Lexicographic optimal homologous chains and applications to point cloud triangulations

David Cohen-Steiner, André Lieutier, Julien Vuillamy

Lexicographic optimal homologous chains and applications to point cloud triangulations

David Cohen-Steiner[1], André Lieutier[2], and Julien Vuillamy[1,2]

[1]Université Côte d'Azur, France and Inria Sophia Antipolis-Mediterranée, France
[2]Dassault Systèmes, France

December 3, 2019

Abstract

This paper considers a particular case of the Optimal Homologous Chain Problem (OHCP), where optimality is meant as a minimal lexicographic order on chains induced by a total order on simplices. The matrix reduction algorithm used for persistent homology is used to derive polynomial algorithms solving this problem instance, whereas OHCP is NP-hard in the general case. The complexity is further improved to a quasilinear algorithm by leveraging a dual graph minimum cut formulation when the simplicial complex is a strongly connected pseudomanifold. We then show how this particular instance of the problem is relevant, by providing an application in the context of point cloud triangulation.

1 Introduction

The computation of minimal simplicial homology generators has been a wide subject of interest for its numerous applications related to shape analysis, computer graphics or computer-aided design. Coined in [18], we recall the Optimal Homologous Chain Problem (OHCP):

Problem 1.1 (OHCP)**.** *Given a p-chain A in a simplicial complex K and a set of weights given by a diagonal matrix W of appropriate dimension, find the* 1*-norm minimal chain* $\Gamma_{\min}$ *homologous to A:*

$$\Gamma_{\min} = \min_{\Gamma, B} ||W \cdot \Gamma||_1 \text{ such that } \Gamma = A + \partial_{d+1} B \text{ and } \Gamma \in \boldsymbol{C_p}(K), B \in \boldsymbol{C_{p+1}}(K)$$

It has been shown that OHCP is NP-hard in the general case when using coefficients in $\mathbb{Z}_2$ [12, 8]. However, we consider a specialization of this problem: the Lexicographic Optimal Homologous Chain Problem (Lex-OHCP). Using coefficients in $\mathbb{Z}_2$, minimality is now meant according to a lexicographic order on chains induced by a total order on simplices. Formulated in the context of OHCP, this would require ordering the simplices using a total order and taking a weight matrix W with generic term $W_{ii} = 2^i$, allowing the L^1-norm minimization to be equivalent to a minimization along the lexicographic order.

After providing required definitions and notations (Section 2), we show how an algorithm based on the matrix reduction algorithm used for the computation of persistent homology [22] allows to solve this particular instance of OHCP in $O(n^3)$ worst case complexity (Section 3). Using a very similar process, we show that the problem of finding a minimal d-chain bounding a given $(d-1)$-cycle admits a similar algorithm with the same algorithmic complexity (Section 4). Section 5 then considers Lex-OHCP in the case where the simplicial complex K is a strongly connected $(d+1)$-pseudomanifold. By formulating it as a Lexicographic Minimum Cut (LMC) dual problem, the algorithm can be improved to a quasilinear complexity. The complexity of the graph minimum cut – or equivalently maximum flow – over arbitrary weights is $\mathcal{O}(E^2)$ for a graph with E edges [23]. Its lexicographic variant can however be performed in $\mathcal{O}(E \log E)$ complexity: the cost of sorting the graph edges and performing a $\mathcal{O}(E\alpha(E))$ algorithm based on disjoint-sets, where α is the inverse Ackermann function. Section 6 legitimizes this restriction of OHCP by providing an application of the developed Lex-OHCP algorithms to point cloud triangulation. After defining a total order closely related to the Delaunay triangulation, we provide details on an open surface algorithm given a boundary and a watertight surface reconstruction algorithm given an interior and exterior information.

Several authors have studied algorithm complexities for the computation of $L1$-norm optimal cycles in homology classes [24, 10, 8, 11, 20, 12, 18, 19]. However, to the best of our knowledge, considering lexicographic-minimal chains in their homology classes is a new idea. When minimal cycles are of codimension 1 in a pseudo-manifold, the idea of considering the minimal cut problem on the dual graph has been previously studied. In particular, Chambers et al. [8] have considered graph duality to derive complexity results for the computation of optimal homologous cycles on 2-manifolds. Chen et al. [12] also use a reduction to a minimum cut problem on a dual graph to compute minimal non-null homologous cycles on n-complexes embedded in $\mathbb{R}^n$. Their polynomial algorithm (Theorem 5.2.3 in [12]) for computing a homology class representative of minimal radius is reminiscent of our algorithm for computing lexicographic minimal representatives (Algorithm 4). In a recent work [19], Dey et al. consider the dual graph of pseudo-manifolds in order to obtain polynomial time algorithms for computing minimal persistent cycles. Since they consider arbitrary weights, they obtain the $\mathcal{O}(n^2)$ complexity of best known minimum cut/maximum flow algorithms [31]. The lexicographic order introduced in our work can be derived from the idea of a variational formulation of the Delaunay triangulation, first introduced in [13] and further studied in [1, 14]. Finally, many methods have been proposed to answer the problem of surface reconstruction in specific acquisition contexts [27, 28, 30]: [29] classifies a large number of these methods according to the assumptions and information used in addition to geometry. In the family of purely geometric reconstruction based on a Delaunay triangulation, one very early contribution is the sculpting algorithm by Boissonnat [6]. The crust algorithm by Amenta et al. [2, 3] and an algorithm based on natural neighbors by Boissonnat et al. [7] were the first algorithms to guarantee a triangulation of the manifold under sampling conditions. However, these general approaches usually have difficulties far from these sampling conditions, in applications where point clouds are noisy or under-sampled. This difficulty can be circumvented by providing additional information on the nature of the surface [17, 21]. Our contribution lies in this category of approaches. We provide some topological information of the surface: a boundary for the open surface reconstruction and an interior region and exterior region for the closed surface reconstruction.

2 Definitions

2.1 Simplicial complexes

Consider an independent family $A = (a_0, \ldots, a_d)$ of points of $\mathbb{R}^N$. We call a $\boldsymbol{d}$**-simplex** σ spanned by A the set of all points: $x = \sum_{i=0}^{d} t_i a_i$, where $\forall i \in [0, d], t_i \geq 0$ and $\sum_{i=0}^{d} t_i = 1$ Any simplex spanned by a subset of A is called a **face** of σ.

A **simplicial complex** K is a collection of simplices such that every face of a simplex of K is in K and the intersection of two simplices of K is either empty either a common face.

2.2 Simplicial chains.

Let K be a simplicial complex of dimension at least d. The notion of chains can be defined with coefficients in any ring but we restrict here the definition to coefficients in the field $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$. A $\boldsymbol{d}$**-chain** A with coefficients in $\mathbb{Z}_2$ is a formal sum of d-simplices :

$$A = \sum_i x_i \sigma_i, \text{ with } x_i \in \mathbb{Z}_2 \text{ and } \sigma_i \in K \tag{1}$$

We denote $\boldsymbol{C_d}(K)$ the vector space over the field $\mathbb{Z}_2$ of d-chains in the complex K. Interpreting the coefficient $x_i \in \mathbb{Z}_2 = \{0, 1\}$ in front of simplex σ_i as indicating the existence of σ_i in the chain A, we can view the d-chain A as a set of simplices : for a d-simplex σ and a d-chain A, we write that $\sigma \in A$ if the coefficient for σ in A is 1. With this convention, the sum of two chains corresponds to the symmetric difference on their sets. In what follows, a d-simplex σ can also be interpreted as the d-chain containing only the d-simplex σ.

2.3 Boundary operator.

For a d-simplex $\sigma = [a_0, \ldots, a_d]$, the **boundary operator** is defined as the operator:

$$\partial_d : \boldsymbol{C_d}(K) \to \boldsymbol{C_{d-1}}(K)$$
$$\partial_d \sigma \underset{\text{def.}}{=} \sum_{i=0}^{d} [a_0, \ldots, \widehat{a_i}, \ldots, a_d]$$

where the symbol $\widehat{a_i}$ means the vertex a_i is deleted from the array. The kernel of the boundary operator $Z_d = \operatorname{Ker} \partial_d$ is called the group of d-cycles and the image of the operator $B_d = \operatorname{Im} \partial_{p+1}$ is the group of d-boundaries. We say two d-chains A and A' are **homologous** if $A - A' = \partial_{d+1} B$, for some $(d+1)$-chain B.

