a) + y\big) = 0 \end{aligned}$$

Hence, $f^1_\ell(a) + y \in \ker(p^1_{\ell\text{-}1}) = \operatorname{im}(p^2_{\ell\text{-}1})$ and there exists $b \in RR_{\ell\text{-}1}$ such that $y = f^1_\ell(a) + p^2_{\ell\text{-}1}(b)$. We conclude that $\begin{pmatrix} p^1_\ell & 0 \\ f^1_\ell & p^2_{\ell\text{-}1} \end{pmatrix}(a, b) = (x, y)$ and $\operatorname{im} \begin{pmatrix} p^1_\ell & 0 \\ f^1_\ell & p^2_{\ell\text{-}1} \end{pmatrix} = \ker \begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}$. ◀

Proof of Proposition 21. In Proposition 20, we proved that $\begin{pmatrix} \alpha_\ell & 0 \end{pmatrix} : \ker \begin{pmatrix} f^0_\ell & p^1_{\ell\text{-}1} \end{pmatrix} \to \ker(\partial_\ell)$ is an epimorphism. Hence, $\pi \circ \begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}$ is an epimorphism. Since

$$\begin{aligned} \pi \circ \begin{pmatrix} \alpha_\ell & 0 \end{pmatrix} \circ \begin{pmatrix} f^0_{\ell+1} & p^1_\ell & 0 \\ \vartheta_{\ell+1} & f^1_\ell & p^2_{\ell\text{-}1} \end{pmatrix} &= \pi \circ \begin{pmatrix} \alpha_\ell \circ f^0_{\ell+1} & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \pi \circ \partial_{\ell+1} \circ \alpha_{\ell+1} & 0 & 0 \end{pmatrix} = 0 \end{aligned}$$

we have $\operatorname{im} \begin{pmatrix} f^0_{\ell+1} & p^1_\ell & 0 \\ \vartheta_{\ell+1} & f^1_\ell & p^2_{\ell\text{-}1} \end{pmatrix} \subseteq \ker\big(\pi \circ \begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}\big)$.

Suppose $(x, y) \in \ker \begin{pmatrix} f^0_\ell & p^1_{\ell\text{-}1} \end{pmatrix}$ such that $\pi \circ \begin{pmatrix} \alpha_\ell & 0 \end{pmatrix}(x, y) = \pi\big(\alpha_\ell(x)\big) = 0$. Then there exists $z \in C_{\ell+1}$ such that $\partial_{\ell+1}(z) = \alpha_\ell(x)$. Moreover, since $\alpha_{\ell+1}$ is an epimorphism, there exists $a \in G_{\ell+1}$ such that $\alpha_{\ell+1}(a) = z$. Using commutativity, we obtain $\alpha_\ell \circ f^0_{\ell+1}(a) =$

$\partial_{\ell+1} \circ \alpha_{\ell+1}(a) = \alpha_\ell(x)$. This implies $\alpha_\ell\big(f^0_{\ell+1}(a) + x\big) = 0$, so $f^0_{\ell+1}(a) + x \in \ker \alpha_\ell = \operatorname{im}(p^1_\ell)$ and there exists $b \in R_\ell$ such that $x = f^0_{\ell+1}(a) + p^1_\ell(b)$. Since

$$\begin{aligned}\begin{pmatrix} f^0_\ell & p^1_{\ell\text{-}1}\end{pmatrix}(x,y) &= f^0_\ell(x) + p^1_{\ell\text{-}1}(y)\\ &= f^0_\ell\big(f^0_{\ell+1}(a) + p^1_\ell(b)\big) + p^1_{\ell\text{-}1}(y)\\ &= f^0_\ell \circ f^0_{\ell+1}(a) + f^0_\ell \circ p^1_\ell(b) + p^1_{\ell\text{-}1}(y)\\ &= p^1_{\ell\text{-}1} \circ \vartheta_{\ell+1}(a) + p^1_{\ell\text{-}1} \circ f^1_\ell(b) + p^1_{\ell\text{-}1}(y)\\ &= p^1_{\ell\text{-}1}\big(\vartheta_{\ell+1}(a) + f^1_\ell(b) + y\big) = 0\end{aligned}$$

we obtain $\vartheta_{\ell+1}(a) + f^1_\ell(b) + y \in \ker(p^1_{\ell\text{-}1}) = \operatorname{im}(p^2_{\ell\text{-}1})$ and there exists $c \in RR_{\ell\text{-}1}$ such that $y = \vartheta_{\ell+1}(a) + f^1_\ell(b) + p^2_{\ell\text{-}1}(c)$. We conclude that $\begin{pmatrix} f^0_{\ell+1} & p^1_\ell & 0\\ \vartheta_{\ell+1} & f^1_\ell & p^2_{\ell\text{-}1}\end{pmatrix}(a,b,c) = (x,y)$ and $\operatorname{im}\begin{pmatrix} f^0_{\ell+1} & p^1_\ell & 0\\ \vartheta_{\ell+1} & f^1_\ell & p^2_{\ell\text{-}1}\end{pmatrix} = \ker\big(\pi \circ \begin{pmatrix}\alpha_\ell & 0\end{pmatrix}\big)$. ◀

