Reducing complexes in Multidimensional Persistence

Claudia Landi

University of Modena and Reggio Emilia

joint work with M. Allili and T. Kaczynski

GETCO 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 exponential complexity and 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 exponential complexity and 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 exponential complexity and 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 exponential complexity and 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 exponential complexity and 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 exponential complexity and 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 exponential complexity and 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
- S-Complexes
- Multidimensional Persistent Homology

Outline

Introduction
- S-Complexes
- Multidimensional Persistent Homology

Partial Matching & Reductions in the framework of persistence

Outline

Introduction
- S-Complexes
- Multidimensional Persistent Homology

Partial Matching & Reductions in the framework of persistence

A matching algorithm for multidimensional persistence

Outline

Introduction
- S-Complexes
- Multidimensional Persistent Homology

Partial Matching & Reductions in the framework of persistence

A matching algorithm for multidimensional persistence

Conclusion

S-Complexes

1. Let S be a finite set with a gradation S_q such that $S_q = \emptyset$ for $q < 0$. For every element $\sigma \in S$ there exists a unique number q such that $\sigma \in S_q$. This number is called the dimension of σ and denoted $\dim \sigma$.

S-Complexes

1. Let S be a finite set with a gradation S_q such that $S_q = \emptyset$ for $q < 0$. For every element $\sigma \in S$ there exists a unique number q such that $\sigma \in S_q$. This number is called the dimension of σ and denoted $\dim \sigma$.
2. Let $\kappa : S \times S \to R$, R a PID, be a function such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$. κ is called the *coincidence index*.

S-Complexes

1. Let S be a finite set with a gradation S_q such that $\mathsf{S}_q = \emptyset$ for $q < 0$. For every element $\sigma \in \mathsf{S}$ there exists a unique number q such that $\sigma \in \mathsf{S}_q$. This number is called the dimension of σ and denoted $\dim \sigma$.
2. Let $\kappa : \mathsf{S} \times \mathsf{S} \to R$, R a PID, be a function such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$. κ is called the *coincidence index*.
3. $C_q(\mathsf{S}) := R(\mathsf{S}_q)$, the free module over R generated by S_q.

S-Complexes

1. Let S be a finite set with a gradation S_q such that $S_q = \emptyset$ for $q < 0$. For every element $\sigma \in S$ there exists a unique number q such that $\sigma \in S_q$. This number is called the dimension of σ and denoted $\dim \sigma$.
2. Let $\kappa : S \times S \to R$, R a PID, be a function such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$. κ is called the *coincidence index*.
3. $C_q(S) := R(S_q)$, the free module over R generated by S_q.
4. We say that (S, κ) is an *S-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.

S-Complexes

1. Let S be a finite set with a gradation S_q such that $S_q = \emptyset$ for $q < 0$. For every element $\sigma \in S$ there exists a unique number q such that $\sigma \in S_q$. This number is called the dimension of σ and denoted $\dim \sigma$.
2. Let $\kappa : S \times S \to R$, R a PID, be a function such that, if $\kappa(\sigma, \tau) \neq 0$, then $\dim \sigma = \dim \tau + 1$. κ is called the *coincidence index*.
3. $C_q(S) := R(S_q)$, the free module over R generated by S_q.
4. We say that (S, κ) is an *S-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. By the homology of an S-complex (S, κ) we mean the homology of the chain complex $(C_*(S), \partial^\kappa_*)$, and we denote it by $H_*(S, \kappa)$ or simply by $H_*(S)$.

Examples of S-Complexes

Main examples of S-complexes:

- simplicial complexes:

Examples of S-Complexes

Main examples of S-complexes:

- simplicial complexes:
  1. assume an ordering of S_0 is given and every simplex σ in S is coded as $[v_0, v_1, \ldots v_q]$, where the vertices $v_0, v_1, \ldots v_q$ are listed according to the prescribed ordering of S_0.

Examples of S-Complexes

Main examples of S-complexes:

- simplicial complexes:
  1. assume an ordering of S_0 is given and every simplex σ in S is coded as $[v_0, v_1, \ldots v_q]$, where the vertices $v_0, v_1, \ldots v_q$ are listed according to the prescribed ordering of S_0.
  2. define

$$\kappa(\sigma, \tau) := \begin{cases} (-1)^i & \text{if } \sigma = [v_0, v_1, \ldots, v_q] \\ & \text{and } \tau = [v_0, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_q] \\ 0 & \text{otherwise.} \end{cases}$$

Examples of S-Complexes

Main examples of S-complexes:

- simplicial complexes:
  1. assume an ordering of S_0 is given and every simplex σ in S is coded as $[v_0, v_1, \ldots v_q]$, where the vertices $v_0, v_1, \ldots v_q$ are listed according to the prescribed ordering of S_0.
  2. define

$$\kappa(\sigma, \tau) := \begin{cases} (-1)^i & \text{if } \sigma = [v_0, v_1, \ldots, v_q] \\ & \text{and } \tau = [v_0, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_q] \\ 0 & \text{otherwise.} \end{cases}$$

  3. obtain an S-complex whose chain complex is the classical simplicial chain complex used in simplicial homology

Examples of S-Complexes

Main examples of S-complexes:

- simplicial complexes:
  1. assume an ordering of S_0 is given and every simplex σ in S is coded as $[v_0, v_1, \ldots v_q]$, where the vertices $v_0, v_1, \ldots v_q$ are listed according to the prescribed ordering of S_0.
  2. define

$$\kappa(\sigma, \tau) := \begin{cases} (-1)^i & \text{if } \sigma = [v_0, v_1, \ldots, v_q] \\ & \text{and } \tau = [v_0, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_q] \\ 0 & \text{otherwise.} \end{cases}$$

  3. obtain an S-complex whose chain complex is the classical simplicial chain complex used in simplicial homology
- cubical complexes

Examples of S-Complexes

Main examples of S-complexes:

- simplicial complexes:
  1. assume an ordering of S_0 is given and every simplex σ in S is coded as $[v_0, v_1, \ldots v_q]$, where the vertices $v_0, v_1, \ldots v_q$ are listed according to the prescribed ordering of S_0.
  2. define

$$\kappa(\sigma, \tau) := \begin{cases} (-1)^i & \text{if } \sigma = [v_0, v_1, \ldots, v_q] \\ & \text{and } \tau = [v_0, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_q] \\ 0 & \text{otherwise.} \end{cases}$$

  3. obtain an S-complex whose chain complex is the classical simplicial chain complex used in simplicial homology
- cubical complexes
- cell complexes

Multi-filtration of an S-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 an S-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 an S-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 $H_i(j^{(a,b)})$:

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

Partial matchings

Given an S-complex (S, κ), a *partial matching* on (S, κ) is a quadruplet $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathtt{m})$ where

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

In particular, $\mathtt{m}(\tau)$ is a primary coface of τ.

Partial matchings

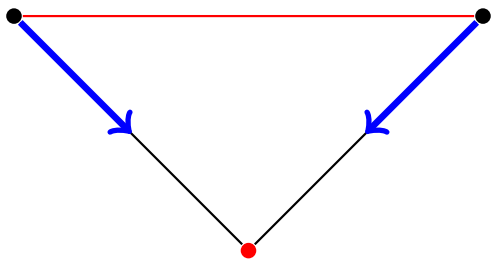
Given an S-complex (S, κ), a *partial matching* on (S, κ) is a quadruplet $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathtt{m})$ where

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

In particular, $\mathtt{m}(\tau)$ is a primary coface of τ.

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

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



Partial matchings

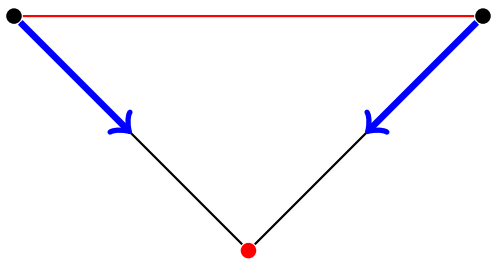
Given an S-complex (S, κ), a *partial matching* on (S, κ) is a quadruplet $(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathtt{m})$ where

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

In particular, $\mathtt{m}(\tau)$ is a primary coface of τ.

$(\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathtt{m})$ 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

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

Reductions

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 an S-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 an S-complex.

Reductions

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 an S-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 an S-complex.

Proposition

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

Reductions

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 an S-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 an S-complex.

Proposition

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

Theorem

*Fixed $\sigma \in A$, define $\overline{A} = A \setminus \{\sigma\}$, $\overline{B} = B \setminus \{m(\sigma)\}$, $\overline{m} = m_{|\overline{A}}$, and $\overline{C} = C$.
If (A, B, C, m) is an acyclic partial matching, then $(\overline{A}, \overline{B}, \overline{C}, \overline{m})$ is an acyclic partial matching too.*

Reductions preserve homology [KMS 1998]

- 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

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

Reductions preserve homology [KMS 1998]

- Define $\pi : C_*(\mathsf{S}) \to C_*(\overline{\mathsf{S}})$ and $\iota : C_*(\overline{\mathsf{S}}) \to C_*(\mathsf{S})$ on generators by setting