2.4 Lexicographic order.

We assume now a total order on the d-simplices of K, $\sigma_1 < \cdots < \sigma_n$, where $n = \dim \boldsymbol{C_d}(K)$. From this order, we define a lexicographic total order on d-chains.

Definition 2.1 (Lexicographic total order)**.** *For $C_1, C_2 \in \boldsymbol{C_d}(K)$:*

$$C_1 \sqsubseteq_{lex} C_2 \underset{def.}{\iff} \begin{cases} C_1 + C_2 = 0 \\ \text{or} \\ \sigma_{\max} = \max\{\sigma \in C_1 + C_2\} \in C_2 \end{cases}$$

This total order naturally extends to a strict total order $\sqsubset_{lex}$ on $\boldsymbol{C_d}(K)$.

3 Lexicographic optimal homologous chain

3.1 Problem statement

In this section, we define the Lexicographic Optimal Homologous Chain Problem (Lex-OHCP), a particular instance of OHCP (Problem 1.1):

Problem 3.1 (Lex-OHCP)**.** *Given a simplicial complex K with a total order on the d-simplices and a d-chain $A \in \boldsymbol{C_d}(K)$, find the unique chain $\Gamma_{\min}$ defined by :*

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \{\Gamma \in \boldsymbol{C_d}(K) \mid \exists B \in \boldsymbol{C_{d+1}}(K), \Gamma - A = \partial_{d+1} B\}$$

Definition 3.1. *A d-chain $A \in \boldsymbol{C_d}(K)$ is said **reducible** if there is a d-chain $\Gamma \in \boldsymbol{C_d}(K)$ (called **reduction**) and a $(d+1)$-chain $B \in \boldsymbol{C_{d+1}}(K)$ such that:*

$$\Gamma \sqsubset_{lex} A \quad \textit{and} \quad \Gamma - A = \partial_{d+1} B$$

*If this property cannot be verified, the d-chain A is said **irreducible**. If A is reducible, we call **total reduction** of A the unique irreducible reduction of A. If A is irreducible, A is said to be its own total reduction.*

Problem 3.1 can be reformulated as finding the total reduction of A.

3.2 Boundary matrix reduction

With $m = \dim \boldsymbol{C_d}(K)$ and $n = \dim \boldsymbol{C_{d+1}}(K)$, we now consider the m-by-n boundary matrix ∂_{d+1} with entries in $\mathbb{Z}_2$. We enforce that rows of the matrix are ordered according to a given strict total order on d-simplices $\sigma_1 < \cdots < \sigma_m$, where the d-simplex σ_i is the basis element corresponding to the i^{th} row of the boundary matrix. The order of columns, corresponding to an order on $(d+1)$-simplices, is not relevant for this section and can be chosen arbitrarily.

For a matrix R, the index of the lowest non-zero coefficient of a column R_j is denoted by $\text{low}(j)$, or sometimes $\text{low}(R_j)$ when we want to explicit the considered matrix. This index is not defined for a column whose coefficients are all zero.

Algorithm 1 is a slightly modified version of the boundary reduction algorithm presented in [22]. Indeed, for our purpose, we do not need the boundary matrix storing all the simplices of all dimensions and apply the algorithm to the sub-matrix $\partial_{d+1} : \boldsymbol{C_{d+1}}(K) \to \boldsymbol{C_d}(K)$. One checks easily that Algorithm 1 has $\mathcal{O}(mn^2)$ time complexity. We now introduce a few lemmas useful for

Algorithm 1: Reduction algorithm for the ∂_{d+1} matrix

```
R = ∂_{d+1}
for j ← 1 to n do
    while R_j ≠ 0 and ∃j_0 < j with low(j_0) = low(j) do
        R_j ← R_j + R_{j_0}
    end
end
```

solving Problem 3.1. We allow ourselves to consider each column R_j of the matrix R, formally an element of $\mathbb{Z}_2^m$, as the corresponding d-chain in the basis $(\sigma_1, \ldots, \sigma_m)$.

Lemma 3.2. *A d-chain A is reducible if and only if at least one of its d-simplices is reducible.*

Proof. If there is a reducible d-simplex $\sigma \in A$, A is reducible by the d-chain $A' = A - \sigma + Red(\sigma)$, where $Red(\sigma)$ is a reduction for σ.
We suppose A to be reducible. Let $\Gamma \sqsubset_{lex} A$ be a reduction for A and B the $(d+1)$-chain such that $\Gamma - A = \partial B$. We denote $\sigma_{\max} = \max\{\sigma \in A - \Gamma\}$. Note that $\sigma_{\max}$ is homologous to $\Gamma - A + \sigma_{\max}$. The chain $\Gamma - A + \sigma_{\max}$ only contains simplices smaller than $\sigma_{\max}$, by definition of the lexicographic order (Definition 2.1). We have thus shown that if A is reducible, it contains at least one simplex that is reducible. □

Lemma 3.3. *After matrix reduction (Algorithm 1), a non-zero column $R_j \neq 0$ can be described as*

$$R_j = \sigma_{\text{low}(j)} + \Gamma, \textit{ where } \Gamma \textit{ is a reduction for } \sigma_{\text{low}(j)}.$$

Proof. As all matrix operations performed on R by the reduction algorithm are linear, each non-zero column R_j of R is in the image of ∂_{d+1}. Therefore, there exist a $(d+1)$-chain B such that $R_j = \sigma_{\text{low}(j)} + \Gamma = \partial_{d+1}B$, which, is equivalent in $\mathbb{Z}_2$ to $\Gamma - \sigma_{\text{low}(j)} = \partial_{d+1}B$. By definition of the low of a column, we also have immediately: $\Gamma \sqsubset_{lex} \sigma_{\text{low}(j)}$. For each non-zero column, the largest simplex is therefore reducible by the other d-simplices of the column. □

Lemma 3.4. *After matrix reduction (Algorithm 1), there is a one-to-one correspondence between the reducible d-simplices and non-zero columns of R:*

$$\sigma_i \in \boldsymbol{C_d}(K) \textit{ is reducible } \iff \exists j \in [1, n], R_j \neq 0 \textit{ and } \text{low}(j) = i$$

Proof. Lemma 3.3 shows immediately that the simplex corresponding to the lowest index of a non-zero column is reducible.

Suppose now that a d-simplex $\tilde{\sigma}$ is reducible and let $\tilde{\Gamma}$ be a reduction of it: $\tilde{\sigma} + \tilde{\Gamma} = \partial_{d+1}B$ and $\tilde{\Gamma} \sqsubset_{lex} \tilde{\sigma}$. Algorithm 1 realizes the matrix factorization $R = \partial_{d+1} \cdot V$, where matrix V is invertible

[22]. It follows that $\operatorname{Im} R = \operatorname{Im} \partial_{d+1}$. Therefore, non-zero columns of R span $\operatorname{Im} \partial_{d+1}$ and since $\tilde{\sigma} + \tilde{\Gamma} = \partial_{d+1} B \in \operatorname{Im} \partial_{d+1}$, there is a family $(R_j)_{j \in J} = (\sigma_{\text{low}(j)}, \Gamma_j)_{j \in J}$ of columns of R such that :

$$\tilde{\sigma} + \tilde{\Gamma} = \sum_{j \in J} \sigma_{\text{low}(j)} + \Gamma_j$$

Every $\sigma_{\text{low}(j)}$ represents the largest simplex of a column, and Γ_j a reduction chain for $\sigma_{\text{low}(j)}$. As observed in section VII.1 of [22], one can check that the low indexes in R are unique after the reduction algorithm. Therefore, there is a $j_{\max} \in J$ such that for all j in $J \setminus \{j_{\max}\}$, $\text{low}(j) < \text{low}(j_{\max})$, which implies:

$$\sigma_{j_{\max}} = \max\{\sigma \in \sum_{j \in J} \sigma_{\text{low}(j)} + \Gamma_j\} = \max\left\{\sigma \in \tilde{\sigma} + \tilde{\Gamma}\right\} = \tilde{\sigma}$$

We have shown that for the reducible simplex $\tilde{\sigma}$, there is a non-zero column $R_{j_{\max}}$ with $\tilde{\sigma} = \sigma_{\text{low}(j_{\max})}$ as its largest simplex. □

3.3 Total reduction algorithm

Combining the three previous lemmas give the intuition on how to construct the total reduction solving Problem 3.1: Lemma 3.2 allows to consider each simplex individually, Lemma 3.4 characterizes the reducible nature of a simplex using the reduced boundary matrix and Lemma 3.3 describes a column of the reduction boundary matrix as a simplex and its reduction. We now present Algorithm 2, referred to as the **total reduction algorithm**. For a d-chain Γ, $\Gamma[i] \in \mathbb{Z}_2$ denotes the coefficient of the i^{th} simplex in the chain Γ.

