Reducing complexes in Multidimensional Persistence

Claudia Landi

University of Modena and Reggio Emilia

joint work with M. Allili, T. Kaczynski and F. Masoni

CAT 2015

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]
- Computation of multi-D persistent homology has polynomial complexity for one-critical filtrations [Carlsson et al 2010]

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]
- Computation of multi-D persistent homology has polynomial complexity for one-critical filtrations [Carlsson et al 2010]
- We know of no way to improve the worst case complexity of the problem. For massive datasets, this can be a severe limitation.

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]
- Computation of multi-D persistent homology has polynomial complexity for one-critical filtrations [Carlsson et al 2010]
- We know of no way to improve the worst case complexity of the problem. For massive datasets, this can be a severe limitation.
- An alternative strategy:
  - reduce the initial complex using geometric and combinatorial methods before computing persistence
  - use reductions that preserve persistent homology groups

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]
- Computation of multi-D persistent homology has polynomial complexity for one-critical filtrations [Carlsson et al 2010]
- We know of no way to improve the worst case complexity of the problem. For massive datasets, this can be a severe limitation.
- An alternative strategy:
  - reduce the initial complex using geometric and combinatorial methods before computing persistence
  - use reductions that preserve persistent homology groups
  - Case of 1D persistence in homology degree 0: [Frosini-Pittore 1999]

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]
- Computation of multi-D persistent homology has polynomial complexity for one-critical filtrations [Carlsson et al 2010]
- We know of no way to improve the worst case complexity of the problem. For massive datasets, this can be a severe limitation.
- An alternative strategy:
  - reduce the initial complex using geometric and combinatorial methods before computing persistence
  - use reductions that preserve persistent homology groups
  - Case of 1D persistence in homology degree 0: [Frosini-Pittore 1999]
  - Case of multiD persistence in homology degree 0: [Cerri et al 2006]:

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]
- Computation of multi-D persistent homology has polynomial complexity for one-critical filtrations [Carlsson et al 2010]
- We know of no way to improve the worst case complexity of the problem. For massive datasets, this can be a severe limitation.
- An alternative strategy:
  - reduce the initial complex using geometric and combinatorial methods before computing persistence
  - use reductions that preserve persistent homology groups
  - Case of 1D persistence in homology degree 0: [Frosini-Pittore 1999]
  - Case of multiD persistence in homology degree 0: [Cerri et al 2006]:
  - Case of 1D persistence in any degree: [Nanda-Mischaikow 2013]

Motivation

- The most common algorithm used for computing 1-D persistent homology has complexity $O(n^3)$ [Zomorodian-Carlsson 2005]
- Computation of multi-D persistent homology has polynomial complexity for one-critical filtrations [Carlsson et al 2010]
- We know of no way to improve the worst case complexity of the problem. For massive datasets, this can be a severe limitation.
- An alternative strategy:
  - reduce the initial complex using geometric and combinatorial methods before computing persistence
  - use reductions that preserve persistent homology groups
  - Case of 1D persistence in homology degree 0: [Frosini-Pittore 1999]
  - Case of multiD persistence in homology degree 0: [Cerri et al 2006]:
  - Case of 1D persistence in any degree: [Nanda-Mischaikow 2013]
- This talk: apply this strategy for multiD persistence in any degree

Outline

Introduction
- Lefschetz Complexes
- Combinatorial Morse theory
- Multidimensional Persistent Homology

Outline

Introduction
- Lefschetz Complexes
- Combinatorial Morse theory
- Multidimensional Persistent Homology

Combinatorial Morse theory in the framework of persistence

Outline

Introduction
- Lefschetz Complexes
- Combinatorial Morse theory
- Multidimensional Persistent Homology

Combinatorial Morse theory in the framework of persistence

A matching algorithm for multidimensional persistence

Outline

Introduction
- Lefschetz Complexes
- Combinatorial Morse theory
- Multidimensional Persistent Homology

Combinatorial Morse theory in the framework of persistence

A matching algorithm for multidimensional persistence

Numerical tests

Lefschetz Complexes

1. S a finite set with a gradation S_q such that:
   - $S_q = \emptyset$ for $q < 0$
   - For every $\sigma \in S$ there exists a unique q, called dimension, s. t. $\sigma \in S_q$.

Lefschetz Complexes

1. S a finite set with a gradation S_q such that:
   - $S_q = \emptyset$ for $q < 0$
   - For every $\sigma \in S$ there exists a unique q, called dimension, s. t. $\sigma \in S_q$.
2. An *incidence function* $\kappa : S \times S \to R$, R a PID, such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$.

Lefschetz Complexes

1. S a finite set with a gradation S_q such that:
   - $S_q = \emptyset$ for $q < 0$
   - For every $\sigma \in S$ there exists a unique q, called dimension, s. t. $\sigma \in S_q$.
2. An *incidence function* $\kappa : S \times S \to R$, R a PID, such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$.
3. $C_q(S) := R(S_q)$, the free module over R generated by S_q.

Lefschetz Complexes

1. S a finite set with a gradation S_q such that:
   - $S_q = \emptyset$ for $q < 0$
   - For every $\sigma \in S$ there exists a unique q, called dimension, s. t. $\sigma \in S_q$.
2. An *incidence function* $\kappa : S \times S \to R$, R a PID, such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$.
3. $C_q(S) := R(S_q)$, the free module over R generated by S_q.
4. (S, κ) is a *Lefschetz complex* if $(C_*(S), \partial^\kappa_*)$ with $\partial^\kappa_q : C_q(S) \to C_{q-1}(S)$ defined on generators $\sigma \in S$ by

$$\partial^\kappa(\sigma) := \sum_{\tau \in S} \kappa(\sigma, \tau)\tau$$

   is a free chain complex with base S.

