Computer Aided Geometric Design 73 (2019) 1–15

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Holes and dependences in an ordered complex ☆

Herbert Edelsbrunner, Katharina Ölsböck *

IST Austria (Institute of Science and Technology Austria), Klosterneuburg, Austria

ARTICLE INFO

Article history:
Available online 20 June 2019

Keywords:
Computational topology
Shape reconstruction
Persistence diagrams
Bases in homology and cohomology
Alpha shapes
Computational experiments

ABSTRACT

We use the canonical bases produced by the tri-partition algorithm in (Edelsbrunner and Ölsböck, 2018) to open and close holes in a polyhedral complex, K. In a concrete application, we consider the Delaunay mosaic of a finite set, we let K be an Alpha complex, and we use the persistence diagram of the distance function to guide the hole opening and closing operations. The dependences between the holes define a partial order on the cells in K that characterizes what can and what cannot be constructed using the operations. The relations in this partial order reveal structural information about the underlying filtration of complexes beyond what is expressed by the persistence diagram.

1. Introduction

This paper studies mechanisms for opening and closing holes in polyhedral complexes. Our primary motivation is the modeling of biomolecules with Alpha shapes, but the methods are more generally applicable. As an illustrative example, we imagine a cell membrane protein with a functional channel for ion transport. We may need a geometric model that represents the channel as a tunnel, but in the Alpha shape of the appropriate scale, the channel may be closed, or there may be tunnels that have nothing to do with the channel or even interfere with the channel, which is worse. We study ways to open and close tunnels, or more generally holes of any dimension. These operations may be triggered interactively or may be controlled automatically by the persistence diagram of the distance function defined by the protein. In other words, we explore the shape reconstruction question in which the holes take front seat in the decision process.

In a broader context, the work in this paper is related to the geometric modeling of biomolecules, the study of cavities in materials, the reconstruction of shapes from point cloud data, and the characterization of shape with persistent homology:

- We refer to (Leach, 2001) for a general introduction to the modeling of molecules, and to (Edelsbrunner and Mücke, 1994) for the original paper on 3-dimensional Alpha shapes, which provide a versatile geometric representation that facilitates the detailed analysis of biomolecules.
- Work on the detection and visualization of cavities in molecules is surveyed in (Simões et al., 2017). Here we mention (Edelsbrunner et al., 1995) for the introduction of the concept of pockets, and (Lee et al., 2017) for organizing synthetic materials in terms of their tunnel systems.

☆ This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 78818 Alpha). It is also partially supported by the DFG Collaborative Research Center TRR 109, 'Discretization in Geometry and Dynamics', through grant no. I02979-N35 of the Austrian Science Fund (FWF).

* Corresponding author.
E-mail addresses: edels@ist.ac.at (H. Edelsbrunner), katharina.oelsboeck@ist.ac.at (K. Ölsböck).

https://doi.org/10.1016/j.cagd.2019.06.003
0167-8396/

- The reconstruction of shapes and images from point cloud data has broad industrial applications. We mention the Wrap algorithm (Edelsbrunner, 2003) for surface reconstruction, to which our work applies. The corresponding filtrations are however beyond the scope of this paper, which focuses on Alpha complexes. In this context, we also mention (Kurlin, 2016), which uses persistent homology to reconstruct cycles for image segmentation purposes.
- Persistent homology describes the multi-scale connectivity of a complex as defined by a monotonic ordering of its cells (Edelsbrunner and Harer, 2010). This paper makes use of both the theory and the algorithms of persistent homology to construct tri-partitions of the complex and corresponding bases in homology and cohomology.

The work reported in this paper is directly based on the combinatorial approach to Hodge theory reported in (Edelsbrunner and Ölsböck, 2018). Given a monotonic ordering of the cells of a polyhedral complex, it is proven that there is a unique tri-partition of the set of p-cells, for every dimension p, namely into a maximal p-tree, a maximal p-cotree, and a remainder whose cardinality is the rank of the p-th reduced homology group. Importantly, the tri-partition implies canonical bases in homology and cohomology, and all this information is readily computed by matrix reduction. Making use of these bases, our main contributions to the state-of-the-art are:

- a formulation of hole closing and opening operations that manipulate a complex based on the persistence diagram of its monotonic ordering;
- an expression of the dependences between the holes in terms of a partial order on the cells that characterizes what is and what is not computable with these operations;
- the decoration of the persistence diagram with structural information about the filtration that goes beyond the pairing of cells.

We have implemented the operations and provide results of preliminary computational experiments. These include statistics for computing the tri-partition and the canonical bases, illustrations of the various operations, and a visualization of the dependences between the holes. With reference to the study of protein structures, we mention one experiment in which the complex represents the structure of *Gramicidin*, which is a cell membrane protein whose channel is surrounded by the alpha helix structure of its backbone. Applying the unfill operation and disregarding side effects, we get a complex with rank 1 first homology group. Its tunnel represents the channel. Applying the unlock operation to this complex, the tunnel is opened by cutting the relatively weaker bonds holding together adjacent rounds of the helix. Again disregarding side effects, this leaves behind a complex with rank 0 first homology group that displays the helix structure of the protein. We mention that this outcome is obtained with just one hole manipulating operation.

Outline. Section 2 provides the background, including the tri-partition of a polyhedral complex and the corresponding canonical bases in homology and cohomology. Section 3 studies the dependences between the holes, which Section 4 uses to implement the recursive hole manipulating operations. Section 5 illustrates the concepts with two case studies. Section 6 quantifies the algorithms with statistics for random points in three dimensions. Section 7 concludes this paper.

2. Background

We will make frequent use of homology and cohomology groups; see (Hatcher, 2002; Munkres, 1984) for general background on these topics. For pragmatic reasons, we use $\mathbb{Z}/2\mathbb{Z}$ coefficients so that cycles and cocycles can be treated as sets. We represent shapes using polyhedral complexes, which are easy to define and more general than simplicial complexes. The methods developed in this paper work for complexes that are more general than polyhedral complexes, but we make no attempt to determine the limit of applicability.

Polyhedral complexes. A *p-cell* is a p-dimensional convex polytope, σ, and we write $\dim\sigma = p$ for its *dimension*. Such a polytope is the convex hull of a finite set of points and therefore closed and bounded. A *supporting hyperplane* has non-empty intersection with the polytope and bounds a closed half-space that contains the polytope. A *face* of σ is the intersection with a supporting hyperplane; it is a convex polytope of dimension less than p. We call σ a *coface* of its faces. A *polyhedral complex*, K, is a collection of cells that is closed under taking faces such that the intersection of any two cells is a face of both. By convention, we require that the empty cell is part of K; its dimension is -1, and it is a face of every cell. A cell is *maximal* if it has no proper coface in K. The *dimension* of K is the maximum dimension of any of its cells. We write $K^{(p)}$ for the *p-skeleton*, which contains all cells of dimension at most p, and $K^p = K^{(p)} \setminus K^{(p-1)}$ for the set of p-cells.

To store a polyhedral complex in the computer, it is common to order the cells — arbitrarily or otherwise — and to encode the face relation in matrix form. Letting $\sigma_0, \sigma_1, \ldots, \sigma_m$ be the ordering, the *boundary matrix*, $\partial[0..m, 0..m]$, is defined by

$$\partial[i,j] = \begin{cases} 1 & \text{if } \sigma_i \subseteq \sigma_j \text{ and } \dim\sigma_i = \dim\sigma_j - 1, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In words, column j of ∂ stores the faces of σ_j whose dimension is $\dim\sigma_j - 1$, and row i stores the cofaces of σ_i whose dimension is $\dim\sigma_i + 1$. Throughout this paper, we use *monotonic orderings* in which every cell is preceded by its faces. The

boundary matrix of a monotonically ordered polyhedral complex is upper-triangular. A *filtration* of K is a nested sequence of subcomplexes that ends with K. For example, $K_0 \subseteq K_1 \subseteq \ldots \subseteq K_m$ with $K_\ell = \{\sigma_0, \sigma_1, \ldots, \sigma_\ell\}$ for every $0 \leq \ell \leq m$ is a filtration iff the ordering of the cells is monotonic.

Homology and cohomology. As mentioned above, we use modulo-2 arithmetic to define the homology and cohomology groups. A *p-chain* is a subset of K^p, and the *sum* of two p-chains is the symmetric difference between the two subsets. With this operation, the p-chains form a group, denoted C_p. The *boundary operator*, $\partial_p \colon \mathsf{C}_p \to \mathsf{C}_{p-1}$, maps every p-cell to the collection of its $(p-1)$-faces, and it maps every p-chain to the sum of the boundaries of its p-cells. A *p-cycle* is a p-chain with empty boundary, and the collection of p-cycles forms a group, denoted $\mathsf{Z}_p \subseteq \mathsf{C}_p$. A *p-boundary* is the boundary of a $(p+1)$-chain, and the collection of p-boundaries forms again a group, $\mathsf{B}_p \subseteq \mathsf{C}_p$. Since the boundary of every boundary is empty, we have $\mathsf{B}_p \subseteq \mathsf{Z}_p \subseteq \mathsf{C}_p$. Finally, the *p-th reduced homology group* is $\tilde{\mathsf{H}}_p = \mathsf{Z}_p/\mathsf{B}_p$, that is, the partition of Z_p in which two p-cycles are *equivalent* if they differ by a p-boundary. We recall that reduced homology arises because we have $\emptyset \in K$, by assumption, and its main difference to conventional homology is that the rank of $\tilde{\mathsf{H}}_0$ is the number of gaps between the components, while the rank of H_0 is the number of components, which exceeds the number of gaps by one. We write $\tilde{\beta}_p = \operatorname{rank} \tilde{\mathsf{H}}_p$, referring to it as the *p-th reduced Betti number* of K.