Algorithm 2: Total reduction algorithm

Inputs: A d-chain Γ, the reduced boundary matrix R
for $i \leftarrow m$ **to** 1 **do**
  if $\Gamma[i] \neq 0$ **and** $\exists j \in [1, n]$ *with* $\text{low}(j) = i$ *in* R **then**
    $\Gamma \leftarrow \Gamma + R_j$
  end
end

Proposition 3.5. *Algorithm 2 finds the total reduction of a given d-chain in $\mathcal{O}(m^2)$ time complexity.*

Proof. In Algorithm 2, let Γ_{i-1} be the value of the variable Γ after iteration i. Since the iteration counter i decreases from m to 1, the input and output of the algorithm are respectively Γ_m and Γ_0. At each iteration, Γ_{i-1} are either equal to Γ_i or $\Gamma_i + R_j$. Since $R_j \in \operatorname{Im} \partial_{d+1}$, Γ_{i-1} is in both cases homologous to Γ_i. Therefore, Γ_0 is homologous to Γ_m. We are left to show that Γ_0 is irreducible. From Lemma 3.2, it is enough to check that it does not contain any reducible simplex. Let σ_i be a reducible simplex and let us show that $\sigma_i \notin \Gamma_0$. Two possibilities may occur:

- if $\sigma_i \in \Gamma_i$, then $\Gamma_{i-1} = \Gamma_i + R_j$. Since $\text{low}(j) = i$, we have $\sigma_i \in R_j$ and therefore $\sigma_i \notin \Gamma_{i-1}$.
- if $\sigma_i \notin \Gamma_i$, then $\Gamma_{i-1} = \Gamma_i$ and again $\sigma_i \notin \Gamma_{i-1}$.

Furthermore, from iterations $i-1$ to 1, the added columns R_j contain only simplices smaller than σ_i and therefore $\sigma_i \notin \Gamma_{i-1} \Rightarrow \sigma_i \notin \Gamma_0$.

Observe that using an auxiliary array allows to compute the correspondence $low(j) \to i$ in time $\mathcal{O}(1)$. The column addition nested inside the loop lead to a $\mathcal{O}(m^2)$ time complexity for Algorithm 2. □

It follows that Problem 3.1 can be solved in $\mathcal{O}(mn^2)$ time complexity, by applying successively Algorithms 1 and 2, or in $\mathcal{O}(N^3)$ complexity if N is the size of the simplicial complex.

4 Lexicographic-minimal chain under imposed boundary

4.1 Problem statement

This section will study a variant of Lex-OHCP (Problem 4.1) in order to solve the subsequent problem of finding a minimal d-chain bounding a given $(d-1)$-cycle (Problem 4.2).

Problem 4.1. *Given a simplicial complex K with a total order on the d-simplices and a d-chain $\Gamma_0 \in \boldsymbol{C_d}(K)$, find :*

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \{\Gamma \in \boldsymbol{C_d}(K) \mid \partial_d \Gamma = \partial_d \Gamma_0\}$$

Problem 4.2. *Given a simplicial complex K with a total order on the d-simplices and a $(d-1)$-cycle A, check if A is a boundary:*

$$B_A \underset{def.}{=} \{\Gamma \in \boldsymbol{C_d}(K) \mid \partial_d \Gamma = A\} \neq \varnothing$$

If it is the case, find the minimal d-chain Γ bounded by A:

$$\Gamma_{min} = \min_{\sqsubseteq_{lex}} B_A$$

In Problem 4.2, finding a representative Γ_0 in the set $B_A \neq \varnothing$ such that $\partial_d \Gamma_0 = A$ is sufficient: we are then taken back to Problem 4.1 to find the minimal d-chain $\Gamma_{\min}$ such that $\partial_d \Gamma_{\min} = \partial_d \Gamma_0 = A$.

4.2 Boundary reduction transformation matrix

As in Section 3, we will derive an algorithmic solution to Problem 4.1 from the result of the boundary matrix reduction algorithm. Note that, unlike Section 3 that used the ∂_{d+1} boundary operator, we are now considering ∂_d, meaning the given total order on d-simplices applies to the greater dimension of the matrix. An arbitrary order can be taken for the $(d-1)$-simplices to build the matrix ∂_d. Indeed, if we see the performed reduction in matrix notation as $R = \partial_d \cdot V$, the minimization steps in this section will be performed on the transformation matrix V, whose rows do follow the given simplicial ordering. The number of zero columns of R is the dimension of $Z_d = \operatorname{Ker} \partial_d$ [22]. Let's denote it by $n^{\text{Ker}} = \dim(Z_d)$. By selecting all columns in V corresponding to zero columns in R, we obtain the matrix V^{Ker}, whose columns $V_1^{\text{Ker}}, \ldots, V_{n^{\text{Ker}}}^{\text{Ker}}$ form a basis of Z_d. We first show a useful property on the matrix V^{Ker}. Note that the low index for any column in V^{Ker} is well defined, as V is invertible.

Lemma 4.1. *Indexes $\left\{ \operatorname{low}(V_i^{\text{Ker}}) \right\}_{i \in [1, n^{\text{Ker}}]}$ are unique:*

$$i \neq j \Rightarrow \operatorname{low}(V_i^{\text{Ker}}) \neq \operatorname{low}(V_j^{\text{Ker}})$$

If $A \in \operatorname{Ker} \partial_d \setminus \{0\}$, there exists a unique column $V_{\max}^{\text{Ker}}$ of V^{Ker} with $\operatorname{low}(V_{\max}^{\text{Ker}}) = \operatorname{low}(A)$.

Proof. Before the boundary matrix reduction algorithm, the initial matrix V is the identity: the low indexes are therefore unique. During iterations of the algorithm, the matrix V is right-multiplied by an column-adding elementary matrix $L_{j_0,j}$, adding column j_0 to j with $j_0 < j$.

$$L_{j_0,j} = \begin{array}{cc} \begin{matrix} & & & j & & \end{matrix} & \\ \begin{bmatrix} 1 & & & & & \\ & 1 & & 1 & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} & \begin{matrix} \\ j_0 \\ \\ \\ \\ \\ \end{matrix} \end{array}$$

Therefore, the indexes $\{\mathrm{low}(V_i), V_i \in V\}$ stay on the diagonal during the reduction algorithm and are therefore unique. The restriction of V to V^{Ker} does not change this property.

If $A \in \mathrm{Ker}\,\partial_d \setminus \{0\}$, A can be written as a non-zero linear combination of columns $(V_i^{\mathrm{Ker}})_{i\in I}$ of V^{Ker}. Call $i_{\max} = \mathrm{low}(A)$ the index of the largest element $\sigma_{i_{\max}}$ in A. Suppose no column of $(V_i^{\mathrm{Ker}})_{i\in I}$ has $i_{\max}$ as its low index. By existence of $\sigma_{i_{\max}}$ in A, there is an odd number of columns $V_j^{\mathrm{Ker}} \in (V_i^{\mathrm{Ker}})_{i\in I}$ satisfying $\sigma_{i_{\max}} \in V_j^{\mathrm{Ker}}$ with $\mathrm{low}(V_j^{\mathrm{Ker}}) > i_{\max}$. We have shown however that the lows of V^{Ker} are unique, which implies the lows of columns V_j^{Ker} would appear in A: this contradicts the definition of $i_{\max}$ as the low of A. $\square$

4.3 Total reduction with imposed boundary

We apply a similar total reduction algorithm as previously introduced in Section 3 but using the matrix V^{Ker}. In the following algorithm, $m = \dim \boldsymbol{C_d}(K)$.