C TreeSolve algorithm

In this section, we discuss an algorithm TreeSolve which efficiently solves linear systems of equations where the coefficient matrix has the structure of the boundary matrix of a tree. We will show that TreeSolve solves such a linear system in linear time. It is based on the following definition of a leaf-order, which allows for an iterative substitution of the solution for edges connected to leaves in subgraphs. We call a vertex v in a tree T a leaf if it is incident to at most one edge e, denoted by $v < e$. Note that every leaf in a tree with more than one vertex has degree one.

▶ **Definition 55** (Leaf-order). *Let T be a tree graph on n vertices. We call an order $v_1, \dots, v_n$ of the vertices of T a leaf-order if v_i is a leaf of $T \setminus \{v_1, \dots, v_{i\text{-}1}\}$ for all $1 \le i \le n$.*

A leaf-order can be found by the following routine.

Algorithm LeafOrder ($T = (V, E)$)

1. Initialize: $\deg\colon V \to \mathbb{N}_0$, with $\deg(v) = 0$ for all $v \in V$, a queue $Q = \emptyset$, and a list $L = \emptyset$;
2. For all $e = (u, v) \in E$, set $\deg(u) = \deg(u) + 1$ and $\deg(v) = \deg(v) + 1$;
3. For all $v \in V$, if $\deg(v) \le 1$ add v to Q;
4. While $Q \neq \emptyset$ do
   a. Take v from Q and set $\deg(v) = 0$;
   b. Add v to L;
   c. Set $\deg(u) = \max\big(\deg(u) - 1, 0\big)$ for all u adjacent to v in T;
   d. If $\deg(u) = 1$, for u adjacent to v, add u to Q

▶ **Proposition 56.** *For a tree $T = (V, E)$ on n vertices, the algorithm* LeafOrder *computes a leaf-order L in $O(n)$ time.*

Proof. Step 2 of the algorithm obviously computes the degree $\deg(v)$ of v in the graph T for all $v \in V$. After Step 3 of the algorithm Q contains all leaves of T. Since every non-empty tree has a leaf, $Q \neq \emptyset$. At this point, either T was a single vertex and Q contains this single vertex of degree zero, or $\deg(v) = 1$ for all $v \in Q$ and, since T is connected, $\deg(v) > 1$ for all $v \in V \setminus Q$. The case where T is a single vertex is obvious. From now on assume that T has at least two vertices. Let $v_1, \dots, v_n$ be the order stored in the list L. We now show by induction on the iterations of step 4 that, before each iteration of step 4, Q contains all leaves

of $T \setminus \{v_1, \ldots, v_{i-1}\}$ and $\deg(u)$ equals the degree of the vertex u in $T \setminus \{v_1, \ldots, v_{i-1}\}$. Before the first iteration of step 4 this is true as argued above. Now suppose this is true before the i-th iteration of step 4. In iteration i, we remove the leaf v_i of $T \setminus \{v_1, \ldots, v_{i-1}\}$ from Q, add it to L, set $\deg(v_i) = 0$, and decrease the degree of every neighbor of v_i in $T \setminus \{v_1, \ldots, v_{i-1}\}$ by one. Thus, $\deg(u)$ now equals the degree of every vertex u in $T \setminus \{v_1, \ldots, v_i\}$. Let u be a leaf of $T \setminus \{v_1, \ldots, v_i\}$. If u was not a leaf in $T \setminus \{v_1, \ldots, v_{i-1}\}$, then its degree has been reduced to one in iteration i and u was added to Q. Otherwise, u was already a leaf in $T \setminus \{v_1, \ldots, v_{i-1}\}$ and, thus, by assumption, it was already in Q. This proves the claim. Since we add a leaf of $T \setminus \{v_1, \ldots, v_{i-1}\}$ to L in each iteration L is a leaf-order of T.