Lefschetz Complexes

1. S a finite set with a gradation S_q such that:
   - $S_q = \emptyset$ for $q < 0$
   - For every $\sigma \in S$ there exists a unique q, called dimension, s. t. $\sigma \in S_q$.
2. An *incidence function* $\kappa : S \times S \to R$, R a PID, such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$.
3. $C_q(S) := R(S_q)$, the free module over R generated by S_q.
4. (S, κ) is a *Lefschetz complex* if $(C_*(S), \partial^\kappa_*)$ with $\partial^\kappa_q : C_q(S) \to C_{q-1}(S)$ defined on generators $\sigma \in S$ by

$$\partial^\kappa(\sigma) := \sum_{\tau \in S} \kappa(\sigma, \tau)\tau$$

   is a free chain complex with base S.
5. The homology of a Lefschetz complex (S, κ) is the homology of $(C_*(S), \partial^\kappa_*)$.

Partial matchings

A *partial matching* on $(\mathtt{S}, \kappa)$ is a quadruplet $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ where

- $\mathtt{A}, \mathtt{B}, \mathtt{C}$ is a partition of $\mathtt{S}$,
- $\mathtt{m} : \mathtt{A} \to \mathtt{B}$ is a map such that, for each $\tau \in \mathtt{A}$, $\kappa(\mathtt{m}(\tau), \tau)$ is invertible.

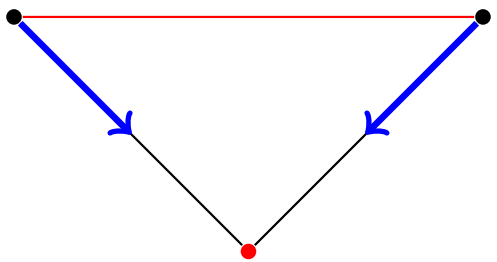
Partial matchings

A *partial matching* on $(\mathtt{S}, \kappa)$ is a quadruplet $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ where

- $\mathtt{A}, \mathtt{B}, \mathtt{C}$ is a partition of $\mathtt{S}$,
- $\mathtt{m} : \mathtt{A} \to \mathtt{B}$ is a map such that, for each $\tau \in \mathtt{A}$, $\kappa(\mathtt{m}(\tau), \tau)$ is invertible.

$(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ is conveniently represented by arrows:

- For all pairs $(\mathtt{m}(\tau), \tau)$, draw an arrow from τ to $\mathtt{m}(\tau)$



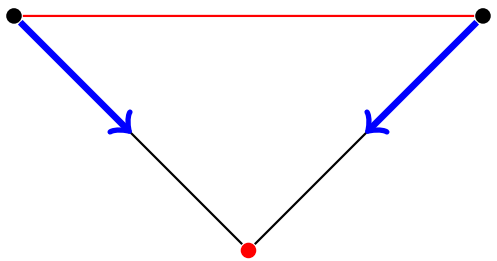
Partial matchings

A *partial matching* on $(\mathtt{S}, \kappa)$ is a quadruplet $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ where

- $\mathtt{A}, \mathtt{B}, \mathtt{C}$ is a partition of $\mathtt{S}$,
- $\mathtt{m} : \mathtt{A} \to \mathtt{B}$ is a map such that, for each $\tau \in \mathtt{A}$, $\kappa(\mathtt{m}(\tau), \tau)$ is invertible.

$(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ is conveniently represented by arrows:

- For all pairs $(\mathtt{m}(\tau), \tau)$, draw an arrow from τ to $\mathtt{m}(\tau)$



- A partial matching is called *acyclic* if there is no non-trivial sequence of simplices

$$\sigma_0 \xrightarrow{\mathtt{m}} \tau_0 \xrightarrow{>} \sigma_1 \xrightarrow{\mathtt{m}} \tau_1 \xrightarrow{>} \ldots \xrightarrow{>} \sigma_n \xrightarrow{\mathtt{m}} \tau_n \xrightarrow{>} \sigma_0$$

One-step reductions [KMS 1998]

Let $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ be a partial matching (not necessarily acyclic) on $(\mathtt{S}, \kappa)$.

One-step reductions [KMS 1998]

Let $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathtt{m})$ be a partial matching (not necessarily acyclic) on (S, κ). A single pair $(\mathtt{m}(\sigma), \sigma)$ can be removed so to obtain again a Lefschetz complex:

- For $\sigma \in \mathsf{A}$, define $(\overline{\mathsf{S}}, \overline{\kappa})$ where $\overline{\mathsf{S}} = \mathsf{S} \setminus \{\mathtt{m}(\sigma), \sigma\}$, and $\overline{\kappa} : \overline{\mathsf{S}} \times \overline{\mathsf{S}} \to R$,

$$\overline{\kappa}(\eta, \xi) = \kappa(\eta, \xi) - \frac{\kappa(\eta, \sigma)\kappa(\mathtt{m}(\sigma), \xi)}{\kappa(\mathtt{m}(\sigma), \sigma)}.$$

- $(\overline{\mathsf{S}}, \overline{\kappa})$ is a Lefschetz complex.