Algorithm 3: Total reduction variant

Inputs : A d-chain Γ and V^{Ker}
for $i \leftarrow m$ **to** 1 **do**
  if $\Gamma[i] \neq 0$ **and** $\exists j \in [1, n^{\mathrm{Ker}}]$ *with* $\mathrm{low}(j) = i$ *in* V^{Ker} **then**
    $\Gamma \leftarrow \Gamma + V_j^{\mathrm{Ker}}$
  end
end

Proposition 4.2. *Algorithm 3 computes the solution for Problem 4.1 in $\mathcal{O}(m^2)$ time complexity.*

Proof. The proof is similar to the one of Proposition 3.5.
In Algorithm 3, we denote by Γ_{i-1} the value of variable Γ after iteration i. Since iteration counter i is decreasing from m to 1, the input and output of the algorithm are respectively Γ_m and Γ_0. Since $V_j^{\mathrm{Ker}} \in \mathrm{Ker}\,\partial_d$, at each iteration $\partial\Gamma_{i-1} = \partial\Gamma_i$ therefore $\partial\Gamma_0 = \partial\Gamma_m$. We are left to show the algorithm's result is the minimal solution.

Suppose there is $\Gamma^\star$ such that $\partial_d\Gamma^\star = \partial\Gamma$ and $\Gamma^\star \sqsubset_{lex} \Gamma_0$. Let's consider the difference $\Gamma_0 - \Gamma^\star$, and its largest element index $\mathrm{low}(\Gamma_0 - \Gamma^\star) = i$, with $\sigma_i \in \Gamma_0$ and $\sigma_i \notin \Gamma^\star$ by Definition 2.1 of the lexicographic order. As $\Gamma_0 - \Gamma^\star \in \mathrm{Ker}\,\partial_d \setminus \{0\}$, there has to be a column V_j^{Ker} in V^{Ker} where $\mathrm{low}\left(V_j^{\mathrm{Ker}}\right) = i$, from Lemma 4.1. Two possibilities may occur at iteration i:

- if $\sigma_i \in \Gamma_i$, then $\Gamma_{i-1} = \Gamma_i + V_j^{\mathrm{Ker}}$. Since $i = \mathrm{low}(j)$, we have $\sigma_i \in V_j^{\mathrm{Ker}}$ and therefore $\sigma_i \notin \Gamma_{i-1}$.
- if $\sigma_i \notin \Gamma_i$, then $\Gamma_{i-1} = \Gamma_i$ and again $\sigma_i \notin \Gamma_{i-1}$.

However, from iterations $i-1$ to 1, the added columns V_j^{Ker} contains only simplices with indices smaller than i and therefore we obtain $\sigma_i \notin \Gamma_{i-1} \Rightarrow \sigma_i \notin \Gamma_0$, a contradiction to the definition of σ_i as the largest element of $\Gamma^\star - \Gamma_0$. $\square$

4.4 Finding a representative of $\mathbf{B_A}$

As previously mentioned, solving Problem 4.2 requires deciding if the set B_A is empty and in case it is not empty, finding an element of the set B_A. Algorithm 3 can then be used to minimize this element under imposed boundary. In the following algorithm, $m = \dim \boldsymbol{C_{d-1}}(K)$ and $n = \dim \boldsymbol{C_d}(K)$.

Proposition 4.3. *Algorithm 4 decides if the set B_A is non-empty, and in that case, finds a representative Γ_0 such that $\partial\Gamma_0 = A$ in $\mathcal{O}(m^2)$ time complexity.*

Proof. We start by two trivial observations from the definition of a reduction. First, A is a boundary if and only if its total reduction is the null chain. Second, if a non-null chain is a boundary, then its greatest simplex is reducible.

Algorithm 4: Finding a representative

```
Inputs  : A (d − 1)-chain A, a boundary matrix R reduced by V
Γ_0 ← ∅
A_0 ← A
for i ← m to 1 do
    if A_0[i] ≠ 0 then
        if ∃j ∈ [1, n] with low(j) = i in R then
            A_0 ← A_0 + R_j
            Γ_0 ← Γ_0 + V_j
        else
            The set B_A is empty.
        end
    end
end
```

If, at iteration i, $A_0[i] \neq 0$, then σ_i is the greatest simplex in A_0. In the case R has no column R_j such that $low(j) = i$, σ_i is not reducible by Lemma 3.4 and therefore A_0 is not a boundary. Since A and A_0 differ by a boundary (added columns of R), A is not a boundary either. This means the set B_A is empty.
The main difference with the previous chain reduction is we keep track of the column operations in Γ_0. If the total reduction of A is null, we have found a linear combination $(R_j)_{j\in J}$ such that $A = \sum_{j\in J} R_j$. We have also computed Γ_0 as the sum of the corresponding columns in V: $\Gamma_0 = \sum_{j\in J} V_j$. As $R = \partial_d \cdot V$, we can now verify:

$$\partial_d \Gamma_0 = \partial_d \left(\sum_{j\in J} V_j \right) = \sum_{j\in J} R_j = A$$

□

5 Efficient algorithm for codimension 1 (dual graph)

In this section we focus on Problem 5.1, a specialization of Problem 3.1, namely when K is a subcomplex of a $(d+1)$-pseudomanifold.

5.1 Problem statement

Recall that a d-dimensional simplicial complex is said **pure** if it is of dimension d and any simplex has at least one coface of dimension d.

Definition 5.1. *A* ***d-pseudomanifold*** *is a pure d-dimensional simplicial complex for which each $(d-1)$-face has exactly two d-dimensional cofaces.*

Definition 5.2. *The* ***dual graph*** *of a d-pseudomanifold $\mathcal{M}$ is the graph whose vertices are in one-to-one correspondence with the d-simplices of $\mathcal{M}$ and whose edges are in one-to-one correspondence with $(d-1)$-simplices of $\mathcal{M}$: an edge e connects two vertices v_1 and v_2 of the graph if and only if e corresponds to the $(d-1)$-face with cofaces corresponding to v_1 and v_2.*

Definition 5.3. *A* ***strongly connected*** *d-pseudomanifold is a d-pseudomanifold whose dual graph is connected.*

Given a strongly connected $(d+1)$-pseudomanifold $\mathcal{M}$ and $\tau_1 \neq \tau_2$ two $(d+1)$-simplices of $\mathcal{M}$, we consider a special case of Problem 3.1 where $K = \mathcal{M} \setminus \{\tau_1, \tau_2\}$ and $A = \partial\tau_1$:

Problem 5.1. *Given a strongly connected $(d+1)$-pseudomanifold $\mathcal{M}$ with a total order on the d-simplices and two distinct $(d+1)$-simplices (τ_1, τ_2) of $\mathcal{M}$, find:*

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \left\{ \Gamma \in \boldsymbol{C_p}(\mathcal{M}) \mid \exists B \in \boldsymbol{C_{d+1}}(\mathcal{M} \setminus \{\tau_1, \tau_2\}), \Gamma - \partial\tau_1 = \partial B \right\}$$

Definition 5.4. *Seeing a graph G as a 1-dimensional simplicial complex, we define the* ***coboundary operator*** $\partial^0 : \boldsymbol{C_0}(G) \to \boldsymbol{C_1}(G)$ *as the linear operator defined by the transpose of the matrix of the boundary operator $\partial_1 : \boldsymbol{C_1}(G) \to \boldsymbol{C_0}(G)$ in the canonical basis of simplices.*[1]

For a given graph $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ and $\mathcal{E}$ respectively denote its vertex and edge sets. For a d-chain $\alpha \in \boldsymbol{C_d}(\mathcal{M})$ and a $(d+1)$-chain $\beta \in \boldsymbol{C_{d+1}}(\mathcal{M})$, $\tilde{\alpha}$ and $\tilde{\beta}$ denote their respective dual 1-chain and dual 0-chain in the dual graph $G(\mathcal{M})$ of $\mathcal{M}$. We easily see that:

Remark 5.5. *For a set of vertices $\mathcal{V}_0 \subset \mathcal{V}$, $\partial^0 \mathcal{V}_0$ is exactly the set of edges in $G = (\mathcal{V}, \mathcal{E})$ that connect vertices in $\mathcal{V}_0$ with vertices in $\mathcal{V} \setminus \mathcal{V}_0$.*

Remark 5.6. *Let $\mathcal{M}$ be a $(d+1)$-pseudomanifold. If $\alpha \in \boldsymbol{C_d}(\mathcal{M})$ and $\beta \in \boldsymbol{C_{d+1}}(\mathcal{M})$, then $\tilde{\alpha} = \partial^0 \tilde{\beta} \iff \alpha = \partial_{d+1}\beta$.*

5.2 Codimension 1 and Lexicographic Min Cut (LMC)

The order on d-simplices of a $(d+1)$-pseudomanifold $\mathcal{M}$ naturally defines a corresponding order on the edges of the dual graph: $\tau_1 < \tau_2 \iff \tilde{\tau}_1 < \tilde{\tau}_2$. This order extends similarly to a lexicographic order $\sqsubseteq_{lex}$ on sets of edges (or, equivalently, 1-chains) in the graph.

In what follows, we say a set of edges $\tilde{\Gamma}$ **disconnects** $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in the graph $(\mathcal{V}, \mathcal{E})$ if $\tilde{\tau}_1$ and $\tilde{\tau}_2$ are not in the same connected component of the graph $(\mathcal{V}, \mathcal{E} \setminus \tilde{\Gamma})$.

Given a graph with weighted edges and two vertices, the min-cut/max-flow problem [23, 31] consists in finding the minimum cut (i.e. set of edges) disconnecting the two vertices, where minimum is meant as minimal sum of weights of cut edges. We consider a similar problem where the minimum is meant in term of a lexicographic order: the Lexicographic Min Cut (LMC).

Problem 5.2 (LMC)**.** *Given a connected graph $G = (\mathcal{V}, \mathcal{E})$ with a total order on $\mathcal{E}$ and two vertices $\tilde{\tau}_1, \tilde{\tau}_2 \in \mathcal{V}$, find the set $\tilde{\Gamma}_{\text{LMC}} \subset \mathcal{E}$ minimal for the lexicographic order $\sqsubseteq_{lex}$, that disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in G.*

Proposition 5.7. $\Gamma_{\min}$ *is solution of Problem 5.1 if and only its dual 1-chain $\tilde{\Gamma}_{\min}$ is solution of Problem 5.2 on the dual graph $G(\mathcal{M})$ of $\mathcal{M}$ where $\tilde{\tau}_1$ and $\tilde{\tau}_2$ are respective dual vertices of τ_1 and τ_2.*

Proof. Problem 5.1 can be equivalently formulated as:

$$\Gamma_{\min} = \min_{\sqsubseteq_{lex}} \{\partial_{d+1}(\tau_1 + B) \mid B \in \boldsymbol{C_{d+1}}(\mathcal{M} \setminus \{\tau_1, \tau_2\})\} \tag{2}$$

Using Observation 5.6, we see that $\Gamma_{\min}$ is the minimum in Equation (2) if and only if its dual 1-chain $\tilde{\Gamma}_{\min}$ satisfies:

$$\tilde{\Gamma}_{\min} = \min_{\sqsubseteq_{lex}} \left\{\partial^0(\tilde{\tau}_1 + \tilde{B}) \mid \tilde{B} \subset \mathcal{V} \setminus \{\tilde{\tau}_1, \tilde{\tau}_2\}\right\} \tag{3}$$

Denoting $\tilde{\Gamma}_{\text{LMC}}$ the minimum of Problem 5.2, we need to show that $\tilde{\Gamma}_{\text{LMC}} = \tilde{\Gamma}_{\min}$.
As $\tilde{\Gamma}_{\text{LMC}}$ disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in $G = (\mathcal{V}, \mathcal{E})$, $\tilde{\tau}_2$ is not in the connected component of $\tilde{\tau}_1$ in $(\mathcal{V}, \mathcal{E} \setminus \tilde{\Gamma}_{\text{LMC}})$. We define $\tilde{B}$ as the connected component of $\tilde{\tau}_1$ in $(\mathcal{V}, \mathcal{E} \setminus \tilde{\Gamma}_{\text{LMC}})$ minus $\tilde{\tau}_1$. We have that $\tilde{B} \subset \mathcal{V} \setminus \{\tilde{\tau}_1, \tilde{\tau}_2\}$. Consider an edge $e \in \partial^0(\tilde{\tau}_1 + \tilde{B})$. From Observation 5.5, e connects a vertex $v_a \in \{\tilde{\tau}_1\} \cup \tilde{B}$ with a vertex $v_b \notin \{\tilde{\tau}_1\} \cup \tilde{B}$. From the definition of $\tilde{B}$, $\tilde{\Gamma}_{\text{LMC}}$ disconnects v_a and v_b in G, which in turn implies $e \in \tilde{\Gamma}_{\text{LMC}}$. We have therefore shown that $\partial^0(\tilde{\tau}_1 + \tilde{B}) \subset \tilde{\Gamma}_{\text{LMC}}$. Using Equation (3), we get:

$$\tilde{\Gamma}_{\min} \sqsubseteq_{lex} \partial^0(\tilde{\tau}_1 + \tilde{B}) \sqsubseteq_{lex} \tilde{\Gamma}_{\text{LMC}} \tag{4}$$

Now we claim that if there is a $\tilde{C} \subset \mathcal{V} \setminus \{\tilde{\tau}_1, \tilde{\tau}_2\}$ with $\tilde{\Gamma} = \partial^0(\tilde{\tau}_1 + \tilde{C})$, then $\tilde{\Gamma}$ disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in $(\mathcal{V}, \mathcal{E})$. Consider a path in G from $\tilde{\tau}_1$ to $\tilde{\tau}_2$. Let v_a be the last vertex of the path that belongs to $\{\tilde{\tau}_1\} \cup \tilde{C}$ and v_b the next vertex on the path (which exists since the $\tilde{\tau}_2$ is not in $\{\tilde{\tau}_1\} \cup \tilde{C}$). From Observation 5.5, we see that the edge (v_a, v_b) must belong to $\tilde{\Gamma} = \partial^0(\tilde{\tau}_1 + \tilde{C})$. We have shown that any path in G connecting $\tilde{\tau}_1$ and $\tilde{\tau}_2$ has to contain an edge in $\tilde{\Gamma}$ and the claim is proved.

[1] In order to avoid to introduce non essential formal definitions, the coboundary operator is defined over chains since, for finite simplicial complexes, the canonical inner product defines a natural bijection between chains and cochains.

In particular, the minimum $\tilde{\Gamma}_{\min}$ disconnects $\tilde{\tau}_1$ and $\tilde{\tau}_2$ in $(\mathcal{V}, \mathcal{E})$. As $\tilde{\Gamma}_{\text{LMC}}$ denotes the minimum of Problem 5.2, $\tilde{\Gamma}_{\text{LMC}} \sqsubseteq_{lex} \tilde{\Gamma}_{\min}$ which, together with Equation (4), gives us $\tilde{\Gamma}_{\text{LMC}} = \tilde{\Gamma}_{\min}$. We have therefore shown the minimum defined by Equation (3) coincides with the minimum defined in Problem 5.2. □

5.3 Algorithm for Lexicographic Min Cut

In light of the new problem equivalency, we will study an algorithm solving Problem 5.2. As we will only consider the dual graph for this section, we leave behind the dual chain notation: vertices $\tilde{\tau}_1$ and $\tilde{\tau}_2$ are replaced by α_1 and α_2, and the solution to the problem is simply noted Γ_{LMC}. The following lemma exposes a constructive property of the solution on subgraphs.

Lemma 5.8. *Consider Γ_{LMC} solution of Problem 5.2 for the graph $G = (\mathcal{V}, \mathcal{E})$ and $\alpha_1, \alpha_2 \in \mathcal{V}$. Let e_0 be an edge in $\mathcal{V} \times \mathcal{V}$ such that $e_0 < \min\{e \in \mathcal{E}\}$. Then:*

(a) The solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$ is either Γ_{LMC} or $\Gamma_{\text{LMC}} \cup \{e_0\}$.