Cohomology is in many ways similar or, more accurately, symmetric or complementary. A *p-cochain* is just a p-chain, and the groups are also the same, $\mathsf{C}^p = \mathsf{C}_p$. The *coboundary operator*, $\delta_p \colon \mathsf{C}^p \to \mathsf{C}^{p+1}$, maps every p-cell to its collection of $(p+1)$-cofaces, and it maps every p-cochain to the sum of the coboundaries of its p-cells. A *p-cocycle* is a p-cochain with empty coboundary, a *p-coboundary* is the coboundary of a $(p-1)$-cochain, and the corresponding groups are $\mathsf{B}^p \subseteq \mathsf{Z}^p \subseteq \mathsf{C}^p$. The *p-th reduced cohomology group* is $\tilde{\mathsf{H}}^p = \mathsf{Z}^p/\mathsf{B}^p$, for every dimension p. We write $\tilde{\beta}^p = \operatorname{rank} \mathsf{H}^p$ for its rank.

Tri-partitions. Following (Kalai, 1983), we call a p-chain that contains no non-empty p-cycle a *p-tree*. To avoid complicated terminology, we do not insist that a p-tree be connected. A p-tree is *maximal* if no p-tree of the same complex properly contains it. For example, every spanning tree of a connected graph is a maximal 1-tree, and every spanning forest of a not necessarily connected graph is a 1-tree. Similarly, we call a p-cochain that contains no non-empty p-cocycle a *p-cotree*. A p-cotree is *maximal* if no p-cotree of the same complex properly contains it. Hodge theory implies the following generalization of the tri-partition of a graph embedded on a closed surface proved in (Rosenstiehl and Read, 1978). We state the result in the terminology of (Edelsbrunner and Ölsböck, 2018).

Proposition 1 *(Tri-partition). Let K be a polyhedral complex. Then there are tri-partitions $A_p \sqcup A^p \sqcup E_p = K^p$, for every dimension p, such that A_p is a maximal p-tree, A^p is a maximal p-cotree, and $|E_p| = \tilde{\beta}_p$.*

The p-cells in E_p form a compact representation of the p-th reduced homology of K: there are $\tilde{\beta}_p$ p-cycles that span $\tilde{\mathsf{H}}_p$ such that each of these cycles contains exactly one of the cells in E_p and each cell in E_p belongs to exactly one of these cycles. The tri-partition whose existence is asserted by Proposition 1 is generally not unique. However, given a monotonic ordering of the cells, we can use matrix reduction to construct a unique tri-partition. The cells in E_p are exactly the cells in the monotonic ordering that give birth to essential homology classes; those in A^p and A_p are the cells that give birth and death to non-essential homology classes, see (Edelsbrunner and Ölsböck, 2018).

Canonical bases. Rather than in the tri-partition itself, we are interested in the bases it defines. For example, for each $\sigma_j \in A^p \sqcup E_p$, there is a unique non-empty p-cycle $\mathsf{z}_p(j) \subseteq A_p \cup \{\sigma_j\}$, which we refer to as the *canonical p-cycle* of σ_j. In contrast, for each $\sigma_j \in A_p$, there is a unique sum of canonical $(p-1)$-cycles that bound in $K_j = \{\sigma_0, \sigma_1, \ldots, \sigma_j\}$ but not in K_{j-1}, and we refer to the unique p-chain $\mathsf{c}_p(j) \subseteq A_p$ whose boundary is this sum as the *canonical p-chain* of σ_j. Symmetrically, for each $\sigma_i \in A_p \sqcup E_p$, there is a unique non-empty p-cocycle $\mathsf{z}^p(i) \subseteq A^p \sqcup \{\sigma_i\}$, which we refer to as the *canonical p-cocycle* of σ_i. Furthermore, for each $\sigma_i \in A^p$, there is a unique sum of canonical $(p+1)$-cocycles that cobound in $K \setminus K_i$ but not in $K \setminus K_{i+1}$, and we refer to the unique p-cochain $\mathsf{c}^p(i) \subseteq A^p$ whose coboundary is this sum as the *canonical p-cochain* of σ_i. As proved in (Edelsbrunner and Ölsböck, 2018), these vectors form bases in homology and cohomology.

Proposition 2 *(Canonical bases). Assume a monotonic ordering of a polyhedral complex, K, and let $K^p = A_p \sqcup A^p \sqcup E_p$ be the corresponding tri-partition in dimension p. Then*

- *$\{\mathsf{z}_p(j) \mid \sigma_j \in A^p \sqcup E_p\}$ is a basis of Z_p.*
- *$\{\mathsf{z}_p(j) \mid \sigma_j \in E_p\}$ generates a basis of $\tilde{\mathsf{H}}_p$.*
- *$\{\partial \mathsf{c}_p(j) \mid \sigma_j \in A_p\}$ is a basis of B_{p-1}.*
- *$\{\mathsf{z}^p(i) \mid \sigma_i \in A_p \sqcup E_p\}$ is a basis of Z^p.*
- *$\{\mathsf{z}^p(i) \mid \sigma_i \in E_p\}$ generates a basis of $\tilde{\mathsf{H}}^p$.*
- *$\{\delta \mathsf{c}^p(i) \mid \sigma_i \in A^p\}$ is a basis of B^{p+1}.*

As explained in (Edelsbrunner and Ölsböck, 2018), the canonical cycles and chains can be computed by column reduction, and the canonical cocycles and cochains can be computed by row reduction of the boundary matrix.

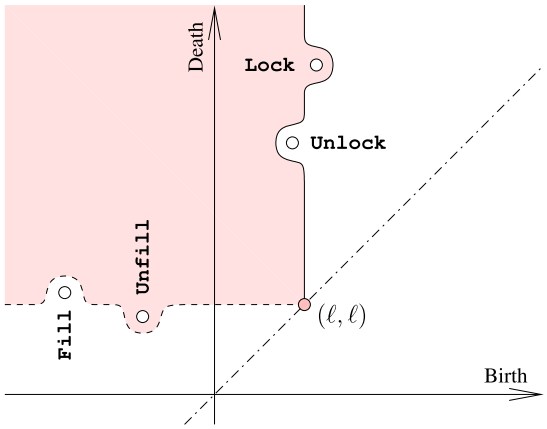


Fig. 1. A distorted upper left quadrant in the persistence diagram. The lock and unlock operations have the effect of advancing or delaying the birth, and the fill and unfill operations have the effect of advancing or delaying the death.

3. Dependence structure

Prior to introducing the operations that manipulate the hole system in a complex, we study the dependences between holes. Expressing them as ordered pairs over the cells, we obtain a partial order whose linear extensions describe the flexibility in manipulating the hole system without side effects.

Persistence in a filtration. Our method is best described by first introducing the persistence diagram of a monotonic ordering of the cells, $\sigma_0, \sigma_1, \ldots, \sigma_m$. We write $\iota_0 \colon 0 < 1 < \ldots < m$ for the corresponding ordering of the indices, and $K_\ell = \{\sigma_0, \sigma_1, \ldots, \sigma_\ell\}$ for the ℓ-th complex in the corresponding filtration. Two indices $i < j$ are the coordinates of a point in the *persistence diagram*, denoted $(i, j) \in \mathrm{Dgm}(\iota_0)$, if adding σ_i to K_{i-1} gives birth to a homology class in K_i, and adding σ_j to K_{j-1} gives death to this very class in K_j; see (Edelsbrunner and Harer, 2010) for the algebraic details of this definition. The *persistence* of this class is $j - i$, which is the vertical distance between the point and the diagonal. If $\dim \sigma_i = p$, then $\dim \sigma_j = p + 1$, and the mentioned homology class is generated by a p-cycle. It is also possible that σ_i gives birth to a class that stays alive throughout the remainder of the filtration. In this case, $(i, \infty) \in \mathrm{Dgm}(\iota_0)$, and the persistence of the corresponding *essential* homology class is ∞. Besides $\mathrm{Dgm}(\iota_0)$, which is a multiset of points, we introduce the *p-dimensional persistence diagram*, denoted $\mathrm{Dgm}_p(\iota_0)$, which is the sub-multiset of points that mark the birth and death of classes generated by p-cycles. Importantly, the Betti numbers of all complexes in the filtration can be recovered from these diagrams (Edelsbrunner and Harer, 2010, page 152).

Proposition 3 *(Quadrants and ranks). For every $0 \le \ell \le m$ and every dimension p, the p-th Betti number of K_ℓ is the number of points $(i, j) \in \mathrm{Dgm}_p(K)$ that satisfy $i \le \ell < j$.*

In words, we get the p-th Betti number of K_ℓ by counting the points of $\mathrm{Dgm}_p(K)$ in the upper left quadrant anchored at the point (ℓ, ℓ) on the diagonal. This quadrant is closed along its right and open along its lower border. We will interpret the opening and closing of holes in K_ℓ as distortions of the quadrant; see Fig. 1.