One-step reductions [KMS 1998]

Let $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{m})$ be a partial matching (not necessarily acyclic) on (S, κ). A single pair $(\mathsf{m}(\sigma), \sigma)$ can be removed so to obtain again a Lefschetz complex:

- For $\sigma \in \mathsf{A}$, define $(\overline{\mathsf{S}}, \overline{\kappa})$ where $\overline{\mathsf{S}} = \mathsf{S} \setminus \{\mathsf{m}(\sigma), \sigma\}$, and $\overline{\kappa} : \overline{\mathsf{S}} \times \overline{\mathsf{S}} \to R$,

$$\overline{\kappa}(\eta, \xi) = \kappa(\eta, \xi) - \frac{\kappa(\eta, \sigma)\kappa(\mathsf{m}(\sigma), \xi)}{\kappa(\mathsf{m}(\sigma), \sigma)}.$$

- $(\overline{\mathsf{S}}, \overline{\kappa})$ is a Lefschetz complex.
- Define $\pi : C_*(\mathsf{S}) \to C_*(\overline{\mathsf{S}})$ and $\iota : C_*(\overline{\mathsf{S}}) \to C_*(\mathsf{S})$ on generators by setting

$$\pi(\tau) = \begin{cases} 0 & \text{if } \tau = \mathsf{m}(\sigma) \\ -\sum_{\xi \in \overline{\mathsf{S}}} \frac{\kappa(\mathsf{m}(\sigma), \xi)}{\kappa(\mathsf{m}(\sigma), \sigma)} \xi & \text{if } \tau = \sigma \\ \tau & \text{otherwise} \end{cases}$$

and

$$\iota(\tau) = \tau - \frac{\kappa(\tau, \sigma)}{\kappa(\mathsf{m}(\sigma), \sigma)} \mathsf{m}(\sigma).$$

One-step reductions [KMS 1998]

Let $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{m})$ be a partial matching (not necessarily acyclic) on (S, κ). A single pair $(\mathsf{m}(\sigma), \sigma)$ can be removed so to obtain again a Lefschetz complex:

- For $\sigma \in \mathsf{A}$, define $(\overline{\mathsf{S}}, \overline{\kappa})$ where $\overline{\mathsf{S}} = \mathsf{S} \setminus \{\mathsf{m}(\sigma), \sigma\}$, and $\overline{\kappa} : \overline{\mathsf{S}} \times \overline{\mathsf{S}} \to R$,

$$\overline{\kappa}(\eta, \xi) = \kappa(\eta, \xi) - \frac{\kappa(\eta, \sigma)\kappa(\mathsf{m}(\sigma), \xi)}{\kappa(\mathsf{m}(\sigma), \sigma)}.$$

- $(\overline{\mathsf{S}}, \overline{\kappa})$ is a Lefschetz complex.
- Define $\pi : C_*(\mathsf{S}) \to C_*(\overline{\mathsf{S}})$ and $\iota : C_*(\overline{\mathsf{S}}) \to C_*(\mathsf{S})$ on generators by setting

$$\pi(\tau) = \begin{cases} 0 & \text{if } \tau = \mathsf{m}(\sigma) \\ -\sum_{\xi \in \overline{\mathsf{S}}} \frac{\kappa(\mathsf{m}(\sigma), \xi)}{\kappa(\mathsf{m}(\sigma), \sigma)} \xi & \text{if } \tau = \sigma \\ \tau & \text{otherwise} \end{cases}$$

and

$$\iota(\tau) = \tau - \frac{\kappa(\tau, \sigma)}{\kappa(\mathsf{m}(\sigma), \sigma)} \mathsf{m}(\sigma).$$

- π and ι are chain equivalences

Multi-filtration of a Lefschetz complex

- In $\mathbb{R}^k$ consider the partial order $a = (a_i) \preceq b = (b_i)$ if and only if $a_i \leq b_i$ for all $i = 1, 2, \ldots, k$;

Multi-filtration of a Lefschetz complex

- In $\mathbb{R}^k$ consider the partial order $a = (a_i) \preceq b = (b_i)$ if and only if $a_i \leq b_i$ for all $i = 1, 2, \ldots, k$;
- A *k-filtration* of S is a family $\mathcal{F} = \{S^a\}_{a \in \mathbb{R}^k}$ of subsets of S with the following properties:
  - $\mathcal{F}$ is nested with respect to inclusions:

$$S^a \subseteq S^b, \text{ for every } a \preceq b$$

  - $\mathcal{F}$ is non-increasing on faces:

$$\text{if } \sigma \in S^a \text{ and } \tau \text{ is a face of } \sigma \text{ then } \tau \in S^a$$

Multi-filtration of a Lefschetz complex

- In $\mathbb{R}^k$ consider the partial order $a = (a_i) \preceq b = (b_i)$ if and only if $a_i \leq b_i$ for all $i = 1, 2, \ldots, k$;
- A *k-filtration* of S is a family $\mathcal{F} = \{\mathsf{S}^a\}_{a \in \mathbb{R}^k}$ of subsets of S with the following properties:
  - $\mathcal{F}$ is nested with respect to inclusions:

$$\mathsf{S}^a \subseteq \mathsf{S}^b, \text{ for every } a \preceq b$$

  - $\mathcal{F}$ is non-increasing on faces:

$$\text{if } \sigma \in \mathsf{S}^a \text{ and } \tau \text{ is a face of } \sigma \text{ then } \tau \in \mathsf{S}^a$$

- Given a function $f : \mathsf{S}_0 \to \mathbb{R}^k$, the *sublevel set filtration* is defined by

$$\mathsf{S}^a = \{\sigma = [v_0, v_1, \ldots, v_q] \in \mathsf{S} \mid f(v_i) \preceq a,\ i = 0, \ldots, q\}.$$

Multidimensional Persistent Homology

Persistence analyzes the homological changes of the filtration as *a* varies:

Multidimensional Persistent Homology

Persistence analyzes the homological changes of the filtration as a varies:

- for $a \preceq b$, consider the homomorphism

$$H_*(j^{(a,b)}) : H_*(\mathsf{S}^a) \to H_*(\mathsf{S}^b).$$

  induced by the inclusion map $j^{(a,b)} : \mathsf{S}^a \hookrightarrow \mathsf{S}^b$.