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

  and

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

- π and ι are chain equivalences with the chain homotopy $D_* : C_*(\mathsf{S}) \to C_{*+1}(\mathsf{S})$ given on generators $\tau \in \mathsf{S}_q$, $q \in \mathbb{Z}$, by

$$D_q(\tau) = \begin{cases} \frac{1}{\kappa(\mathtt{m}(\sigma),\sigma)}\mathtt{m}(\sigma) & \text{if } \tau = \sigma \\ 0 & \text{otherwise} \end{cases}$$

Filtration-preserving matching

A partial matching $(\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{m})$ on a filtered S-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 S-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 S-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

Proposition

The maps $\pi_{|C_*(S^a)} : C_*(S^a) \to C_*(\overline{S}^a)$ *and* $\iota_{|C_*(\overline{S}^a)} : C_*(\overline{S}^a) \to C_*(S^a)$ *defined by restriction are chain homotopy equivalences. Moreover, 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.

Reductions preserve persistent homology

Proposition

The maps $\pi_{|C_*(S^a)} : C_*(S^a) \to C_*(\overline{S}^a)$ *and* $\iota_{|C_*(\overline{S}^a)} : C_*(\overline{S}^a) \to C_*(S^a)$ *defined by restriction are chain homotopy equivalences. Moreover, 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.

Theorem

For every $a \preceq b \in \mathbb{R}^k$, $H_*^{a,b}(S)$ *is isomorphic to* $H_*^{a,b}(\overline{S})$.

The matching algorithm: introduction

- We start from the matching algorithm of [King-Knudson-Mramor 2005] for homology
  - input: a simplicial complex and an $\mathbb{R}$-valued function on its vertices
  - output: an acyclic partial matching
  - recursive
  - vertex-based and lower-link-based
  - cancels pairs of critical cells to achieve optimality

The matching algorithm: introduction

- We start from the matching algorithm of [King-Knudson-Mramor 2005] for homology
  - input: a simplicial complex and an $\mathbb{R}$-valued function on its vertices
  - output: an acyclic partial matching
  - recursive
  - vertex-based and lower-link-based
  - cancels pairs of critical cells to achieve optimality
- and extend it for multidimensional persistent homology
  - input: a simplicial complex, an $\mathbb{R}^k$-valued function and an ordering on its vertices
  - output: a filtration-preserving acyclic partial matching
  - recursive
  - vertex-based and lower-link-based
  - without cancellation

Indexing map for vertices

Lemma

There exists an injective function $I : S_0 \to \mathbb{N}$ *such that, for each* $v, w \in S_0$ *with* $v \neq w$, $f(v) \subsetneq f(w)$ *implies* $I(v) < I(w)$.

Indexing map for vertices

Lemma

There exists an injective function $I : S_0 \to \mathbb{N}$ *such that, for each* $v, w \in S_0$ *with* $v \neq w$, $f(v) \not\subseteq f(w)$ *implies* $I(v) < I(w)$.

Construction of I:

Indexing map for vertices

Lemma

There exists an injective function $I : S_0 \to \mathbb{N}$ *such that, for each* $v, w \in S_0$ *with* $v \neq w$, $f(v) \precneqq f(w)$ *implies* $I(v) < I(w)$.

Construction of I:

- $f : S_0 \to \mathbb{R}^k$ induces a partial order on the vertices

Indexing map for vertices

Lemma

There exists an injective function $I : S_0 \to \mathbb{N}$ *such that, for each* $v, w \in S_0$ *with* $v \neq w$, $f(v) \precneqq f(w)$ *implies* $I(v) < I(w)$.

Construction of I:

- $f : S_0 \to \mathbb{R}^k$ induces a partial order on the vertices
- represent it by a Directed Acyclic Graph

Indexing map for vertices

Lemma

There exists an injective function $I : S_0 \to \mathbb{N}$ *such that, for each* $v, w \in S_0$ *with* $v \neq w$, $f(v) \precneqq f(w)$ *implies* $I(v) < I(w)$.

Construction of I:

- $f : S_0 \to \mathbb{R}^k$ induces a partial order on the vertices
- represent it by a Directed Acyclic Graph
- apply the topological sorting algorithm

Indexing map for vertices

Lemma

There exists an injective function $I : S_0 \to \mathbb{N}$ *such that, for each* $v, w \in S_0$ *with* $v \neq w$, $f(v) \precneqq f(w)$ *implies* $I(v) < I(w)$.