Reduction and dependence. To capture the dependences between the homology classes in a filtration, we will refer to details in the reduced matrix computed to derive the persistence diagram and the bases mentioned in the previous section. Let ∂ be the boundary matrix with rows and columns ordered according to ι_0, and let R be obtained from ∂ by applying a sequence of left-to-right column additions. For each $0 \le j \le m$, we write $\mathrm{low}(j)$ for the row of the lowest non-zero entry in column j, and we set $\mathrm{low}(j) = -\infty$ if the entire column is 0. We call R *left-to-right reduced* if $\mathrm{low}(j) \neq \mathrm{low}(k)$ for all non-zero columns $j \neq k$ in R. The *standard column reduction algorithm* in persistent homology (Edelsbrunner and Harer, 2010, Chapter VII) produces such a matrix. After initializing the matrices R, U, C to the boundary matrix, the identity matrix, and the zero matrix, the algorithm maintains $R = \partial U$ while reducing R and using C for book-keeping purposes:

```
1   R = ∂; U = Id; C = 0;
2   for j = 0 to m do
3     while ∃0 ≤ ℓ < j with low(ℓ) = low(j) > −∞ do
4       R[., j] = R[., j] + R[., ℓ];
5       U[., j] = U[., j] + U[., ℓ];
6       C[ℓ, j] = C[ℓ, j] + 1 (mod 2).
```

Line 4 performs a column addition, line 5 maintains $R = \partial U$, and line 6 records the operation for later reference. Instead of the standard method, we could also use the *exhaustive column reduction algorithm* (Edelsbrunner and Zomorodian, 2003), whose only difference is in the condition that controls the while loop, substituting "$R[\mathrm{low}(\ell), j] \neq 0$" for "$\mathrm{low}(\ell) = \mathrm{low}(j) > -\infty$". As suggested by the name, the exhaustive reduction algorithm keeps reducing column j even after $\mathrm{low}(j)$ has been established, which is when the standard reduction algorithm moves on to the next column. While the two algorithms may compute different matrices R, they are both reduced, the same columns are zero and non-zero, and the non-zero columns have their lowest non-zero entries in the same rows (Cohen-Steiner et al., 2006). Note that the pairs $(\mathrm{low}(j), j)$ for non-zero columns j correspond exactly to the points with finite coordinates in the persistence diagram. However, the canonical bases might differ for the two algorithms.

We fix the reduced matrix R and use it as a starting point to construct five maps that capture different types of dependences between the homology classes. The goal is to distinguish the ordered pairs that are necessary to keep R reduced from the others, which can be swapped without affecting R. Writing $[m] = \{0, 1, \ldots, m\}$, we introduce maps $\mathsf{X}\colon [m] \to 2^{[m]}$ with $\mathsf{X} \in \{\delta, \mathsf{BD}, \mathsf{DB}, \mathsf{BB}, \mathsf{DD}\}$, and to define these maps, we write (b_i, d_i) for the unique point in $\mathrm{Dgm}(\iota_0)$ for which either $b_i = i$ or $d_i = i$:

$$\delta(i) = \{j \mid \sigma_i \subseteq \sigma_j \text{ and } \dim \sigma_j = \dim \sigma_i + 1\}; \tag{2}$$

$$\mathsf{BD}(i) = \{j \mid (i, j) \in \mathrm{Dgm}(\iota_0)\}; \tag{3}$$

$$\mathsf{DB}(i) = \{j \mid i = d_i < j = b_j \text{ and } (C[i, j] = 1 \text{ or } R[i, d_j] = 1)\}; \tag{4}$$

$$\mathsf{BB}(i) = \{j \mid i = b_i < j = b_j \text{ and } R[i, d_j] = 1\}; \tag{5}$$

$$\mathsf{DD}(i) = \{j \mid i = d_i < j = d_j \text{ and } C[i, j] = 1\}. \tag{6}$$

Each maps the index of a cell to the indices of a subset of cells that appear later in the monotonic ordering, with the understanding that these ordered pairs ought to be maintained. The coboundary map, δ, maps a cell to its cofaces of one dimension higher. BD maps a cell that gives birth to a non-essential homology class to the cell that gives death to this class. The other three maps encode dependences between cells of the same dimension that result from the reduction process. They record each column addition ($C[i, j] = 1$), and each non-zero entry above the lowest in each non-zero column ($R[i, d_j] = 1$). We get the three maps by differentiating between birth- and death-giving cells in the domain and the codomain. In the reverse direction, we define $\mathsf{X}^T(j) = \{i \mid j \in \mathsf{X}(i)\}$ for each X. For example, the inverse of δ, $\delta^T = \partial$, maps a cell to its faces of one dimension lower. Note that each of the last three maps can be unambiguously drawn as a collection of arrows connecting points of the persistence diagram because the map specifies which coordinates of the endpoints the arrow connects.

Partial order. We summarize all five maps in a partial order, $\mathcal{P}$, which we refer to as the *dependence structure* of ι_0, or more precisely of the left-to-right reduced boundary matrix computed as explained above. Specifically, $\mathcal{P}$ is the transitive closure of the collection of pairs $i < j$ such that $j \in \mathsf{X}(i)$ for at least one $\mathsf{X} \in \{\delta, \mathsf{BD}, \mathsf{DB}, \mathsf{BB}, \mathsf{DD}\}$. For a permutation ι of ι_0, let $R(\iota)$ be the matrix R after reordering the columns and rows according to ι. Among other things, we prove that $R(\iota)$ is left-to-right reduced for all linear extensions ι of $\mathcal{P}$.

Theorem 4 *(Necessity and sufficiency). Let R be a left-to-right reduced version of the boundary matrix. The corresponding dependence structure, $\mathcal{P} \subseteq [m]^2$, is the smallest partial order on $[m]$ with linear extension ι_0 such that every linear extension ι of $\mathcal{P}$ satisfies*

- *ι corresponds to a monotonic ordering of the cells;*
- *$R(\iota)$ is left-to-right reduced;*
- *the pairing defined by $R(\iota)$ is the same as that of $R(\iota_0)$.*

Proof. We first prove that $\mathcal{P}$ is sufficient to satisfy the three claimed properties. Given two orderings, ι_0 and ι, an *inversion* is a pair of indices i, j that are ordered differently in ι_0 and in ι. Let N be the number of inversions, which we interpret as the distance from ι_0 to ι. An *elementary transposition* swaps two adjacent items, and if these items define an inversion, then the transposition decreases N by 1. It is easy to see that there is a sequence of N elementary transpositions that transforms ι_0 to ι, and none of these transpositions violates the monotonicity of the ordering since $\sigma_i \subseteq \sigma_j$ implies that i, j is not an inversion. An elementary transposition in the sequence translates to swapping the two corresponding columns and the two corresponding rows in the boundary matrix. We swap these columns and rows in the reduced version of the boundary matrix and argue that this preserves the three claimed properties. We distinguish between four cases depending on whether σ_i and σ_j give birth or death; see Fig. 2.

Case BD: σ_i gives birth and σ_j gives death. Since i, j is an inversion, we have $j \notin \mathsf{BD}(i)$, which implies that (i, j) is not a point in $\mathrm{Dgm}(\iota_0)$. The matrix remains left-to-right reduced and the pairing stays the same after swapping the columns because column i is 0, and after swapping the rows because row j does not contain the lowest non-zero entry of any column.

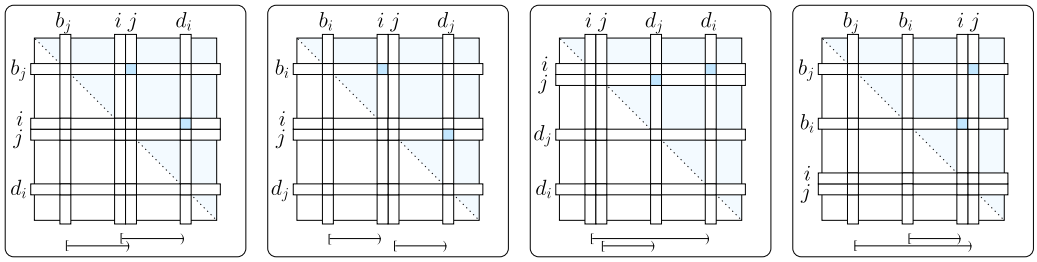


Fig. 2. Swapping the adjacent columns i and j and the corresponding adjacent rows. *First panel:* the birth-death case with $\text{low}(d_i) = i$ and $\text{low}(j) = b_j$ marked and the corresponding intervals shown. *Next three panels:* the death-birth case, the birth-birth case, and death-death case.

Case DB: σ_i gives death and σ_j gives birth. To swap the corresponding columns, we need that column i had no role in reducing column j to 0, and this is guaranteed because $j \notin \mathsf{DB}(i)$ implies $C[i, j] = 0$. To also swap the corresponding rows, we need the entry in row i and column d_j be 0, and this is guaranteed because $j \notin \mathsf{DB}(i)$ implies $R[i, d_j] = 0$.

Case BB: both σ_i and σ_j give birth. Their columns are 0 and can therefore be swapped without trouble. Swapping the rows is more delicate, and we first consider the case $d_j < d_i$ illustrated in the third panel of Fig. 2. We need the entry in row i and column d_j be 0, but this is guaranteed by $j \notin \mathsf{BB}(i)$. We second consider the case $d_i < d_j$, which is not shown in the figure. Because of $j \notin \mathsf{BB}(i)$, the entry right above $\text{low}(d_j)$ is 0, so swapping the rows preserves the pairing and the matrix to be left-to-right reduced.