Multidimensional Persistent Homology

Persistence analyzes the homological changes of the filtration as a varies:

- for $a \preceq b$, consider the homomorphism

$$H_*(j^{(a,b)}) : H_*(\mathsf{S}^a) \to H_*(\mathsf{S}^b).$$

  induced by the inclusion map $j^{(a,b)} : \mathsf{S}^a \hookrightarrow \mathsf{S}^b$.
- The *i-th persistent homology group* of the filtration at (a, b) is image of the map $H_i(j^{(a,b)})$:

$$H_i^{a,b}(\mathsf{S}) := \operatorname{im} H_i(j^{(a,b)})$$

Filtration-preserving matching

A partial matching $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ on a filtered Lefschetz complex is said to *preserve the filtration* when, for every $a \in \mathbb{R}^k$,

$$\text{If } \sigma \in \mathsf{S}^a \text{ then } \mathtt{m}(\sigma) \in \mathsf{S}^a.$$

Filtration-preserving matching

A partial matching $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ on a filtered Lefschetz complex is said to *preserve the filtration* when, for every $a \in \mathbb{R}^k$,

$$\text{If } \sigma \in \mathsf{S}^a \text{ then } \mathtt{m}(\sigma) \in \mathsf{S}^a.$$

A filtration-preserving partial matching on S naturally induces a filtration on $\overline{\mathsf{S}}$: for each $\tau \in \overline{\mathsf{S}}$,

$$\tau \in \overline{\mathsf{S}}^a \iff \tau \in \mathsf{S}^a.$$

Filtration-preserving matching

A partial matching $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ on a filtered Lefschetz complex is said to *preserve the filtration* when, for every $a \in \mathbb{R}^k$,

$$\text{If } \sigma \in \mathsf{S}^a \text{ then } \mathtt{m}(\sigma) \in \mathsf{S}^a.$$

A filtration-preserving partial matching on S naturally induces a filtration on $\overline{\mathsf{S}}$: for each $\tau \in \overline{\mathsf{S}}$,

$$\tau \in \overline{\mathsf{S}}^a \iff \tau \in \mathsf{S}^a.$$

Proposition

Let $(\overline{S}, \overline{\kappa})$ be obtained from (S, κ) by reduction of the pair $(\mathtt{m}(\sigma), \sigma)$. Then, for each $q \in \mathbb{Z}$,

$$\pi(C_q(S^a)) \subseteq C_q(\overline{S}^a),\ \iota(C_q(\overline{S}^a)) \subseteq C_q(S^a),\ D_q(C_q(S^a)) \subseteq C_{q+1}(\overline{S}^a)$$

Reductions preserve persistent homology

Theorem

Let $\sigma \in A$ and let $(\overline{S}, \overline{\kappa})$ be obtained from (S, κ) by reduction of the pair $(m(\sigma), \sigma)$. Then the diagram

$$\begin{array}{ccc} H_*(S^a) & \xrightarrow{H_*(j^{(a,b)})} & H_*(S^b) \\ \downarrow \cong & & \downarrow \cong \\ H_*(\overline{S}^a) & \xrightarrow{H_*(j^{(a,b)})} & H_*(\overline{S}^b) \end{array}$$

commutes and $H_*^{a,b}(S)$ is isomorphic to $H_*^{a,b}(\overline{S})$.

Iterated reductions

Proposition

If (A, B, C, m) is acyclic then, for any $\tau \in A \setminus \{\sigma\}$, $\overline{\kappa}(m(\tau), \tau)$ is invertible. Furthermore, $\overline{\kappa}(m(\tau), \tau) = \kappa(m(\tau), \tau)$.

Iterated reductions

Proposition

If (A, B, C, m) is acyclic then, for any $\tau \in A \setminus \{\sigma\}$, $\overline{\kappa}(m(\tau), \tau)$ is invertible. Furthermore, $\overline{\kappa}(m(\tau), \tau) = \kappa(m(\tau), \tau)$.

Corollary

Let $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathtt{m})$ be an acyclic partial matching on (S, κ). Given a fixed $\sigma \in \mathsf{A}$, define $\overline{\mathsf{A}} = \mathsf{A} \setminus \{\sigma\}$, $\overline{\mathsf{B}} = \mathsf{B} \setminus \{\mathtt{m}(\sigma)\}$, $\overline{\mathtt{m}} = \mathtt{m}_{|\overline{\mathsf{A}}}$, and $\overline{\mathsf{C}} = \mathsf{C}$. Then $(\overline{\mathsf{C}}, \overline{\mathtt{m}} : \overline{\mathsf{A}} \to \overline{\mathsf{B}})$ is an acyclic partial matching on $(\overline{\mathsf{S}}, \overline{\kappa})$.

Corollary

For every $a \preceq b \in \mathbb{R}^k$, $H_*^{a,b}(\mathsf{C}) \cong H_*^{a,b}(\mathsf{S})$. Moreover, the diagram

$$\begin{array}{ccc} H_*(\mathsf{S}^a) & \xrightarrow{H_*(j^{(a,b)})} & H_*(\mathsf{S}^b) \\ \downarrow \cong & & \downarrow \cong \\ H_*(\mathsf{C}^a) & \xrightarrow{H_*(j^{(a,b)})} & H_*(\mathsf{C}^b) \end{array}$$

commutes.