(b) $\Gamma_{\text{LMC}} \cup \{e_0\}$ is solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$ if and only if α_1 and α_2 are connected in $(\mathcal{V}, \mathcal{E} \cup \{e_0\} \setminus \Gamma_{\text{LMC}})$.

Proof. Let's call Γ'_{LMC} the solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$. Since $\Gamma'_{\text{LMC}} \cap \mathcal{E}$ disconnects α_1 and α_2 in $(\mathcal{V}, \mathcal{E})$, one has $\Gamma_{\text{LMC}} \sqsubseteq_{lex} \Gamma'_{\text{LMC}}$. Since $\Gamma_{\text{LMC}} \cup \{e_0\}$ disconnects α_1 and α_2 in $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$, we also have $\Gamma'_{\text{LMC}} \sqsubseteq_{lex} \Gamma_{\text{LMC}} \cup \{e_0\}$. Therefore, $\Gamma_{\text{LMC}} \sqsubseteq_{lex} \Gamma'_{\text{LMC}} \sqsubseteq_{lex} \Gamma_{\text{LMC}} \cup \{e_0\}$.

As $e_0 < \min\{e \in \mathcal{E}\}$, there is no set in $\mathcal{E} \cup \{e_0\}$ strictly between Γ_{LMC} and $\Gamma_{\text{LMC}} \cup \{e_0\}$ for the lexicographic order. It follows that Γ'_{LMC} is either equal to Γ_{LMC} or $\Gamma_{\text{LMC}} \cup \{e_0\}$. The choice for Γ'_{LMC} is therefore ruled by the property that it should disconnect α_1 and α_2: if α_1 and α_2 are connected in $(\mathcal{V}, \mathcal{E} \cup \{e_0\} \setminus \Gamma_{\text{LMC}})$, Γ_{LMC} does not disconnect α_1 and α_2 in $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$ and $\Gamma_{\text{LMC}} \cup \{e_0\}$ has to be the solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$. On the other hand, if α_1 and α_2 are not connected in $(\mathcal{V}, \mathcal{E} \cup \{e_0\} \setminus \Gamma_{\text{LMC}})$, then both Γ_{LMC} and $\Gamma_{\text{LMC}} \cup \{e_0\}$ disconnect α_1 and α_2 in $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$, but as $\Gamma_{\text{LMC}} \sqsubset_{lex} \Gamma_{\text{LMC}} \cup \{e_0\}$, $\Gamma_{\text{LMC}} \cup \{e_0\}$ is not the solution for $(\mathcal{V}, \mathcal{E} \cup \{e_0\})$. □

Building an algorithm from Lemma 5.8 suggests a data structure able to check if vertices α_1 and α_2 are connected in the graph: the disjoint-set data structure, introduced for finding connected components [25], does exactly that. In this structure, each set of elements has a different root value, called representative. Calling the operation **MakeSet** on an element creates a new set containing this element. The **FindSet** operation, given an element of a set, returns the representative of the set. For all elements of the same set, **FindSet** will of course return the same representative. Finally, the structure allows merging two sets, by using the **UnionSet** operation. After this operation, elements of both sets will have the same representative.

We now describe Algorithm 5. The algorithm expects a set of edges sorted in decreasing order according to the lexicographic order.

Proposition 5.9. *Algorithm 5 computes the solution of Problem 5.2 for a given graph $(\mathcal{V}, \mathcal{E})$ and two vertices $\alpha_1, \alpha_2 \in \mathcal{V}$. Assuming the input set of edges $\mathcal{E}$ are sorted, the algorithm has $\mathcal{O}(n\alpha(n))$ time complexity, where n is the cardinal of $\mathcal{E}$ and α the inverse Ackermann function.*

Proof. We denote by e_i the i^{th} edge along the decreasing order and Γ^i_{LMC} the value of the variable Γ_{LMC} of the algorithm after iteration i. The algorithm works by incrementally adding edges in decreasing order and tracking the growing connected components of the set associated with α_1 and α_2 in $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\} \setminus \Gamma^i_{\text{LMC}})$, for $i = 1, \ldots, n$.

At the beginning, no edges are inserted, and $\Gamma^0_{\text{LMC}} = \varnothing$ is indeed solution for $(\mathcal{V}, \varnothing)$. With Lemma 5.8, we are guaranteed at each iteration i to find the solution for $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\})$ by only adding to $\Gamma^{i-1}_{\text{LMC}}$ the current edge e_i if α_1 and α_2 are connected in $\{e \in \mathcal{E}, e \geq e_i\} \setminus \Gamma^{i-1}_{\text{LMC}}$, which is done in the if-statement. If the edge is not added, we update the connectivity of the graph $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\} \setminus \Gamma^i_{\text{LMC}})$ by merging the two sets represented by r_1 and r_2. After each iteration, Γ^i_{LMC} is solution for $(\mathcal{V}, \{e \in \mathcal{E}, e \geq e_i\})$ and when all edges are processed, Γ^n_{LMC} is solution for $(\mathcal{V}, \mathcal{E})$.

The complexity of the **MakeSet**, **FindSet** and **UnionSet** operations have been shown to be respectively $\mathcal{O}(1)$, $\mathcal{O}(\alpha(v))$ and $\mathcal{O}(\alpha(v))$, where $\alpha(v)$ is the inverse Ackermann function on

Algorithm 5: Lexicographic Min Cut

```
Inputs : G = (V, E) and α1, α2 ∈ V, with E = {e_i, i = 1, ..., n} in decreasing order
Output: Γ_LMC
Γ_LMC ← ∅
for v ∈ V do
  | MakeSet(v)
end
for e ∈ E in decreasing order do
  | e = (v1, v2) ∈ V × V
  | r1 ← FindSet(v1), r2 ← FindSet(v2)
  | c1 ← FindSet(α1), c2 ← FindSet(α2)
  | if {r1, r2} = {c1, c2} then
  |   | Γ_LMC ← Γ_LMC ∪ e
  | else
  |   | UnionSet(r1, r2)
  | end
end
```

the cardinal of the vertex set [32]. Assuming sorted edges as input of the algorithm – which is performed in $\mathcal{O}(n\ln(n))$, the algorithm runs in $\mathcal{O}(n\alpha(n))$ time complexity. □

6 Application to point cloud triangulation

In all that precedes, the order on simplices was not specified and one can wonder if choosing such an ordering makes the specialization of OCHP too restrictive for it to be useful. In this section, we give a concrete example where this restriction makes sense and provides a simple and elegant application to the problem of point cloud triangulation. Whereas all that preceded dealt with an abstract simplicial complex, we now consider a bijection between vertices and a set of points in Euclidean space, allowing to compute geometric quantities on simplices.

6.1 Simplicial ordering

Recent works have studied a characterization of the 2D Delaunay triangulation as a lexicographic minimum over 2-chains. Denote by $\mathrm{R_B}(\sigma)$ the radius of the smallest enclosing ball and $\mathrm{R_C}(\sigma)$ the radius of the circumcircle of a 2-simplex σ. Based on [16, 15], we define the total order on 2-simplices:

$$\sigma_1 \leq \sigma_2 \iff \begin{cases} \mathrm{R_B}(\sigma_1) < \mathrm{R_B}(\sigma_2) \\ \text{or} \\ \mathrm{R_B}(\sigma_1) = \mathrm{R_B}(\sigma_2) \quad \text{and} \quad \mathrm{R_C}(\sigma_1) \geq \mathrm{R_C}(\sigma_2) \end{cases} \tag{5}$$

Under generic condition on the position of points, we can show this order is total. In what follows, the lexicographic order $\sqsubseteq_{lex}$ is induced by this order on simplices. The following proposition from [15] shows a strong link between the simplex ordering and the 2D Delaunay triangulation.