Case DD: both σ_i and σ_j give death. Since $j \notin \mathsf{DD}(i)$, we can swap their columns while preserving the pairing and keeping the matrix left-to-right reduced. This holds both in the case $b_j < b_i$, which is illustrated in the fourth panel of Fig. 2, and in the case $b_i < b_j$, which is not shown. Swapping the rows causes no complications because they do not contain the lowest non-zero entry of any column.

We second prove that $\mathcal{P}$ is necessary to satisfy the three claimed properties, by which we mean that every properly contained partial order has linear extensions that violates at least one of these properties. Letting $\mathcal{P}_1 \subseteq \mathcal{P}$ be properly contained, then there exists a pair $(i, j) \in \mathcal{P} \setminus \mathcal{P}_1$ that is not derived by transitivity from other pairs in $\mathcal{P}$. Hence, $j \in \mathsf{X}(i)$ for at least one $\mathsf{X} \in \{\delta, \mathsf{BD}, \mathsf{DB}, \mathsf{BB}, \mathsf{DD}\}$. Letting ι_1 be a linear extension of $\mathcal{P}_1$ with $j < i$, every sequence of elementary transpositions that changes ι_0 to ι_1 contains one that swaps i with j. Let R' and R'' be the matrix R right before and right after swapping columns i, j and rows i, j. We can assume that before the transposition of i with j, all claimed properties are satisfied. Since $(i, j) \in \mathcal{P}$, this implies that after the transposition at least one of the properties is violated. Indeed, if $j \in \delta(i)$, then the ordering is no longer monotonic, if $j \in \mathsf{BD}(i)$, then the pairing changes, if $C[i, j] = 1$, then R'' requires a right-to-left column addition, and if $R[i, d_j] = 1$, then the pairing represented by R'' is different from that of $R = R(\iota_0)$. Either way, $\mathcal{P}_1$ violates at least one of the three properties we require from its linear extensions. □

4. Operations

Intuitively, we open holes by removing basis vectors in cohomology and close holes by adding basis vectors in homology. The dependences between the holes determine the range of complexes that can be computed by these operations, namely exactly all complexes in the filtrations defined by linear extensions of the partial order introduced in Section 3.

Opening and closing holes. We consider four types of operations, which are motivated by the fate of a p-dimensional hole in a filtration — or more precisely, of the corresponding homology class, which is generated by a p-cycle: it is born when the last p-cell completes the cycle, and it dies when the last $(p+1)$-cell completes the chain that makes the p-cycle homologous to an older p-cycle. This includes the case when the p-cycle becomes trivial. Similar to birth and death, we *lock* by completing a p-cycle and we *fill* by completing a $(p+1)$-chain. Going backward, we *unfill* by puncturing the $(p+1)$-chain, and we *unlock* by disconnecting the p-cycle; see Fig. 3 for the case $p = 1$. We have such an operation for each dimension, which we indicate by writing $\texttt{Lock}_p$, $\texttt{Fill}_p$, $\texttt{Unfill}_p$, $\texttt{Unlock}_p$. Locally, there is no difference between $\texttt{Lock}_p$ and $\texttt{Fill}_{p-1}$, but while the latter closes a $(p-1)$-dimensional hole, the former operation closes the last remaining entrance into a p-dimensional hole, which it thus creates. Since $\texttt{Unlock}_p$ and $\texttt{Unfill}_p$ are the inverses of $\texttt{Lock}_p$ and $\texttt{Fill}_p$, we see that we are really dealing with four views of one and the same operation, which of course has an instantiation in every dimension.

The difference between birth and death and the four operations is that the latter are less local. This is best illustrated for $\texttt{Unfill}_p$, whose goal is to resurrect a p-cycle from the dead. It could be that after the death of this p-cycle, several more $(p+1)$-chains were added that would have closed the p-cycle if it were still alive. Instead, these $(p+1)$-cells gave rise to $(p+1)$-cycles. If we now puncture only the original $(p+1)$-chain, we kill such a $(p+1)$-cycle instead of resurrecting the p-cycle. In order to reach its goal, $\texttt{Unfill}_p$ must first unlock all $(p+1)$-cycles that prevent the resurrection and finally puncture the original $(p+1)$-chain. There are additional dependences, which will be part of the formal description of the four operations.

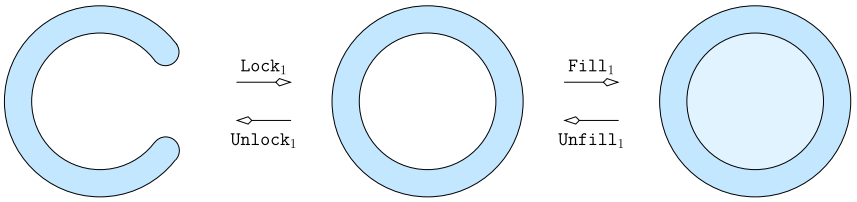


Fig. 3. A p-cycle can be created by adding a p-dimensional piece (*lock*) or by removing a $(p+1)$-dimensional piece (*unfill*). Symmetrically, a p-cycle can be destroyed by removing a p-dimensional piece (*unlock*) or by adding a $(p+1)$-dimensional piece (*fill*).

Recursive implementation. Given a point, A, in the persistence diagram, we write b_A for the first and d_A for the second coordinate, which we recall are indices of cells. We write x_A if it is not clear which of the two coordinates it is. We now explain how we implement the four operations. To this end, we introduce the *status*, which maps each point $A \in \mathrm{Dgm}(\iota_0)$ to one of three attributes, which for $M = K_\ell$ is the *future* if $\ell < b_A$, the *presence* if $b_A \le \ell < d_A$, and the *past* if $d_A \le \ell$. After manipulating the complex, the status of a point or hole depends on the operations.

The lock operation moves a point $B = (b_B, d_B)$ from the future to the presence by adding the cell with index b_B, which gives birth to the corresponding homology class (line 6). Recursively, the operation adds all dependent cells that may not appear after this cell in the ordering, as specified by the maps ∂, DB^T, and BB^T. This includes the faces of the cell (lines 2 and 3), the canonical cycle, $\mathsf{z}_p(b_B)$, stored in column b_B of U, but also other dependent cells (lines 4 and 5). Similarly, the fill operation moves $B = (b_B, d_B)$ from the presence to the past by adding the cell with index d_B, which gives death to the corresponding homology class (line 6), and by recursively adding other cells as necessary. In the event that B is not yet in the presence, it first moves it from the future to the presence.

```
1  Lock_p(b_B):   if B in future then
2                   for x_A ∈ ∂(b_B) do if x_A = b_A then M = Lock_{p−1}(b_A)
3                                       elseif x_A = d_A then M = Fill_{p−2}(d_A);
4                   for d_A ∈ DB^T(b_B) do M = Fill_{p−1}(d_A);
5                   for b_A ∈ BB^T(b_B) do M = Lock_p(b_A);
6                   return M ∪ {σ_{b_B}} with B in presence.
```

```
1  Fill_p(d_B):   if B not in past then
2                   for x_A ∈ ∂(d_B) do if x_A = b_A then M = Lock_p(b_A)
3                                       elseif x_A = d_A then M = Fill_{p−1}(d_A);
4                   for d_A ∈ DD^T(d_B) do M = Fill_p(d_A);
5                   if B in future then M = Lock_p(b_B);
6                   return M ∪ {σ_{d_B}} with B in past.
```

To prove that the two operations avoid infinite loops, we note that each recursive call decreases the parameter, which for $\mathtt{Lock}_p$ is the index of the cell that gives birth, and for $\mathtt{Fill}_p$ is the index of the cell that gives death. The second two operations delay the birth or death by removing cells. They are symmetric to the first two operations:

```
1  Unlock_p(b_A): if A not in future then
2                   for x_B ∈ δ(b_A) do if x_B = b_B then M = Unlock_{p+1}(b_B)
3                                       elseif x_B = d_B then M = Unfill_p(d_B);
4                   for b_B ∈ BB(b_A) do M = Unlock_p(b_B);
5                   if A in past then M = Unfill_p(d_A);
6                   return M \ {σ_{b_A}} with A in future.
```

```
1  Unfill_p(d_A): if A in past then
2                   for x_B ∈ δ(d_A) do if x_B = b_B then M = Unlock_{p+2}(b_B)
3                                       elseif x_B = d_B then M = Unfill_{p+1}(d_B);
4                   for b_B ∈ DB(d_A) do M = Unlock_{p+1}(b_B);
5                   for d_B ∈ DD(d_A) do M = Unfill_p(d_B);
6                   return M \ {σ_{d_A}} with A in presence.
```

To prove that the two operations avoid infinite loops, we note that each recursive call increases the parameter, which for $\mathtt{Unlock}_p$ is the index of the cell that gives birth, and for $\mathtt{Unfill}_p$ is the index of the cell that gives death.

Observe that a complex K_ℓ in the filtration of ι_0 contains exactly those cells of K that give birth and death to holes represented by points in the past plus those that give birth to holes represented by points in the presence. This relation is maintained by the four operations; that is: M consists of all cells of K whose index is a coordinate of a point in the past or the first coordinate of a point in the presence.