The matching algorithm: introduction

- We start from the matching algorithm of [Robins-Wood-Sheppard 2010] for 1D persistent homology
  - input: a 3D cubical complex and an $\mathbb{R}$-valued injective function on its vertices
  - output: a filtration preserving acyclic partial matching
  - vertex based
  - lower-star based

The matching algorithm: introduction

- We start from the matching algorithm of [Robins-Wood-Sheppard 2010] for 1D persistent homology
  - input: a 3D cubical complex and an $\mathbb{R}$-valued injective function on its vertices
  - output: a filtration preserving acyclic partial matching
  - vertex based
  - lower-star based
- and extend it for multidimensional persistent homology
  - input: a simplicial complex, and a component-wise injective $\mathbb{R}^k$-valued function on its vertices and an ordering on its vertices
  - output: a multi-filtration-preserving acyclic partial matching
  - simplex based
  - lower-star based

Component-wise injective function on the vertices

$f = (f_1, \ldots, f_k) : S_0 \to \mathbb{R}^k$ is componentwise injective if each f_i is injective.

Component-wise injective function on the vertices

$f = (f_1, \ldots, f_k) : S_0 \to \mathbb{R}^k$ is componentwise injective if each f_i is injective.

Given $\tilde{f} : S_0 \to \mathbb{R}^k$, we obtain a component-wise injective $f : S_0 \to \mathbb{R}^k$ by perturbing f_i:

- set $\eta_i = \min\{|\tilde{f}_i(v) - \tilde{f}_i(w)| : v, w \in S_0 \wedge f_i(v) \neq f_i(w)\}$;
- for each i, order the n vertices of S_0 by an integer index j increasing with $\tilde{f}_i$;
- let $f_i(v_j) = \tilde{f}_i(v_j) + j\eta_i/n^s$, with $s \geq 1$

Component-wise injective function on the vertices

$f = (f_1, \ldots, f_k) : \mathsf{S}_0 \to \mathbb{R}^k$ is componentwise injective if each f_i is injective.

Given $\tilde{f} : \mathsf{S}_0 \to \mathbb{R}^k$, we obtain a component-wise injective $f : \mathsf{S}_0 \to \mathbb{R}^k$ by perturbing f_i:

- set $\eta_i = \min\{|\tilde{f}_i(v) - \tilde{f}_i(w)| : v, w \in \mathsf{S}_0 \wedge f_i(v) \neq f_i(w)\}$;
- for each i, order the n vertices of S_0 by an integer index j increasing with $\tilde{f}_i$;
- let $f_i(v_j) = \tilde{f}_i(v_j) + j\eta_i/n^s$, with $s \geq 1$

Extend f to a function $f : \mathsf{S} \to \mathbb{R}^k$ as follows.

$$f(\sigma) = (f_1(\sigma), \ldots, f_k(\sigma)) \qquad \text{with} \qquad f_i(\sigma) = \max_{v \in \mathsf{S}_0(\sigma)} f_i(v).$$

Indexing map for vertices

Lemma

There exists an injective function $I : S \to \mathbb{N}$ such that, for each $\sigma, \tau \in S$ with $\sigma \neq \tau$, if $\sigma \subseteq \tau$ or $f(\sigma) \precneqq f(\tau)$ then $I(\sigma) < I(\tau)$.

Indexing map for vertices

Lemma

There exists an injective function $I : S \to \mathbb{N}$ such that, for each $\sigma, \tau \in S$ with $\sigma \neq \tau$, if $\sigma \subseteq \tau$ or $f(\sigma) \precneqq f(\tau)$ then $I(\sigma) < I(\tau)$.

Construction of I:

Indexing map for vertices

Lemma

There exists an injective function $I : S \to \mathbb{N}$ such that, for each $\sigma, \tau \in S$ with $\sigma \neq \tau$, if $\sigma \subseteq \tau$ or $f(\sigma) \precneqq f(\tau)$ then $I(\sigma) < I(\tau)$.

Construction of *I*:

- consider the poset $(S, \sqsubseteq)$ with $\sigma \sqsubseteq \tau$ if and only if either $\sigma = \tau$ or $\sigma \neq \tau$ but in the latter case σ is a face of τ or $f(\sigma) \precneqq f(\tau)$

Indexing map for vertices

Lemma

There exists an injective function $I : S \to \mathbb{N}$ such that, for each $\sigma, \tau \in S$ with $\sigma \neq \tau$, if $\sigma \subseteq \tau$ or $f(\sigma) \precneqq f(\tau)$ then $I(\sigma) < I(\tau)$.

Construction of *I*:

- consider the poset $(S, \sqsubseteq)$ with $\sigma \sqsubseteq \tau$ if and only if either $\sigma = \tau$ or $\sigma \neq \tau$ but in the latter case σ is a face of τ or $f(\sigma) \precneqq f(\tau)$
- represent it by a Directed Acyclic Graph

Indexing map for vertices

Lemma

There exists an injective function $I : S \to \mathbb{N}$ such that, for each $\sigma, \tau \in S$ with $\sigma \neq \tau$, if $\sigma \subseteq \tau$ or $f(\sigma) \precneqq f(\tau)$ then $I(\sigma) < I(\tau)$.

Construction of *I*:

- consider the poset $(S, \sqsubseteq)$ with $\sigma \sqsubseteq \tau$ if and only if either $\sigma = \tau$ or $\sigma \neq \tau$ but in the latter case σ is a face of τ or $f(\sigma) \precneqq f(\tau)$
- represent it by a Directed Acyclic Graph
- apply the topological sorting algorithm

Indexing map for vertices

Lemma

There exists an injective function $I : S \to \mathbb{N}$ such that, for each $\sigma, \tau \in S$ with $\sigma \neq \tau$, if $\sigma \subseteq \tau$ or $f(\sigma) \precneqq f(\tau)$ then $I(\sigma) < I(\tau)$.