Proposition 6.1 (Proposition 7.9 in [15])**.** *Let $\mathbf{P} = \{P_1, \ldots, P_N\} \subset \mathbb{R}^2$ with $N \geq 3$ be in general position and let $K_\mathbf{P}$ be the 2-dimensional full complex over $\mathbf{P}$. Denote by $\beta_\mathbf{P} \in C_1(K_\mathbf{P})$ the 1-chain made of edges belonging to the boundary of $\mathcal{CH}(\mathbf{P})$. If $\Gamma_{\min} = \min_{\sqsubseteq_{lex}}\{\Gamma \in \boldsymbol{C_2}(K_\mathbf{P}), \partial\Gamma = \beta_\mathbf{P}\}$, the simplicial complex $|\Gamma_{\min}|$ support of $\Gamma_{\min}$ is the Delaunay triangulation of $\mathbf{P}$.*

As the 2D Delaunay triangulation has some well-known optimality properties, such as maximizing the minimal angle, we can hope that using the same order to minimize 2-chains in dimension 3 will keep some of those properties. In fact, it has been shown that for a Čech or Vietoris-Rips complex, under strict conditions linking the point set sampling, the parameter of the complex and the reach of the underlying manifold of Euclidean space, the minimal lexicographic chain using

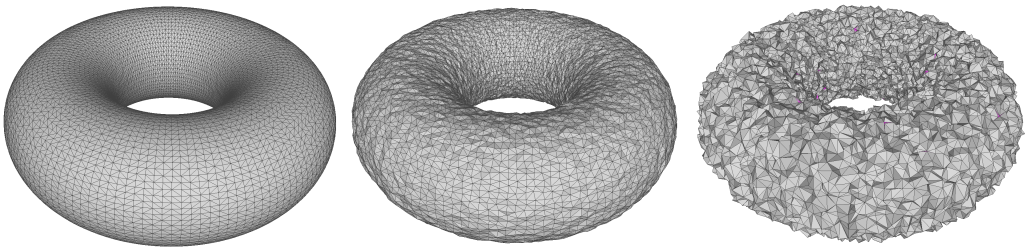

Figure 1: Watertight reconstructions under different perturbations. Under small perturbations (first two images from the left), the reconstruction is a triangulation of the sampled manifold. A few non-manifold configurations appear however under larger perturbations (Rightmost image).

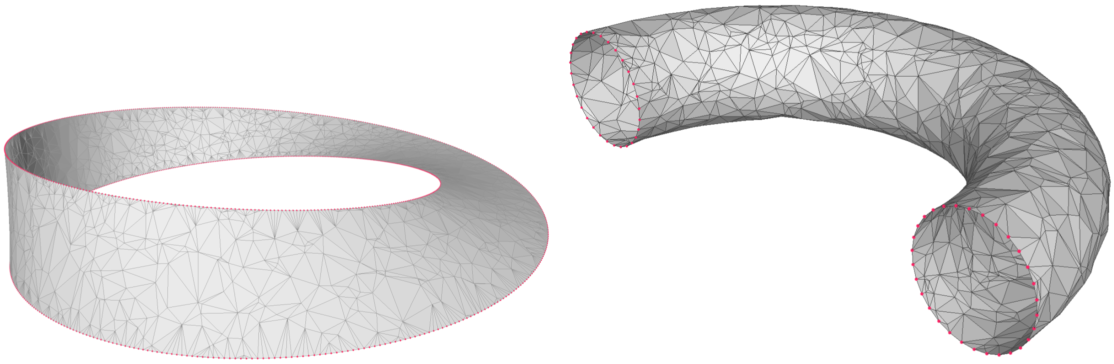

Figure 2: Open surface triangulations under imposed boundaries (red cycles).

the described simplex order is a triangulation of the sampled manifold [15]. Experimental results (Figure 1) show that this property remains true relatively far from these theoretical conditions.

6.2 Open surface triangulation

Using the Phat library [5], we generate a Čech complex of the point cloud and the points of a provided cycle, with a sufficient parameter to capture the topology of the object [9, 4]. After constructing the 2-boundary, we apply the boundary reduction algorithm, slightly modified to calculate as well the transformation matrix V. We then apply Algorithm 4, and in the case the cycle is a boundary, we get a chain bounded by the provided cycle. We then apply Algorithm 3 to minimize the chain under imposed boundary. Figure 2 shows results of this method.

6.3 Closed surface triangulation

Using Algorithm 5 requires a strongly connected 3-pseudomanifold: we therefore use a 3D Delaunay triangulation, for its efficiency and non-parametric nature, using the CGAL library [26], and complete it into a 3-sphere by connecting, for any triangle on the convex hull of the Delaunay triangulation, its dual edge to an "infinite" dual vertex.

Experimentally, sorting triangles does not require exact predicates: the $\mathrm{R_B}$ and $\mathrm{R_C}$ quantities can simply be calculated in fixed precision. The quasilinear complexity of Algorithm 5 makes it competitive in large point cloud applications (Figure 3). Outliers are naturally ignored and, being parameter free, the algorithm adapts to non uniform point densities.

The choice of α_1 and α_2 defines the location of the closed separating surface. We can guide the algorithm by interactively adding multiple α_1 and α_2 regions as depicted in Figure 4. Algorithm 5 requires to be slightly modified to take as input multiple α_1, α_2: after creating all sets with MakeSet, we need to combine all α_1 sets together, and all α_2 sets together. The algorithm remains unchanged for the rest.

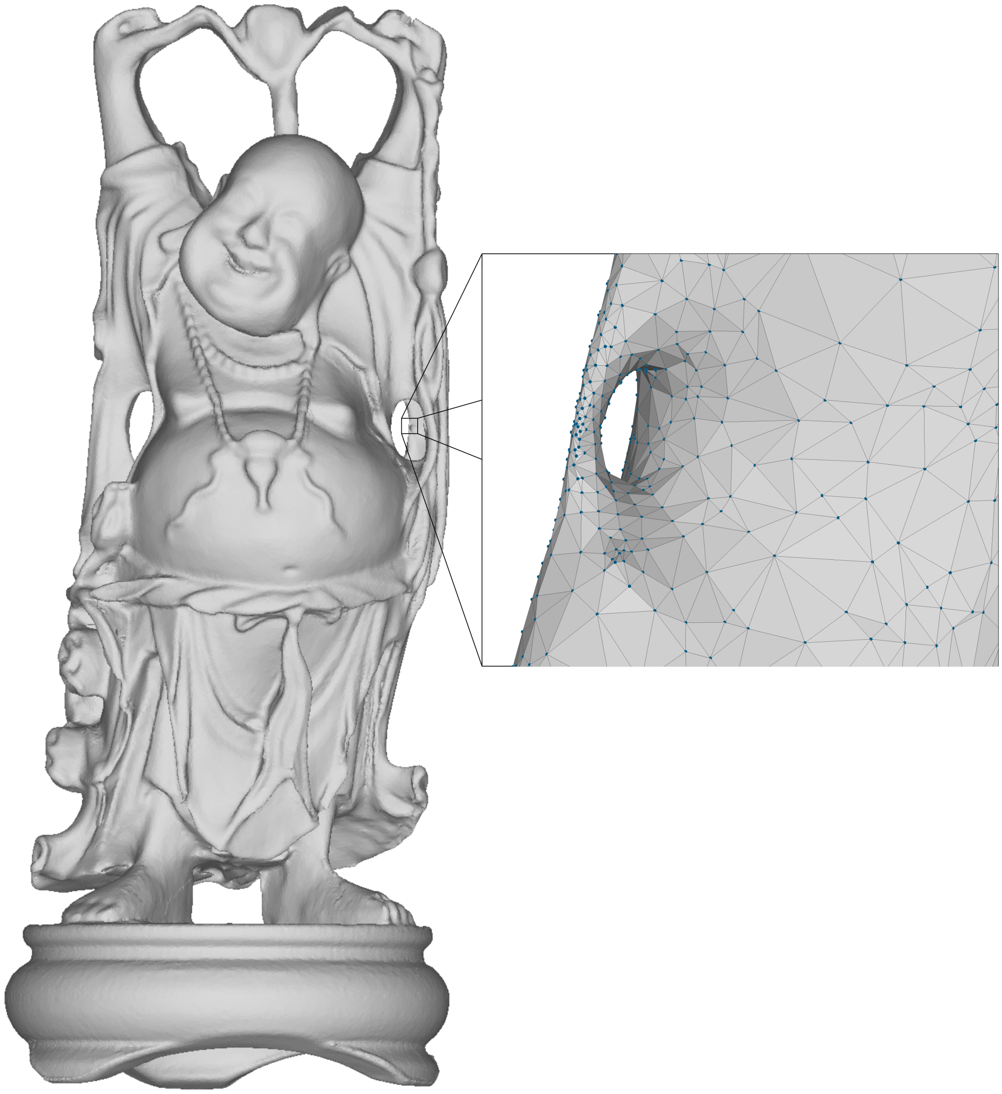

Figure 3: Closed surface triangulation of 440K points in 7.33 seconds. Beside the point cloud, the only user input is one inner tetrahedron.