Status and ordering. In contrast to the vineyard algorithm in (Cohen-Steiner et al., 2006), our four operations do not maintain an ordering of the cells that is compatible with the constructed complex. Instead, they maintain the status of the points in the persistence diagram. We call a status *consistent* with the dependence structure, if the following conditions are satisfied for any two points $A = (b_A, d_A)$ and $B = (b_B, d_B)$ in $\mathrm{Dgm}(\iota_0)$:

$$(b_A, d_B) \in \mathcal{P} \Longrightarrow [A \text{ is in the future } \Rightarrow B \text{ is not in the past}]; \tag{7}$$

$$(d_A, b_B) \in \mathcal{P} \Longrightarrow [A \text{ is not in the past } \Rightarrow B \text{ is in the future}]; \tag{8}$$

$$(b_A, b_B) \in \mathcal{P} \Longrightarrow [A \text{ is in the future } \Rightarrow B \text{ is in the future}]; \tag{9}$$

$$(d_A, d_B) \in \mathcal{P} \Longrightarrow [A \text{ is not in the past } \Rightarrow B \text{ is not in the past}]. \tag{10}$$

It is not difficult to see that the status defined for K_ℓ is consistent with $\mathcal{P}$. We claim that the hole manipulating operations preserve consistency.

Lemma 5 *(Consistency of status). Let M be obtained by executing a finite sequence of hole manipulating operations starting with K_ℓ. Then the status of M is consistent with $\mathcal{P}$.*

Proof. We prove the claim by induction. Assuming conditions (7) to (10) of the definition of consistency are satisfied before an operation, we show that they are also satisfied after the operation. There are four operations to be considered, but because the arguments are almost the same in all cases, we focus on locking.

$\texttt{Lock}_p(b_B)$. When we lock B, its status changes from the future to the presence. A point $A = (b_A, d_A)$ is affected by this status change only if $d_A < b_B$ or $b_A < b_B$ in $\mathcal{P}$. We see that (8) and (9) are relevant. Since B is no longer in the future, A must move to the past in the first case and out of the future in the second case. The operation does exactly that: lines 2 and 3 adjust the faces, line 4 adjusts the points with a relation captured by (8), and line 5 adjusts the points with a relation captured by (9). Line 6 finally changes the status of B. Together, lines 2 to 6 capture all points with dependences expressed in $\mathcal{P}$. □

We show that the consistency of the status implies the existence of a linear extension of the partial order such that M belongs to the filtration of this monotonic ordering.

Lemma 6 *(Existence of linear extension). Let M be a subcomplex of K whose status is consistent with the dependence structure. Then there exists a linear extension, ι, of $\mathcal{P}$ and an index, $k \in [0, m]$, such that M is the k-th complex in the filtration of ι.*

Proof. Recall that $M \subseteq K$ contains every cell whose index is a coordinate of a point in the past or the first coordinate of a point in the presence. Hence, $N = K \setminus M$ contains every cell whose index is a coordinate of a point in the future or the second coordinate of a point in the presence. We claim there is a linear extension $\iota = \iota_M \iota_N$ of $\mathcal{P}$ such that $i \in \iota_M$ iff $\sigma_i \in M$ and $j \in \iota_N$ iff $\sigma_j \in N$. To prove this, it suffices to show that all pairs in $\mathcal{P}$ that go between M and N go in fact from M to N. Indeed, in this case we let ι_M and ι_N be arbitrary linear extensions of $\mathcal{P}$ restricted to M and to N, and we get $\iota = \iota_M \iota_N$ as a linear extension of $\mathcal{P}$. To get a contradiction, assume there is a pair $(j, i) \in \mathcal{P}$ with $j = x_A \in \iota_N$ and $i = x_B \in \iota_M$.

Case $j = b_A, i = d_B$. Since the index of every cell in N is a coordinate of a point in the future or the second coordinate of a point in the presence, A must be in the future. Similarly, since the index of every cell in M is a coordinate of a point in the past or the first coordinate of a point in the presence, B must be in the past. But this contradicts condition (7) in the definition of a consistent status.

Case $j = d_A, i = b_B$. A cannot be in the past and B cannot be in the future, which contradicts (8).

Case $j = b_A, i = b_B$. A must be in the future and B cannot be in the future, which contradicts (9).

Case $j = d_A, i = d_B$. A cannot be in the past and B must be in the past, which contradicts (10).

We conclude that $\iota = \iota_M \iota_N$ is a linear extension of $\mathcal{P}$. Setting $k = |M|$, this implies that M is the k-th complex in the corresponding filtration. □

By Lemma 6, every complex that can be constructed by a sequence of hole manipulating operations from an initial complex in the filtration of ι_0 belongs to the filtration of a linear extension of $\mathcal{P}$. Conversely, given a complex, M, in the filtration of a linear extension of $\mathcal{P}$, it is possible to design a sequence of operations that constructs M from an initial complex in the filtration of ι_0. To this end, we adjust the status of every point in the persistence diagram using the appropriate operation, if needed. Hence, the dependence structure describes precisely what can and what cannot be constructed within this framework. Write $\mathcal{K}(\mathcal{P})$ for the set of complexes in the filtrations defined by linear extensions of $\mathcal{P}$.

Corollary 7 *(Power and limitation). A complex M is constructible by a finite sequence of hole manipulating operations applied to a complex in the filtration of ι_0 iff $M \in \mathcal{K}(\mathcal{P})$.*

5. Case studies

In this section, we illustrate the results of the hole manipulating operations on Alpha shapes as defined in (Edelsbrunner and Mücke, 1994). We begin with the formal introduction of this shape representation, and follow up with concrete examples that demonstrate the utility of the operations.

Alpha shapes. Let $X \subseteq \mathbb{R}^d$ be finite and in general position. The *Voronoi domain* of a point $x \in X$ is the region of points, denoted $\mathrm{dom}(x) \subseteq \mathbb{R}^d$, that are at least as close to x as to any other point $y \in X$. The *Voronoi tessellation* is the set of domains, $\mathrm{Vor}(X) = \{\mathrm{dom}(x) \mid x \in X\}$. The *Delaunay mosaic* or *Delaunay triangulation* is isomorphic to the nerve of the tessellation, $\mathrm{Del}(X) = \{\sigma \subseteq X \mid \bigcap_{x\in\sigma} \mathrm{dom}(x) \neq \emptyset\}$, where we adopt the convention from combinatorial topology and identify a set of points, σ, with its convex hull. Assuming general position, $\mathrm{Vor}(X)$ is a primitive decomposition of $\mathbb{R}^d$ into convex polyhedra, and $\mathrm{Del}(X)$ is the geometric realization of a simplicial complex in $\mathbb{R}^d$.

For $r \geq 0$, write $B_r(x)$ for the closed ball with center $x \in X$ and radius r, and define $\mathrm{dom}_r(x) = \mathrm{dom}(x) \cap B_r(x)$. It is not difficult to see that these restricted domains form a convex decomposition of the union of balls: $\bigcup_{x\in X} B_r(x) = \bigcup_{x\in X} \mathrm{dom}_r(x)$. The *Alpha complex* of X for radius r is the geometric realization of the nerve of the restricted domains:

$$\mathrm{Alpha}_r(X) = \{\sigma \subseteq X \mid \bigcap_{x\in\sigma} \mathrm{dom}_r(x) \neq \emptyset\}. \tag{11}$$

It is a subcomplex of the Delaunay mosaic, and it has the same homotopy type as the union of balls (Edelsbrunner and Harer, 2010). The *Alpha shape* is the underlying space of the Alpha complex, namely the set of points in $\mathbb{R}^d$ covered by at least one simplex in $\mathrm{Alpha}_r(X)$. It is convenient to introduce the function $f\colon \mathrm{Del}(X) \to \mathbb{R}$ that is defined such that $f^{-1}[-\infty, r] = \mathrm{Alpha}_r(X)$ for every $r \in \mathbb{R}$. We call f the *radius function* of the Delaunay mosaic, and we note that it maps every simplex to the radius of the smallest sphere that passes through the vertices of the simplex so that no point of X is contained in the open ball bounded by the sphere. Traditionally, this is called the *smallest empty circumsphere* of the simplex. We have $f(\sigma) = 0$ for every vertex, and $f(\emptyset) = -\infty$, by convention.