To initialize and compute deg of T, we need to process all vertices and edges of T once. Since a tree has $n - 1$ edges, this is clearly $O(n)$. In the third step, we process all vertices which is again $O(n)$. Since each vertex is added and removed from Q exactly once there are n iterations in step 4. If $\Gamma(v)$ denotes the set of neighbors of v, then the update of neighbor degrees is $O(\sum_{v \in V} |\Gamma(v)|) = O(|E|) = O(n)$. Hence, the overall worst case complexity is $O(n)$. ◀

The following algorithm TreeSolve solves a linear system of equations $Ax = b$, where A is the boundary matrix of a tree T, in linear time by simply substituting the solution using a leaf-order. Each column of A and row of x corresponds to an edge in T and each row of A and b corresponds to a vertex in T. The algorithm takes as input the tree $T = (V, E)$ and the right-hand side b as a map $b\colon V \to \mathbb{F}_2$ representing the entries of the rows of b.

Algorithm TreeSolve $(T = (V, E), b\colon V \to \mathbb{F}_2)$
1. Call LeafOrder(T) to compute a leaf-order $v_1, \ldots, v_n$ of T;
2. Initialize $x\colon E \to \{\bot\} \cup \mathbb{F}_2$ by $x(e) = \bot$ for all $e \in E$;
3. For $i \coloneqq 1$ to $n - 1$ do
   a. Find $e \in E$ such that $v_i < e$ and $x(e) = \bot$;
   b. Set $x(e) = b(v_i) + \sum_{v_i < e' \wedge x(e') \neq \bot} x(e')$ (in $\mathbb{F}_2$)

▶ **Theorem 57.** *Let $Ax = b$ be a system of linear equations such that $b \in \mathrm{im}(A)$ and A is the boundary matrix of a tree T on n vertices. The algorithm* TreeSolve *computes a solution of $Ax = b$ in $O(n)$ time.*

Proof. Let $v_1, \ldots, v_n$ be the leaf-order computed in step 1. By Definition 55 and Proposition 56, v_i is a leaf of the tree $T_i \coloneqq T \setminus \{v_1, \ldots, v_{i-1}\}$, obtained by iteratively removing the leaves $v_1, \ldots, v_{i-1}$ from T. For every $1 \leq i \leq n - 1$, the leaf v_i is incident to a unique edge e_i in T_i and removing the leaf v_i from T_i removes exactly the one edge e_i. In step 2, we set $x(e) \coloneqq \bot$ for all $e \in T$ meaning that the values on all edges are undetermined. Let $A[v, :]$ denote the row of A corresponding to the vertex $v \in T$. Then, viewing x as a column vector, we get $A[v, :]x = \sum_{v < e \in E} x(e)$.

We now show by induction on i that after each iteration of step 3, the set of edges $E_{i+1} \coloneqq E(T_{i+1}) = \{e \in E(T) | x(e) = \bot\}$ and $A[v_i, :]x = b(v_i)$. For $i = 1$, the vertex v_1 is a leaf in T, there is exactly one edge e_1 incident to v_1 in T and $x(e) = \bot$ for all $e \in E$. The algorithm picks e_1 and sets $x(e_1) = b(v_1)$. Thus, we get $A[v_1, :]x = b(v_1)$ and $x(e_1) \neq \bot$. Since T_2 is obtained from $T = T_1$ by removing the vertex v_1 and the edge e_1, we also get $E_2 = \{e \in E(T) | x(e) = \bot\}$. Suppose the statement is true for all $i = 1, \ldots, k - 1$. By assumption, $E_k = \{e \in E(T) | x(e) = \bot\}$ before iteration k. Therefore, in iteration k, the algorithm picks the unique edge e_k incident to the leaf v_k in T_k. Every other edge e' incident to v_k in T is not in T_k anymore and thus, $x(e') \neq \bot$. In step 3*b*, we set $x(e_k) = b(v_k) + \sum_{v_k < e' \wedge x(e') \neq \bot} x(e')$.

Thus, after step 3b, we get $A[v_k,:]x = \sum_{v_k<e'\in E} x(e') = \sum_{v_k<e'\wedge x(e')\neq\perp} x(e') = b(v_k)$. Moreover, $x(e_k) \neq \perp$ after step 3b implies $E_{k+1} = \{e \in E(T) | x(e) = \perp\}$.