Construction of *I*:

- consider the poset $(S, \sqsubseteq)$ with $\sigma \sqsubseteq \tau$ if and only if either $\sigma = \tau$ or $\sigma \neq \tau$ but in the latter case σ is a face of τ or $f(\sigma) \precneqq f(\tau)$
- represent it by a Directed Acyclic Graph
- apply the topological sorting algorithm
- this algorithm has linear complexity

The lower star of a simplex

The lower star of a simplex

Given $f : \mathsf{S} \to \mathbb{R}^k$, the *lower star* of a simplex is the set

$$L(\sigma) = \{\alpha \in \mathsf{S} \mid \sigma \subseteq \alpha \quad \text{and} \quad f(\alpha) \preceq f(\sigma)\},$$

and the *reduced lower stars* is the set $L_*(\sigma) = L(\sigma) \setminus \{\sigma\}$.

The lower star of a simplex

Given $f : S \to \mathbb{R}^k$, the *lower star* of a simplex is the set

$$L(\sigma) = \{\alpha \in S \mid \sigma \subseteq \alpha \quad \text{and} \quad f(\alpha) \preceq f(\sigma)\},$$

and the *reduced lower stars* is the set $L_*(\sigma) = L(\sigma) \setminus \{\sigma\}$.

Lemma

If f is component-wise injective on the vertices, then the following statements hold:

(1) If $\tau \in L(\sigma)$, then $f(\tau) = f(\sigma)$.

(2) If $\tau \in L_*(\sigma)$, then $I(\sigma) < I(\tau)$.

(3) If $f(\sigma) = f(\tau)$ then there exists $\alpha \subseteq \sigma \cap \tau$ with $f(\alpha) = f(\sigma) = f(\tau)$.

(4) Assume that σ_1 and σ_2 are two distinct simplices of $\mathcal{K}$ such that $L(\sigma_1) \cap L(\sigma_2) \neq \emptyset$. Then, there exists a simplex $\beta \in \mathcal{K}$ such that $L(\sigma_1) \cup L(\sigma_2) \subseteq L(\beta)$ and $I(\beta) \leq \min\{I(\sigma_1), I(\sigma_2)\}$.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S} \to \mathbb{R}^k$ component-wise injective on the vertices, and an indexing $I : \mathsf{S} \to \mathbb{N}$.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.

The matching algorithm for multiD persistence

Input: A finite simplicial complex $\mathtt{S}$ with a function $f : \mathtt{S} \to \mathbb{R}^k$ component-wise injective on the vertices, and an indexing $I : \mathtt{S} \to \mathbb{N}$.
Output: Three lists $\mathtt{A}, \mathtt{B}, \mathtt{C}$ of simplices of $\mathtt{S}$, and a function $\mathtt{m} : \mathtt{A} \to \mathtt{B}$.

Set $\mathtt{A}, \mathtt{B}, \mathtt{C} = \emptyset$; $\mathtt{classified}(\sigma)$=false $\forall \sigma \in \mathtt{S}$; $\mathtt{PQzero}, \mathtt{PQone} = \emptyset$

The matching algorithm for multiD persistence

Input: A finite simplicial complex $\mathtt{S}$ with a function $f : \mathtt{S} \to \mathbb{R}^k$ component-wise injective on the vertices, and an indexing $I : \mathtt{S} \to \mathbb{N}$.
Output: Three lists $\mathtt{A}, \mathtt{B}, \mathtt{C}$ of simplices of $\mathtt{S}$, and a function $\mathtt{m} : \mathtt{A} \to \mathtt{B}$.

Set $\mathtt{A}, \mathtt{B}, \mathtt{C} = \emptyset$; $\mathtt{classified}(\sigma)$=false $\forall \sigma \in \mathtt{S}$; $\mathtt{PQzero}, \mathtt{PQone} = \emptyset$
for $i = 1$ to $\#S$

The matching algorithm for multiD persistence

Input: A finite simplicial complex $\mathtt{S}$ with a function $f : \mathtt{S} \to \mathbb{R}^k$ component-wise injective on the vertices, and an indexing $I : \mathtt{S} \to \mathbb{N}$.
Output: Three lists $\mathtt{A}, \mathtt{B}, \mathtt{C}$ of simplices of $\mathtt{S}$, and a function $\mathtt{m} : \mathtt{A} \to \mathtt{B}$.

Set $\mathtt{A}, \mathtt{B}, \mathtt{C} = \emptyset$; `classified`($\sigma$)=false $\forall \sigma \in \mathtt{S}$; `PQzero`, `PQone` $= \emptyset$
for $i = 1$ to $\#S$

set $\sigma := I^{-1}(i)$

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S} \to \mathbb{R}^k$ component-wise injective on the vertices, and an indexing $I : \mathsf{S} \to \mathbb{N}$.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.

Set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$; `classified`($\sigma$)=false $\forall \sigma \in \mathsf{S}$; $\mathtt{PQzero}, \mathtt{PQone} = \emptyset$
for $i = 1$ to $\#S$

set $\sigma := I^{-1}(i)$

if `classified`(σ)=false

if $L_*(\sigma)$ contains no cells

add σ to C, `classified`(σ)=true

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : S \to \mathbb{R}^k$ component-wise injective on the vertices, and an indexing $I : S \to \mathbb{N}$.
Output: Three lists A, B, C of simplices of S, and a function $m : A \to B$.