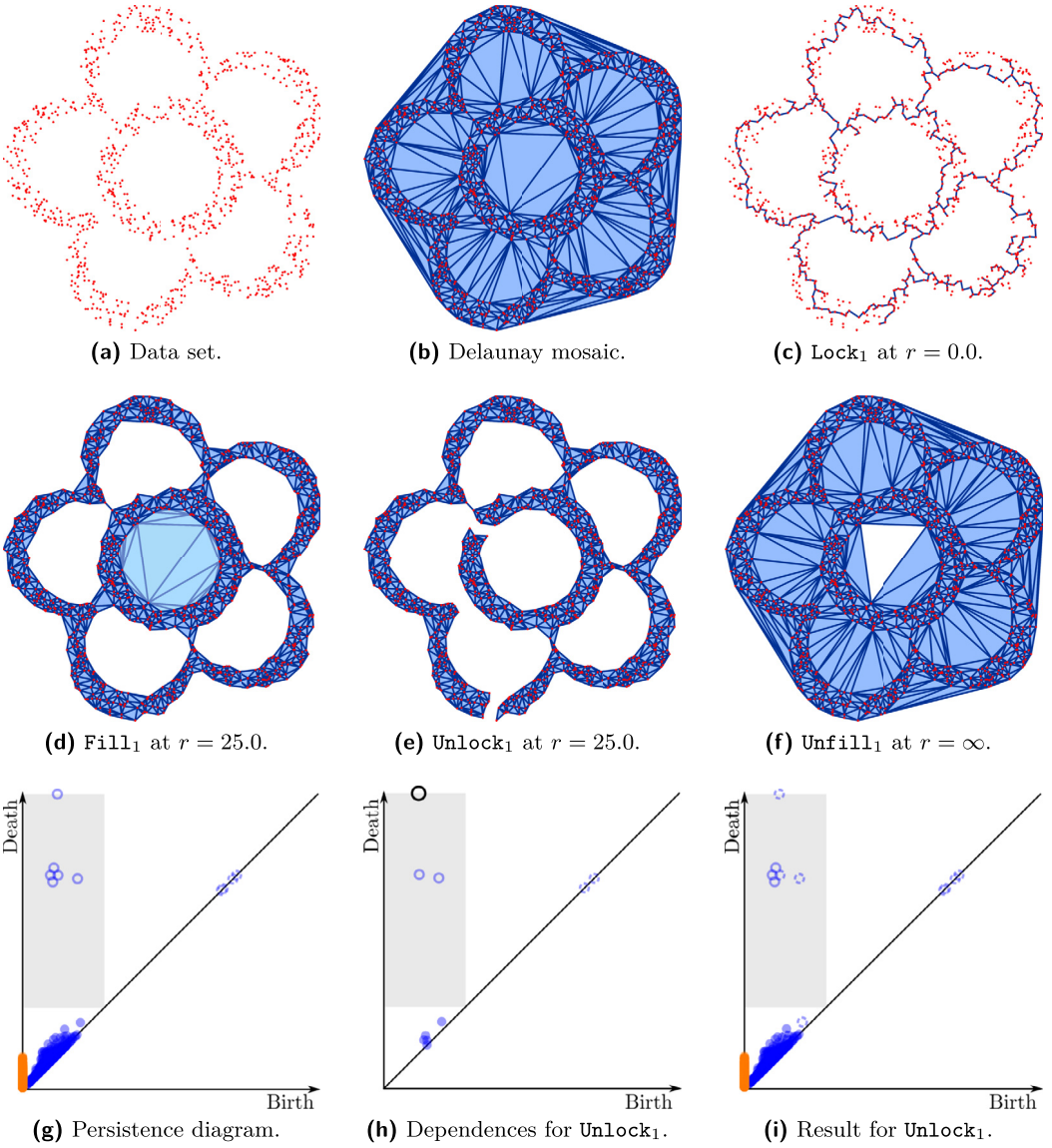
The radius function implies a partial order on the simplices, which we extend to a total order by breaking ties in favor of lower-dimensional simplices, while breaking any remaining ties arbitrarily. In other words, we order the simplices of $\mathrm{Del}(X)$ as $\sigma_0, \sigma_1, \ldots, \sigma_m$ such that $0 \leq i \leq j \leq m$ implies $f(\sigma_i) < f(\sigma_j)$, or $f(\sigma_i) = f(\sigma_j)$ and $\dim \sigma_i \leq \dim \sigma_j$. This is a monotonically ordered simplicial complex to which the operations described in Section 4 can be applied.

Case study in $\mathbb{R}^2$. For illustrative purposes, we start with a 2-dimensional example of 1 000 points sampled from the drawing of a flower; see Fig. 4 with panels (a) to (i). The data set is shown in (a), and the Delaunay mosaic — which we recall consists of all triangles whose circumcircles do not enclose any of the points — is shown in (b).

The next four panels illustrate the operations with one example each. In $\mathbb{R}^2$, we only have cycles of dimension 0 and 1, and because the latter are more interesting, all four examples are for 1-dimensional homology, which is indicated by the index of the operation. In panel (c), we start with the Alpha complex for $r = 0.0$ and lock the six 1-cycles of largest persistence. The result is a 1-dimensional complex with six loops. If we think of the data as a noisy sample of a line drawing, this could serve as a reconstruction while preserving the homotopy type. Note, however, the extra edges caused by the dependence structure in homology that are attached to the six 1-cycles. Comparing the result with the Alpha complex for $r = 25.0$, which has the same six holes, we observe that only a few of the edges and none of the triangles are forced by the dependence structure and therefore appear in the reconstruction in (c). Starting with the Alpha complex for $r = 25.0$, we show the result of filling the most persistent, inner hole in panel (d), and the result of unlocking the corresponding 1-cycle in panel (e). Observe that the unlocking operation recursively unlocks two of the five petals as well in order to connect the inner hole with the outside. In contrast, the unfill operation applied to the entire Delaunay mosaic in (f) removes a single triangle and there are no side effects caused by the dependence structure.

The persistence diagram of the radius function guides the application of the hole manipulating operations. Panel (g) shows the diagram for the 1 000 points example, using orange for the points that represent 0-cycles and blue for the points that represent 1-cycles. Encoding the status by drawing filled, unfilled, and dashed circles for holes in the past, presence, and future, we see the diagram for $r = 25.0$. Indeed, there are 6 holes in the Alpha complex, which correspond to the 6 points in the shaded upper-left quadrant. They are drawn as unfilled circles, while the points below the quadrant are drawn as filled circles, and the points to the right of the quadrant are drawn as dashed circles. In (h), we highlight the point with maximum persistence, which we select for unlocking, and we show the points of all dependent cells. The unlock operation moves all these point into the future unless they are already there; see panel (i) and compare it with the diagrams in panels (g) and (h).

Case study in $\mathbb{R}^3$. We study the effects of the hole manipulating operations on the Alpha shapes of Gramicidin A. This is a small protein that acts as an ion channel embedded in cell membranes, which explains the tunnel alongside the structure. We get the coordinates of its atom centers from the Protein Data Base (PDB) and construct Alpha complexes based on this point set. Fig. 5 shows two of the Alpha shapes as well as the persistence diagram of the radius function. We observe that one point is significantly more persistent than the others; it corresponds to the ion channel of the protein.



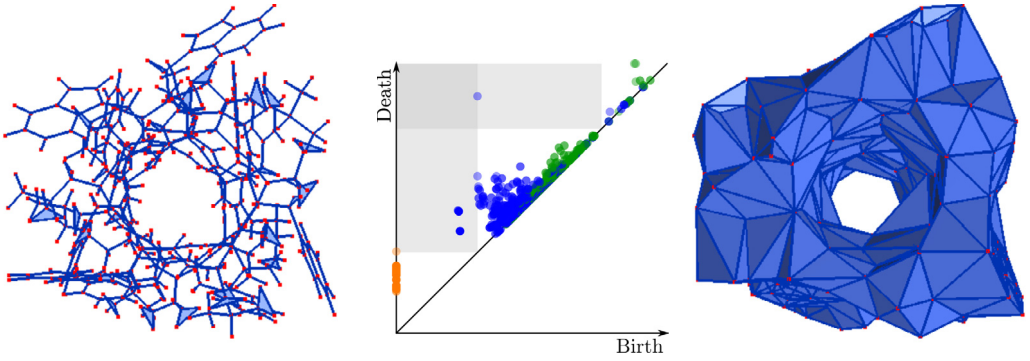
(a) Data set. **(b)** Delaunay mosaic. **(c)** $\texttt{Lock}_1$ at $r = 0.0$.

(d) $\texttt{Fill}_1$ at $r = 25.0$. **(e)** $\texttt{Unlock}_1$ at $r = 25.0$. **(f)** $\texttt{Unfill}_1$ at $r = \infty$.

(g) Persistence diagram. **(h)** Dependences for $\texttt{Unlock}_1$. **(i)** Result for $\texttt{Unlock}_1$.

Fig. 4. **(a)** The data set in $\mathbb{R}^2$, and **(b)** its Delaunay mosaic. Panels **(c)**, **(d)**, **(e)**, **(f)** show the results of a lock, fill, unlock, unfill operation, each applied to an Alpha complex of the data set. The last three panels show the persistence diagram of the radius function in **(g)**, the dependences encountered by the unlock operation applied to the point highlighted in black in **(h)**, and the resulting persistence diagram in (i).

We use the operations to manipulate the 1-cycle of highest persistence, which corresponds to the functional channel of the protein, and the 2-cycle of highest persistence, which encloses most of the volume defining the channel, as we will see shortly; see Fig. 6 with panels (a) to (i). Locking the 1-cycle at $r = 0.0$ in panel (a) effectively adds the canonical 1-cycle of the birth edge. In addition, the operation adds a small number of other edges that are forced by the dependence structure. The canonical cycle moves from the future to the presence, while the additional edges give death to 0-cycles, which move from the presence to the past. Locking the 2-cycle at $r = 0.0$ in panel (c), we see that it encloses a good portion of the volume in the tunnel, which implies that the narrow openings of the tunnel are located near its opposite ends. Filling the 1-cycle at $r = 0.93$ in panels (d) and (f) results in almost the same surface, except that it remains open at one end and therefore does not enclose any volume. Unlocking the maximum persistence 1-cycle at $r = 2.35$ in (g) gives a surprising result: instead of slicing open the cylinder with a straight cut along one side, we see a spiral cut that leaves a spiraling tube revealing the helix structure of the protein. Indeed, the connections are weaker and the distances are larger between contiguous 360° turns of the helix than along the helix, so cutting there is the action of least effort that achieves the desired result. On the other hand, unfilling the same 1-cycle at $r = 3.10$ in (i) carves out a narrow tunnel that passes through the protein. The operation resurrects this tunnel by moving the corresponding 1-cycle from the past back to the presence. It opens a few additional tunnels as side effects caused by the dependence structure.