Construction of I:

- $f : S_0 \to \mathbb{R}^k$ induces a partial order on the vertices
- represent it by a Directed Acyclic Graph
- apply the topological sorting algorithm
- this algorithm has linear complexity

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
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 S with a function $f : S_0 \to \mathbb{R}^k$ and an indexing $I : S_0 \to \mathbb{N}$ on its vertices.
Output: Three lists A, B, C of simplices of S, and a function $m : A \to B$.
function Partition (complex S, function f, indexing map I)

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.
   If $\mathsf{S}'(v)$ is empty, then add v to C. Else

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.
   If $\mathsf{S}'(v)$ is empty, then add v to C. Else

   add v to A.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.
   If $\mathsf{S}'(v)$ is empty, then add v to C. Else

      add v to A.
      let $f' : \mathsf{S}'_0(v) \to \mathbb{R}^k$ and $I' : \mathsf{S}'_0(v) \to \mathbb{N}$ be the restrictions of f and I.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.
   If $\mathsf{S}'(v)$ is empty, then add v to C. Else

      add v to A.
      let $f' : \mathsf{S}'_0(v) \to \mathbb{R}^k$ and $I' : \mathsf{S}'_0(v) \to \mathbb{N}$ be the restrictions of f and I.
      Call Partition (recursively): $(\mathsf{A}', \mathsf{B}', \mathsf{C}', \mathtt{m}') = \text{Partition}(\mathsf{S}'(v), f', I')$.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.
   If $\mathsf{S}'(v)$ is empty, then add v to C. Else

   add v to A.
   let $f' : \mathsf{S}'_0(v) \to \mathbb{R}^k$ and $I' : \mathsf{S}'_0(v) \to \mathbb{N}$ be the restrictions of f and I.
   Call Partition (recursively): $(\mathsf{A}', \mathsf{B}', \mathsf{C}', \mathtt{m}') = \text{Partition}(\mathsf{S}'(v), f', I')$.
   Set $\mathsf{D}' = \{w \in \mathsf{C}'_0 \mid f(w) \text{ is minimal in } \mathsf{C}'_0 \text{ w.r.t. } \precneqq\}$.
   Set w_0 as the vertex with smallest index I' in D'.
   Add $[v, w_0]$ to B and define $\mathtt{m}(v) = [v, w_0]$.
   For each $\sigma \in \mathsf{C}' \setminus \{w_0\}$, add $v * \sigma$ to C.
   For each $\sigma \in \mathsf{A}'$, add $v * \sigma$ to A, $v * \mathtt{m}'(\sigma)$ to B, and set $\mathtt{m}(v * \sigma) = v * \mathtt{m}'(\sigma)$.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.
   If $\mathsf{S}'(v)$ is empty, then add v to C. Else

   add v to A.
   let $f' : \mathsf{S}'_0(v) \to \mathbb{R}^k$ and $I' : \mathsf{S}'_0(v) \to \mathbb{N}$ be the restrictions of f and I.
   Call Partition (recursively): $(\mathsf{A}', \mathsf{B}', \mathsf{C}', \mathtt{m}') = \text{Partition}(\mathsf{S}'(v), f', I')$.
   Set $\mathsf{D}' = \{w \in \mathsf{C}'_0 \mid f(w) \text{ is minimal in } \mathsf{C}'_0 \text{ w.r.t. } \precneqq\}$.
   Set w_0 as the vertex with smallest index I' in D'.
   Add $[v, w_0]$ to B and define $\mathtt{m}(v) = [v, w_0]$.
   For each $\sigma \in \mathsf{C}' \setminus \{w_0\}$, add $v * \sigma$ to C.
   For each $\sigma \in \mathsf{A}'$, add $v * \sigma$ to A, $v * \mathtt{m}'(\sigma)$ to B, and set $\mathtt{m}(v * \sigma) = v * \mathtt{m}'(\sigma)$.
3. For each $\sigma \in \mathsf{S} \setminus (\mathsf{A} \cup \mathsf{B} \cup \mathsf{C})$, add σ to C.