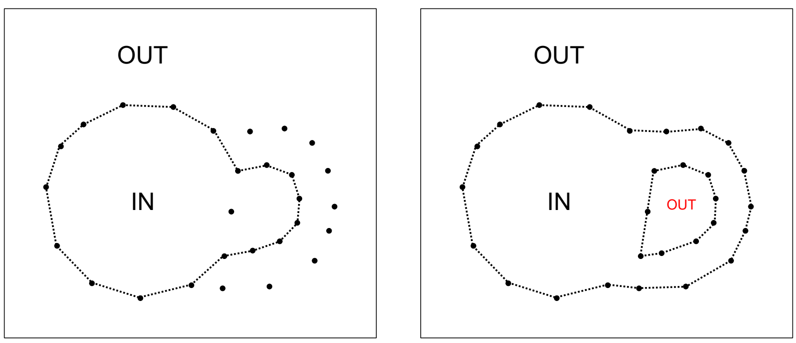

Figure 4: Providing additional topological information can improve the result of the reconstruction. Here the lexicographic order on 1-chains is induced by edge length comparison.

References

[1] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph.*, 24(3):617–625, 2005. URL: https://doi.org/10.1145/1073204.1073238, doi:10.1145/1073204.1073238.

[2] Nina Amenta, Marshall W. Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, Orlando, FL, USA, July 19-24, 1998*, pages 415–421, 1998. URL: https://doi.org/10.1145/280814.280947, doi:10.1145/280814.280947.

[3] Nina Amenta, Sunghee Choi, Tamal K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Int. J. Comput. Geometry Appl.*, 12(1-2):125–141, 2002. URL: https://doi.org/10.1142/S0218195902000773, doi:10.1142/S0218195902000773.

[4] Dominique Attali, André Lieutier, and David Salinas. Vietoris-rips complexes also provide topologically correct reconstructions of sampled shapes. *Comput. Geom.*, 46(4):448–465, 2013. URL: https://doi.org/10.1016/j.comgeo.2012.02.009, doi:10.1016/j.comgeo.2012.02.009.

[5] Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76 – 90, 2017. Algorithms and Software for Computational Topology. doi:10.1016/j.jsc.2016.03.008.

[6] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984. URL: https://doi.org/10.1145/357346.357349, doi:10.1145/357346.357349.

[7] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry, Clear Water Bay, Hong Kong, China, June 12-14, 2000*, pages 223–232, 2000. URL: https://doi.org/10.1145/336154.336208, doi:10.1145/336154.336208.

[8] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, pages 377–385, 2009. URL: https://doi.org/10.1145/1542362.1542426, doi:10.1145/1542362.1542426.

[9] Frédéric Chazal, David Cohen-Steiner, and André Lieutier. A sampling theory for compact sets in euclidean space. *Discrete & Computational Geometry*, 41(3):461–479, 2009. URL: https://doi.org/10.1007/s00454-009-9144-8, doi:10.1007/s00454-009-9144-8.

[10] Chao Chen and Daniel Freedman. Quantifying homology classes. *CoRR*, abs/0802.2865, 2008. arXiv:0802.2865.

[11] Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Comput. Geom.*, 43(2):169–181, 2010. URL: https://doi.org/10.1016/j.comgeo.2009.06.004, doi:10.1016/j.comgeo.2009.06.004.

[12] Chao Chen and Daniel Freedman. Hardness results for homology localization. *Discrete & Computational Geometry*, 45(3):425–448, 2011. URL: https://doi.org/10.1007/s00454-010-9322-8, doi:10.1007/s00454-010-9322-8.

[13] Long Chen. Mesh smoothing schemes based on optimal delaunay triangulations. In *Proceedings of the 13th International Meshing Roundtable, IMR 2004, Williamsburg, Virginia, USA, September 19-22, 2004*, pages 109–120, 2004. URL: http://imr.sandia.gov/papers/abstracts/Ch317.html.

[14] Long Chen and Michael Holst. Efficient mesh optimization schemes based on optimal delaunay triangulations. *Computer Methods in Applied Mechanics and Engineering*, 200(9):967–984, 2011. doi:https://doi.org/10.1016/j.cma.2010.11.007.

[15] David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Lexicographic optimal chains and manifold triangulations. 2019.

[16] David Cohen-Steiner, André Lieutier, and Julien Vuillamy. Regular triangulations as lexicographic optimal chains. 2019.

[17] Tamal K. Dey and Samrat Goswami. Tight cocone: A water-tight surface reconstructor. *J. Comput. Inf. Sci. Eng.*, 3(4):302–307, 2003.

[18] Tamal K. Dey, Anil N. Hirani, and Bala Krishnamoorthy. Optimal homologous cycles, total unimodularity, and linear programming. *SIAM J. Comput.*, 40(4):1026–1044, 2011. URL: https://doi.org/10.1137/100800245, doi:10.1137/100800245.

[19] Tamal K. Dey, Tao Hou, and Sayan Mandal. Computing minimal persistent cycles: Polynomial and hard cases. *CoRR*, abs/1907.04889, 2019. URL: http://arxiv.org/abs/1907.04889, arXiv:1907.04889.

[20] Tamal K. Dey, Jian Sun, and Yusu Wang. Approximating loops in a shortest homology basis from point data. In David G. Kirkpatrick and Joseph S. B. Mitchell, editors, *Proceedings of the 26th ACM Symposium on Computational Geometry, Snowbird, Utah, USA, June 13-16, 2010*, pages 166–175. ACM, 2010. URL: https://doi.org/10.1145/1810959.1810989, doi:10.1145/1810959.1810989.

[21] Herbert Edelsbrunner. *Surface Reconstruction by Wrapping Finite Sets in Space*, pages 379–404. Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-642-55566-4_17.

[22] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

[23] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972. URL: https://doi.org/10.1145/321694.321699, doi:10.1145/321694.321699.

[24] Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 1038–1046, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070581.

[25] Bernard A. Galler and Michael J. Fischer. An improved equivalence algorithm. *Commun. ACM*, 7(5):301–303, 1964. URL: https://doi.org/10.1145/364099.364331, doi:10.1145/364099.364331.

[26] Clément Jamin, Sylvain Pion, and Monique Teillaud. 3D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edition, 2019. URL: https://doc.cgal.org/5.0/Manual/packages.html#PkgTriangulation3.

[27] Michael M. Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, 2013. URL: https://doi.org/10.1145/2487228.2487237, doi:10.1145/2487228.2487237.

[28] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. *Comput. Graph. Forum*, 28(8):2275–2290, 2009. doi:10.1111/j.1467-8659.2009.01530.x.

[29] Sylvain Lefebvre and Michela Spagnuolo, editors. *Eurographics 2014 - State of the Art Reports, Strasbourg, France, April 7-11, 2014*. Eurographics Association, 2014. URL: https://diglib.eg.org/handle/10.2312/7707.

[30] Yangyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4):52, 2011. URL: https://doi.org/10.1145/2010324.1964947, doi:10.1145/2010324.1964947.

[31] James B. Orlin. Max flows in o(nm) time, or better. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 765–774, 2013. URL: https://doi.org/10.1145/2488608.2488705, doi:10.1145/2488608.2488705.

[32] Robert Endre Tarjan and Jan van Leeuwen. Worst-case analysis of set union algorithms. *J. ACM*, 31(2):245–281, 1984. URL: https://doi.org/10.1145/62.2160, doi:10.1145/62.2160.