(a) Alpha shape at $r = 0.93$. **(b)** Persistence diagram. **(c)** Alpha shape at $r = 2.35$.

Fig. 5. Alpha shapes and persistence diagram of Gramicidin A. Triangles with no incident tetrahedra are drawn in pale blue, while the others are drawn in darker blue, with the shade depending on the normal vector. The high persistent tunnel is born at $r = 0.93$ and is the only hole of the Alpha complex at $r = 2.35$. The persistence diagram uses orange for 0-cycles, blue for 1-cycles, and green for 2-cycles. The holes of the Alpha complexes at $r = 0.93$ and $r = 2.35$ are highlighted by showing the corresponding quadrants anchored on the diagonal. In the quadrant for $r = 0.93$, we seem to see only two points, but they are both of non-trivial multiplicity and correspond to the pentagons and hexagons of the Alpha shape in **(a)**.

Table 1
Average numbers of simplices in the Delaunay mosaic of a Poisson point process in the unit cube.

	$p = -1$	0	1	2	3	Total
Experiment I	1.0	124.1	817.7	1 357.6	663.1	2 963.5
Experiment II	1.0	510.8	3 699.5	6 323.5	3 133.9	13 668.7
Experiment III	1.0	998.4	7 397.6	12 730.7	6 330.5	27 458.3

Table 2
Comparison between the standard reduction algorithm and the exhaustive reduction algorithm for random points in $[0, 1]^3$. We quantify the *density* of a matrix as the percentage of non-zero elements. *Upper half:* matrices computed by column reduction. *Lower half:* matrices computed by row reduction, in which Q corresponds to R, V to U, and D to C.

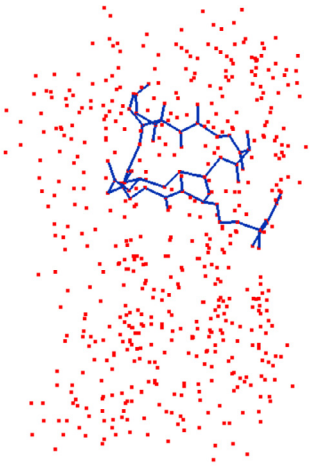
	Experiment I		Experiment II		Experiment III	
	std	exh	std	exh	std	exh
density of R	0.061	0.159	0.014	0.046	0.007	0.026
density of U	0.203	0.336	0.065	0.124	0.039	0.078
density of C	0.118	0.076	0.032	0.018	0.017	0.009
density of Q	0.093	0.135	0.022	0.036	0.011	0.019
density of V	0.203	0.308	0.065	0.107	0.039	0.065
density of D	0.142	0.044	0.043	0.010	0.024	0.005

6. Statistics

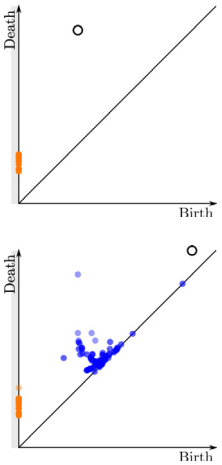
This section presents statistics about the sizes of the structures relevant for the hole manipulating operations. We have C++ implementations in 2 and 3 dimensions using the CGAL-library (CGAL) for geometric computations, and collect data for random point sets generated according to the Poisson point process. We only show results for the more important 3-dimensional case. We perform Experiments I, II, and III for an expected number of 125, 512, and 1000 points in $[0, 1]^3$, respectively, averaging the results over 100 runs each. We begin with the size of the simplicial complex, which in our case is the Delaunay mosaic of the points; see Table 1. Since the Delaunay mosaic is homologically trivial, the numbers of simplices determine the sizes of the components of the tri-partition discussed in Section 2.

Densities of matrices. We compute the dependence structure defined in Section 3 with the exhaustive reduction algorithm. In comparison to the standard reduction algorithm, it produces denser matrices R and U but performs fewer column additions. Comparing the density of C for the standard and the exhaustive column reduction algorithms in Table 2, we see that the latter uses only about half the number of column additions. Perhaps this is because the extra time invested in properly reducing early columns pays off later, when these columns are used to reduce later columns. The difference between standard and exhaustive reduction is even more pronounced when we work with rows rather than with columns.

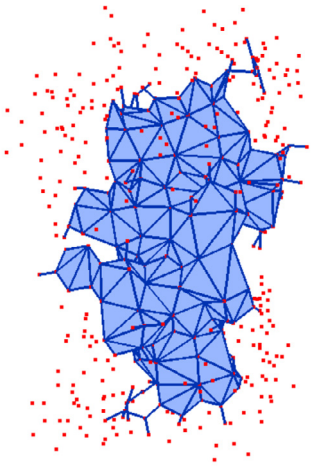
The densities of the matrices have a direct influence on the number of pairs that make up the dependence structure. Focusing on the dependence structure for homology, U stores the canonical cycles and chains that are used to close holes,



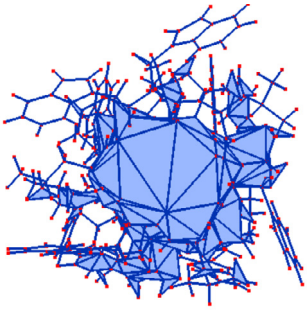
(a) $\texttt{Lock}_1$ at $r = 0.0$.



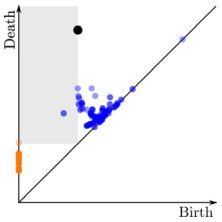
(b) Dependences.



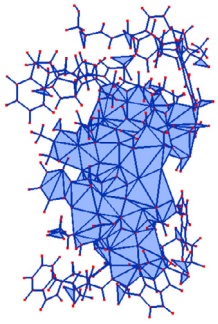
(c) $\texttt{Lock}_2$ at $r = 0.0$.



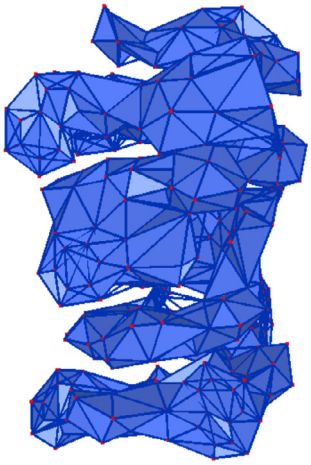
(d) $\texttt{Fill}_1$ at $r = 0.93$, top view.



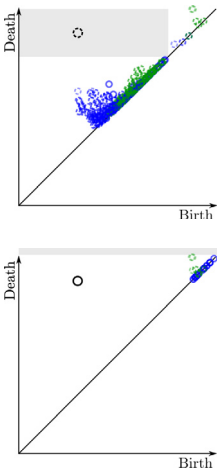
(e) Dependences.



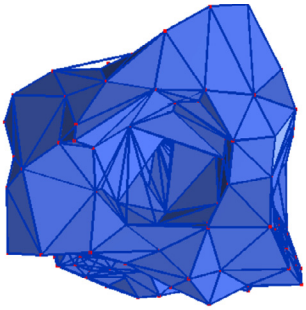
(f) $\texttt{Fill}_1$ at $r = 0.93$.



(g) $\texttt{Unlock}_1$ at $r = 2.35$.



(h) Dependences.



(i) $\texttt{Unfill}_1$ at $r = 3.10$.

Fig. 6. Results of operations manipulating the 1-cycle and 2-cycle of highest persistence in the Delaunay mosaic of Gramicidin A. The *middle column* shows the dependences for the operations illustrated in the corresponding rows. **(a)** Locking the 1-cycle at $r = 0.0$. **(c)** Locking the 2-cycle at $r = 0.0$. **(d)** Top view after filling the 1-cycle at $r = 0.93$. **(f)** Side view of the same complex. **(g)** Unlocking the 1-cycle at $r = 2.35$ reveals the helix structure of the protein. **(i)** Unfilling the 1-cycle at $r = 3.10$ opens a narrow tunnel passing through the length of the protein (not really visible).

Table 3
Average number of dependences of different types originating from a p-simplex. *Upper half:* to capture the pairs relevant for locking and filling, we count the dependences in forward direction, distinguishing the ones that originate from birth-giving and from death-giving simplices. *Lower half:* to capture the pairs relevant for unlocking and unfilling, we count the dependences in backward direction, distinguishing again the ones that originate from birth-giving and from death-giving simplices.

	Experiment I				Experiment II				Experiment III			
	0	1	2	3	0	1	2	3	0	1	2	3
δ birth	13.2	4.8	1.9	-	14.5	4.9	2.0	-	14.8	4.9	2.0	-
BD	1.0	1.0	1.0	-	1.0	1.0	1.0	-	1.0	1.0	1.0	-
BB	0.9	0.2	0.0	-	0.9	0.3	0.1	-	0.9	0.3	0.1	-
δ death	13.2	6.2	2.0	0.0	13.8	6.5	2.0	0.0	14.8	6.6	2.0	0.0
DB	123.1	55.4	10.2	0.0	509.8	87.8	13.7	0.0	997.4	104.8	15.3	0.0
DD	0.0	1.2	1.4	0.9	0.0	1.3	1.5	1.0	0.0	1.3	1.6	1.0
total	16.0	14.5	8.4	0.9	17.4	18.5	10.2	1.0	17.7	20.6	11.0	1.0
∂ birth	1.0	2.0	3.0	-	1.0	2.0	3.0	-	1.0	2.0	3.0	-
DB^T	1.0	9.8	10.7	-	1.0	14.0	14.0	-	1.0	16.3	15.5	-
BB^T	0.9	0.2	0.0	-	0.9	0.3	0.1	-	0.9	0.3	0.1	-
∂ death	1.0	2.0	3.0	4.0	1.0	2.0	3.0	4.0	1.0	2.0	3.0	4.0
BD^T	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DD^T	0.0	1.2	1.4	0.9	0.0	1.3	1.5	1.0	0.0	1.3	1.6	1.0
total	2.8	10.8	9.5	5.9	2.9	14.6	11.2	6.0	2.9	16.7	12.0	6.0

and C stores the same information in hierarchical form. The canonical cocycles and cochains that are used to open holes are stored in V, but since the dependence structure is built from the column and not the row reduced matrices, they are replaced by functionally equivalent information.