Set $A, B, C = \emptyset$; `classified`(σ)=false $\forall \sigma \in S$; `PQzero`, `PQone` $= \emptyset$
for $i = 1$ to $\#S$

set $\sigma := I^{-1}(i)$

if `classified`(σ)=false

if $L_*(\sigma)$ contains no cells

add σ to C, `classified`(σ)=true

else

set $\delta :=$ the (primary) coface in $L_*(\sigma)$ of minimal index $I(\delta)$
add σ to A and δ to B and define $m(\sigma) = \delta$,
`classified`(σ)=true, `classified`(δ)=true
add all $\alpha \in L_*(\sigma)$ to `PQzero` if `num_unclass_faces`$_\sigma(\alpha) = 0$
add all $\alpha \in L_*(\sigma)$ to `PQone` if `num_unclass_faces`$_\sigma(\alpha) = 1 \wedge \alpha > \delta$

while PQone $\neq \emptyset$ or PQzero $\neq \emptyset$

  while PQone $\neq \emptyset$

    set $\alpha :=$ PQone.pop_front

while PQone $\neq \emptyset$ or PQzero $\neq \emptyset$

    while PQone $\neq \emptyset$

        set $\alpha :=$ PQone.pop_front

        if num_unclass_faces$_\sigma(\alpha)$ = 0

            add α to PQzero

while PQone $\neq \emptyset$ or PQzero $\neq \emptyset$

  while PQone $\neq \emptyset$

    set $\alpha :=$ PQone.pop_front

    if num_unclass_faces$_\sigma(\alpha)$ = 0

      add α to PQzero

    else

      add $\lambda \in$ unclass_faces$_\sigma(\alpha)$ to A, add α to B and define m$(\lambda) = \alpha$, classified(α)=true, classified(λ)=true, remove λ from PQzero

      add all $\beta \in L_*(\sigma)$ to PQone if num_unclass_faces$_\sigma(\beta)$ = 1 and either $\beta > \alpha$ or $\beta > \lambda$

  endwhile

while PQone $\neq \emptyset$ or PQzero $\neq \emptyset$

  while PQone $\neq \emptyset$

    set $\alpha :=$ PQone.pop_front

    if num_unclass_faces$_\sigma(\alpha)$ = 0

      add α to PQzero

    else

      add $\lambda \in$ unclass_faces$_\sigma(\alpha)$ to A, add α to B and define m$(\lambda) = \alpha$, classified(α)=true, classified(λ)=true, remove λ from PQzero

      add all $\beta \in L_*(\sigma)$ to PQone if num_unclass_faces$_\sigma(\beta)$ = 1 and either $\beta > \alpha$ or $\beta > \lambda$

  endwhile

  if PQzero $\neq \emptyset$

    $\gamma :=$ PQzero.pop_front

    add γ to C, classified(γ)=true

    add all $\tau \in L_*(\sigma)$ to PQone if num_unclass_faces$_\sigma(\tau)$ = 1 $\wedge$ $\tau > \gamma$

  endif

endwhile

Correctness and complexity

Proposition

Each cell is processed exactly once by the algorithm and it is paired with some other cell or classified as critical. Hence the matching algorithm always terminates.

Theorem

The matching algorithm produces a partial matching (A, B, C, m) *that is acyclic and filtration-preserving.*

Correctness and complexity

Proposition

Each cell is processed exactly once by the algorithm and it is paired with some other cell or classified as critical. Hence the matching algorithm always terminates.

Theorem

The matching algorithm produces a partial matching (A, B, C, m) *that is acyclic and filtration-preserving.*

Complexity

The matching algorithm produces $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ in $O(\gamma \cdot \log \gamma \cdot (\#\mathtt{S}))$ steps, where γ is the maximum number of cofaces of simplices in $\mathtt{S}$.

The computational complexity of the reductions is $O((\#\mathtt{S}) \cdot \gamma \cdot (\#\mathtt{C})^2)$

[Mischaikow-Nanda 2013]

Numerical tests

- Considered 2D simplicial complexes.
- Filtered each complex by the $\mathbb{R}^2$-valued function defined on vertices.
- Present the results in a table where
  - row 1 shows the number of vertices, edges, faces, and the total number of cells of each considered mesh S,
  - while row 2 shows the same quantities referred to the cell complex C obtained by using our matching algorithm to reduce S.
  - Finally, row 3 shows the ratio between the second and the first rows, expressing them in percentage points. In other words, the lower are those ratios, the higher is the reduction rate.

Numerical tests on synthetic data

Table: Reduction performance on five different triangulations of the sphere.

$f = (x, y)$	sphere_1	sphere_2	sphere_3	sphere_4	sphere_5
#S	38	242	962	1538	2882
#C	4	20	98	178	278
%	10.5263	8.2645	10.1871	11.5735	9.6461