The matching algorithm for multiD persistence

Input: A finite simplicial complex S with a function $f : \mathsf{S}_0 \to \mathbb{R}^k$ and an indexing $I : \mathsf{S}_0 \to \mathbb{N}$ on its vertices.
Output: Three lists $\mathsf{A}, \mathsf{B}, \mathsf{C}$ of simplices of S, and a function $\mathtt{m} : \mathsf{A} \to \mathsf{B}$.
function Partition (complex S, function f, indexing map I)

1. Initially, set $\mathsf{A}, \mathsf{B}, \mathsf{C} = \emptyset$.
2. For each $v \in \mathsf{S}_0$,

   Let $\mathsf{S}'(v) = \{\tau \in \mathsf{S} \mid v * \tau \in \mathsf{S} \wedge \forall \text{ vertex } w \in \tau, f(w) \precneqq f(v)\}$.
   If $\mathsf{S}'(v)$ is empty, then add v to C. Else

   add v to A.
   let $f' : \mathsf{S}'_0(v) \to \mathbb{R}^k$ and $I' : \mathsf{S}'_0(v) \to \mathbb{N}$ be the restrictions of f and I.
   Call Partition (recursively): $(\mathsf{A}', \mathsf{B}', \mathsf{C}', \mathtt{m}') = \text{Partition}(\mathsf{S}'(v), f', I')$.
   Set $\mathsf{D}' = \{w \in \mathsf{C}'_0 \mid f(w) \text{ is minimal in } \mathsf{C}'_0 \text{ w.r.t. } \precneqq\}$.
   Set w_0 as the vertex with smallest index I' in D'.
   Add $[v, w_0]$ to B and define $\mathtt{m}(v) = [v, w_0]$.
   For each $\sigma \in \mathsf{C}' \setminus \{w_0\}$, add $v * \sigma$ to C.
   For each $\sigma \in \mathsf{A}'$, add $v * \sigma$ to A, $v * \mathtt{m}'(\sigma)$ to B, and set $\mathtt{m}(v * \sigma) = v * \mathtt{m}'(\sigma)$.
3. For each $\sigma \in \mathsf{S} \setminus (\mathsf{A} \cup \mathsf{B} \cup \mathsf{C})$, add σ to C.
4. **return** A, B, C, $\mathtt{m}$.

Correctness, reduction, and complexity

Theorem

The matching algorithm produces a partial matching (A, B, C, m) *that is acyclic. Moreover, if* $\sigma \in S^a$ *then* $m(\sigma) \in S^a$.

Correctness, reduction, and complexity

Theorem

The matching algorithm produces a partial matching (A, B, C, m) *that is acyclic. Moreover, if* $\sigma \in S^a$ *then* $m(\sigma) \in S^a$.

Corollary

For every $a \preceq b \in \mathbb{R}^k$, $H_*^{a,b}(\mathsf{S})$ is isomorphic to $H_*^{a,b}(\mathsf{C})$.

Correctness, reduction, and complexity

Theorem

The matching algorithm produces a partial matching (A, B, C, m) *that is acyclic. Moreover, if* $\sigma \in S^a$ *then* $m(\sigma) \in S^a$.

Corollary

For every $a \preceq b \in \mathbb{R}^k$, $H_*^{a,b}(S)$ is isomorphic to $H_*^{a,b}(C)$.

Theorem

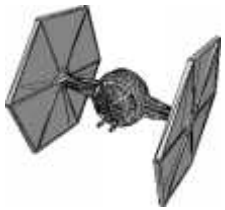
The algorithm produces (A, B, C, m) *in less than* $2\gamma^d(d+1)!N$ *steps, where*

- d *: dimension of* S
- N *: cardinality of* S_0
- γ *: upper bound on the number of cofaces of any simplex in* S

Numerical tests

- Considered four simplicial complexes.
- Each complex was filtered by the $\mathbb{R}^2$-valued function defined on vertices by $f(v) = (|x|, |y|)$.
- 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

	 tie	 space_shuttle	 x_wing	 space_station
$\#S_0$	2014	2376	3099	5749
$\#C_0$	228	121	175	1879
%	11.3	5.1	5.6	32.7
$\#S_1$	5944	6330	9190	15949
$\#C_1$	3343	3699	3605	11158
%	56.2	58.4	39.2	70.0
$\#S_2$	3827	3952	6076	10237
$\#C_2$	3012	3576	3415	9316
%	78.7	90.5	56.2	91.0
#S	11785	12658	18365	31935
#C	6583	7396	7195	22353
%	55.9	58.4	39.2	70.0

Conclusions

- The reduction algorithm has linear complexity vs polynomial complexity of persistence computation
- May be a convenient pre-processing technique
- Full paper available at `arXiv:1310.8089v2`
- On-going research focuses on improving the results obtained so far:
  - using a simplex-based and lower-star-based algorithm inspired from [Robins et al. 2010]

Conclusions

- The reduction algorithm has linear complexity vs polynomial complexity of persistence computation
- May be a convenient pre-processing technique
- Full paper available at `arXiv:1310.8089v2`
- On-going research focuses on improving the results obtained so far:
  - using a simplex-based and lower-star-based algorithm inspired from [Robins et al. 2010]

Thank you for your attention!