Number of dependences. The total number of dependences is roughly an order of magnitude larger than the number of simplices. Specifically, we have about 26, 145, and 318 thousand dependences on average in Experiments I, II, and III, and a comparison with Table 1 shows that this is about ten times the total number of simplices in the mosaics. We provide detailed quantitative information in Table 3, which differentiates between types and dimensions. More precisely, for each type, dimension, and experiment, Table 3 gives the average number of pairs of the given type that originate from a simplex of the given dimension. Some of these numbers do not depend on the sampled points, such as the dependences of type BD, of which there is exactly one per simplex (counting the pair twice, once in forward direction and once in backward direction), and the face pairs, of which there are exactly $p+1$ for each p-simplex. Because of symmetry, we get the same total number of pairs in forward and in backward direction. Since pairs go from left to right and lower-dimensional cells tend to precede higher-dimensional cells in this order, it is not surprising that the average out-degree in the forward direction is higher for lower dimensions and in the backward direction is higher for higher dimensions. Note that there is exactly one vertex that gives death, and this vertex accumulates the largest number of dependences of type DB.

We observe that the numbers barely change between experiments, which suggests that they are primarily local. The numbers we get for the Gramicidin data are very similar to those in Table 3.

Size of operations. Recall that the operations open and close holes by recursive computation. Rather than just the pairs originating from the simplex to which the operation is applied, we need the entire emanating paths to understand the action taken by an operation. Consider for example the lock operation applied to σ_i. By design, the algorithm adds the missing simplices in the canonical cycle defined by σ_i. Rather than fetching these simplices from the matrix U, which stores the canonical cycle in column i, the algorithm finds the missing simplices by following paths in $\mathcal{P}$. Generally, these paths contain more simplices than just those in the canonical cycle. The same can be said about filling, and the situation is similar but not entirely symmetric for unlocking and unfilling, which replace the removal of the canonical cocycles and cochains by functionally equivalent deletions of cells. This asymmetry is caused by basing the dependence structure on the column reduced rather than the row reduced matrices.

Table 4 sheds light on the difference by giving the average sizes of the canonical cycles, chains, cocycles, and cochains, as well as the average numbers of cells along the relevant paths of the dependence structure. The latter set contains all cells that are possibly affected by the operation, which implies that their number gives an upper bound on the average number of status changes per operation, but this upper bound is likely to be rather loose. We observe an anomaly in Experiment III, in which the average size of the canonical 2-cochain is 13.7, while the average number of dependent 2-cells for unlocking is only 13.0. This does not contradict the correctness of the unlock operation, which we recall is based on the column reduced matrices and therefore finds a faster way to unlock the 2-cycles than by removing the canonical 2-cochains.

Table 4
Upper half: average number of simplices of a canonical cycle, cochain, chain, cocycle. *Lower half:* average number of dependences for locking, filling, unlocking, unfilling. We disregard the status and count every dependent cell. The order of rows in the two halves is parallel, stressing the relation between the operation and the targeted feature, which for $\texttt{Lock}_p$ is a p-cycle, for $\texttt{Fill}_p$ is a $(p+1)$-chain, for $\texttt{Unlock}_p$ is a p-cochain, and for $\texttt{Unfill}_p$ is a $(p+1)$-cocycle.

	Experiment I			Experiment II			Experiment III		
	$p=0$	1	2	$p=0$	1	2	$p=0$	1	2
p-cycle	2.0	10.9	12.1	2.0	17.6	17.3	2.0	21.9	20.1
$(p+1)$-chain	4.5	12.1	7.0	6.6	25.9	10.8	8.3	35.7	12.9
p-cochain	5.1	8.6	6.8	6.8	15.5	11.1	7.3	20.3	13.7
$(p+1)$-cocycle	56.6	11.6	1.0	105.1	17.1	1.0	135.2	19.9	1.0
$\texttt{Lock}_p$	4.0	50.2	139.0	4.5	112.6	382.2	5.0	176.0	633.2
$\texttt{Fill}_p$	15.1	88.1	157.1	24.5	234.2	414.6	34.4	382.8	677.2
$\texttt{Unlock}_p$	707.7	78.9	7.1	2 093.6	222.0	11.0	3 570.2	362.7	13.0
$\texttt{Unfill}_p$	686.7	77.1	6.0	2 065.8	218.5	9.8	3 536.4	357.8	11.9

7. Discussion

The main contribution of this paper is a mathematical framework for manipulating hole systems in complexes and software that implements the operations in 3 dimensions. The main new concept is the dependence structure of an ordered complex, which is a partial order on the cells such that the filtrations of its linear extensions characterize what can and what cannot be constructed within this framework. Here are some structural questions about the framework that remain open:

- We can reduce the boundary matrix with column or with row operations and we can choose a strategy anywhere between standard and exhaustive reduction. Characterize how the partial orders computed with different reduction algorithms differ from each other.
- Is it true that the linear extensions of the partial orders obtained from all possible reduced versions of an ordered boundary matrix exhaust the equivalence class of monotonic orderings with same persistence pairing? If yes, is there a compact representation of this collection of monotonic orderings?
- Keeping the reduced matrix constant is a rather stringent requirement. Can this be relaxed — for example to keeping the birth-death pairs constant — without sacrificing any of the structural results?

The existence of the partial order opens up new opportunities, such as decorating the persistence diagram with additional structural information about the data, or polynomial-time algorithms for questions that seemed unapproachable before.

- What is the geometric or topological meaning of the degree of a cell in the dependence structure, possibly differentiating between types of pairs?
- Are there worthwhile optimization questions on hole systems over the collection of linear extensions of a partial order that can be solved in polynomial time, for example by flow algorithms?

The software introduced in this paper promises to be useful in the study of biomolecules, as mentioned in the introduction. It will be interesting to determine application questions in this area that have the potential to benefit from the new capabilities, and to incorporate the software in domain-specific packages that help in the better understanding of the biochemical basis of life.

Declaration of Competing Interest

We do not have any competing interests.

References

Cohen-Steiner, D., Edelsbrunner, H., Morozov, D., 2006. Vines and vineyards by updating persistence in linear time. In: Proc. 22nd Ann. Sympos. Comput. Geom., pp. 119–126.

CGAL. Computational Geometry Algorithms Library. http://www.cgal.org.

Edelsbrunner, H., 2003. Surface reconstruction by wrapping finite point sets in space. In: Aronov, B., Basu, S., Pach, J., Sharir, M. (Eds.), Discrete and Computational Geometry. The Goodman–Pollack Festschrift. Springer-Verlag, pp. 379–404.

Edelsbrunner, H., Facello, M.A., Fu, P., Liang, J., 1995. Measuring proteins and voids in proteins. In: Proc. 28th Ann. Hawaii Internat. Conf. System Sci.. In: Biotech. Comput., vol. V, pp. 256–264.

Edelsbrunner, H., Harer, J.L., 2010. Computational Topology. An Introduction. Amer. Math. Soc., Providence, Rhode Island.

Edelsbrunner, H., Mücke, E.P., 1994. Three-dimensional alpha shapes. ACM Trans. Graph. 13, 43–72.

Edelsbrunner, H., Ölsböck, K., 2018. Tri-partitions and bases of an ordered complex. Austria, Klosterneuburg, Austria. Manuscript, IST.

Edelsbrunner, H., Zomorodian, A., 2003. Computing linking numbers of a filtration. Homology, Homotopy, and Applications 5, 19–37.

Hatcher, A., 2002. Algebraic Topology. Cambridge Univ. Press, Cambridge, England.
Kalai, G., 1983. Enumeration of q-acyclic simplicial complexes. Isr. J. Math. 45, 337–351.
Kurlin, V., 2016. A fast persistence-based segmentation of noisy 2D clouds with provable guarantees. Pattern Recognit. Lett. 83, 3–12.
Leach, A.R., 2001. Molecular Modelling: Principles and Applications, second edition. Prentice Hall.
Lee, Y.L., Barthel, S.D., Dlotko, P., Moosavi, S.M., Hess, K., Smit, B., 2017. Quantifying similarity of pore-geometry in nanoporous materials. Nat. Commun. 8, 15396.
Munkres, J.R., 1984. Elements of Algebraic Topology. Perseus, Cambridge, Massachusetts.
Rosenstiehl, P., Read, R.C., 1978. On the principal edge tripartition of a graph. Ann. Discrete Math. 3, 195–226.
Simões, T., Lopes, D., Dias, S., Fernandes, F., Pereira, J., Jorge, J., Bajaj, C., Gomes, A., 2017. Geometric detection algorithms for cavities on protein surfaces in molecular graphics: a survey. Comput. Graph. Forum 36, 643–683.