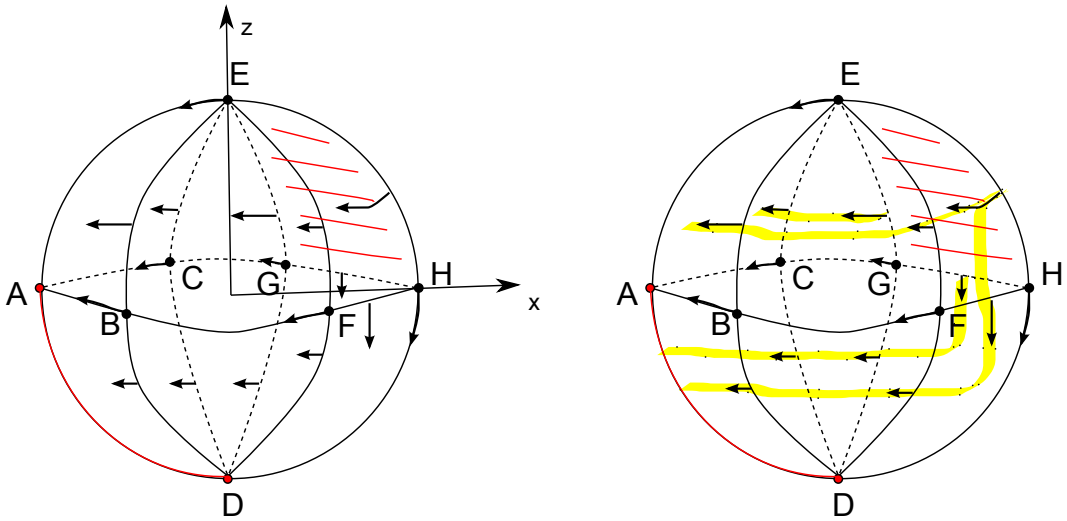


Table: Reduction performance on different triangulations of the torus.

$f = (x, y)$	torus_96	torus_4608	torus_7200
#S	96	4608	7200
#C	8	128	156
%	8.3333	2.7778	2.1667

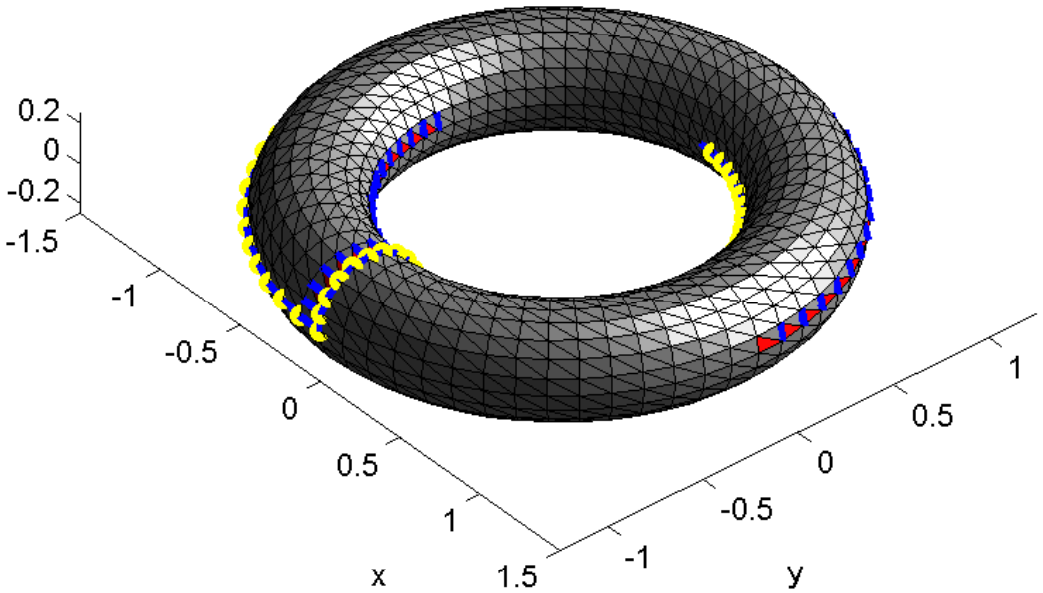
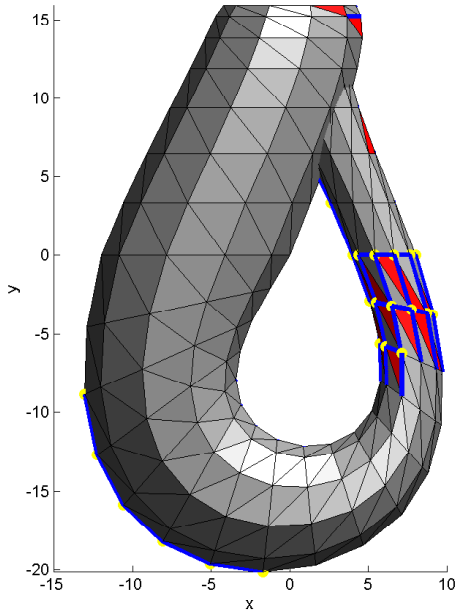
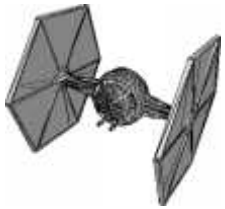
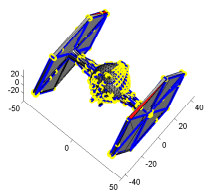
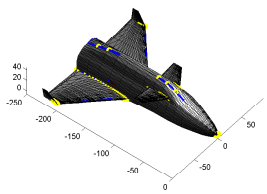
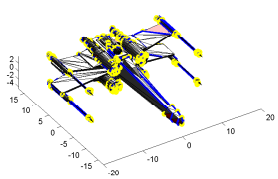
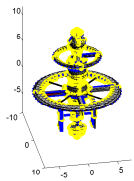


Table: Reduction performance on different triangulations approximating an immersion of the Klein bottle.

$f = (x, y)$	klein_89	klein_187	klein_491	klein_1881
$\#\mathcal{K}$	89	187	491	1881
#C	19	35	59	257
%	21.3483	18.7166	12.0163	13.6629



Numerical tests on real data

$f = (\|x\|, \|z\|)$	 tie	space_shuttle	x_wing	space_station
#S	11785	12658	18365	31935
#C	2287	484	2449	5149
%	19.4060	3.8237	13.3351	16.1234
C