We conclude that $A[v_i,:]x = b(v_i)$ for all $1 \leq i \leq n-1$. We now show that the final equation $A[v_n,:]x = b(v_n)$ is also satisfied. Let $\overline{1} = \begin{pmatrix}1 & \cdots & 1\end{pmatrix}$ and $\overline{0} = \begin{pmatrix}0 & \cdots & 0\end{pmatrix}$. Since A has exactly two non-zero entries per column, we get $\overline{1}A = \sum_{i=1}^{n} A[v_i,:] = \overline{0}$. By assumption $b \in \operatorname{im}(A)$, thus, there exists a y such that $Ay = b$, which implies $\overline{1}Ay = \overline{0}y = 0 = \overline{1}b$. In particular, $\sum_{i=1}^{n-1} b(v_i) = b(v_n)$. Therefore, we obtain $\overline{1}Ax = \sum_{i=1}^{n-1} A[v_i,:]x + A[v_n,:]x = \sum_{i=1}^{n-1} b(v_i) + A[v_n,:]x = b(v_n) + A[v_n,:]x = 0$ and also the last equation is satisfied.

By Proposition 56, step 1 takes $O(n)$ time. Step 2 obviously takes $O(n)$ time. Using an incidence list, checking the incident edges of all vertices and summing over all incident edges of all vertices is $O(\sum_{v\in V} |\Gamma(v)|) = O(|E|) = O(n)$. Therefore, in total, the worst-case complexity is $O(n)$. ◀

D Computing presentations of homology from PiReps

In this section, we discuss how to compute a presentation of the homology of a PiRep as in Definition 18. We consider a chain complex segment

$$Q_{-1} \xleftarrow{q_0} Q_0 \xleftarrow{q_1} Q_1 \tag{21}$$

where the Q_i are projective and the maps q_i are represented by P-graded matrices. We start by extending $Q_{-1} \xleftarrow{q_0} Q_0$ to an exact sequence of projective modules:

$$Q_{-1} \xleftarrow{q_0} Q_0 \xleftarrow{u_0} U_0 \xleftarrow{u_1} U_1. \tag{22}$$

This can be done, for example, by using the algorithm[2] MakeExact in [6]. Since, by exactness $\ker(q_0) = \operatorname{im}(u_0)$, u_1 is a presentation of $\ker(q_0)$. Since $\operatorname{im}(q_1) \subseteq \ker(q_0)$ we obtain the following diagram:

$$\begin{array}{ccccccc} & & & & Q_1 & & \\ & & & & \downarrow{\scriptstyle q_1} \quad \searrow{\scriptstyle s} & & \\ 0 & \longleftarrow & \ker(q_0) & \xleftarrow{u_0} & U_0 & \xleftarrow{u_1} & U_1 \end{array} \tag{23}$$

where u_0 is an epimorphism. Therefore, by the lifting property of projective modules, we get a lift s of q_1 along u_0. This lift can be computed by solving a P-graded linear system, as in Definition 9, using Proposition 10. We can interpret U_0 and U_1 as the generators and relations of the kernel and Q_1 as the initial relations on Q_0. We now have all the ingredients for a presentation of the homology. Let $\pi\colon \ker(q_0) \to \ker(q_0)/\operatorname{im}(q_1)$ denote the projection to the cokernel.

▶ **Theorem 58.** *The following exact sequence:*

$$0 \longleftarrow \ker(q_0)/\operatorname{im}(q_1) \xleftarrow{\pi \circ u_0} U_0 \xleftarrow{\begin{pmatrix}u_1 & s\end{pmatrix}} U_1 \oplus Q_1 \tag{24}$$

is a presentation of the homology of (21).

[2] The algorithm MakeExact in [6] is formulated in terms of injective resolutions but in our setting the projective case is completely dual.

Proof. Since u_0 and π are epimorphisms, $\pi \circ u_0$ is also an epimorphism.

Since $\pi \circ u_0 \circ \begin{pmatrix} u_1 & s \end{pmatrix} = \begin{pmatrix} \pi \circ u_0 \circ u_1 & \pi \circ u_0 \circ s \end{pmatrix} = \begin{pmatrix} 0 & \pi \circ q_1 \end{pmatrix} = 0$, we have $\operatorname{im} \begin{pmatrix} u_1 & s \end{pmatrix} \subseteq \ker(\pi \circ u_0)$.

Let $x \in \ker(\pi \circ u_0)$. Then $\pi \circ u_0(x) = 0$ implies that there exists $y \in Q_1$ such that $u_0(x) = q_1(y) = u_0 \circ s(y)$. Since $u_0\big(x + s(y)\big) = 0$ and $x + s(y) \in \ker(u_0) = \operatorname{im}(u_1)$, there exists $z \in U_1$ such that $x = u_1(z) + s(y)$. We conclude that $\begin{pmatrix} u_1 & s \end{pmatrix}(z, y) = x$ and $\operatorname{im} \begin{pmatrix} u_1 & s \end{pmatrix} = \ker(\pi \circ u_0)$. ◀

The presentation of Theorem 58 is not necessarily minimal. In [6], the authors also discuss methods to minimize resolutions that can be used to minimize the presentation (24).