Multi-parameter persistent homology: applications and algorithms

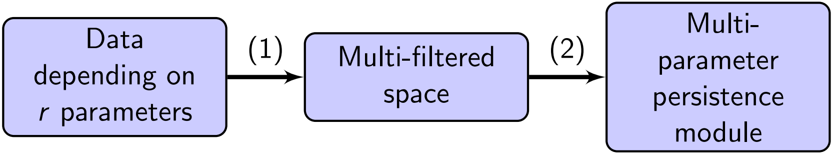
Nina Otter

Mathematical Institute, University of Oxford

Gudhi/Top Data Workshop

Porquerolles, 18 October 2016

Multi-parameter persistent homology pipeline



Step (1): from data to multi-filtered spaces

Define the following partial order on $\mathbb{N}^r$:
$(u_1, \ldots, u_r) \leq (v_1, \ldots, v_r)$ iff $u_i \leq v_i$ for all $i = 1, \ldots, r$.

A *multi-filtered space K* is a set of spaces $\{K_u\}_{u \in \mathbb{N}^r}$ such that $K_u \subseteq K_v$ if $u \leq v$ for all $u, v \in \mathbb{N}^r$.

Step (1): from data to multi-filtered spaces

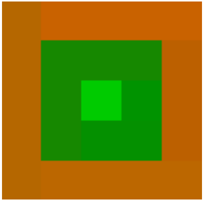
Define the following partial order on $\mathbb{N}^r$:
$(u_1, \ldots, u_r) \leq (v_1, \ldots, v_r)$ iff $u_i \leq v_i$ for all $i = 1, \ldots, r$.

A *multi-filtered space* K is a set of spaces $\{K_u\}_{u \in \mathbb{N}^r}$ such that $K_u \subseteq K_v$ if $u \leq v$ for all $u, v \in \mathbb{N}^r$.

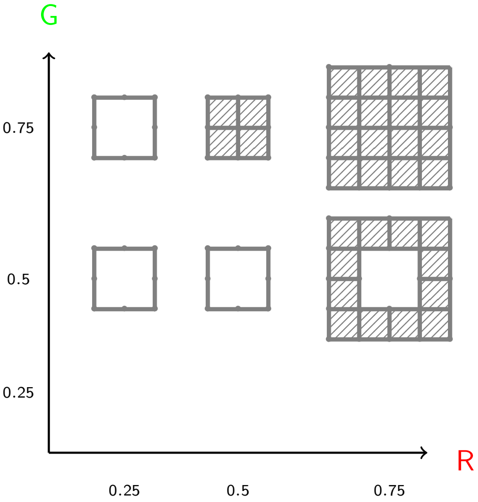
Map $f\colon X \to \mathbb{R}^r$ $\longrightarrow$ r-filtered simplicial complex

digital image with color vectors of length r $\longrightarrow$ r-filtered cubical complex

Step (1): from data to multi-filtered spaces: example



$\longrightarrow$



Step (2): from multi-filtered spaces to multi-parameter persistence modules

r-filtered space	$\xrightarrow{H_i}$	r-parameter persistence module

An *r-parameter persistence module* is a tuple $(\{M_i\}_{i\in\mathbb{N}^r}, \{\phi_{i,j}\}_{i\leq j\in\mathbb{N}^r})$ where:

- for each $i \in \mathbb{N}^r$ we have that M_i is a k-module
- for every $i \leq j$ we have that $\phi_{i,j}\colon M_i \to M_j$ is a k-module homomorphism such that whenever $i \leq k \leq j$ we have

$$\phi_{k,j} \circ \phi_{i,k} = \phi_{i,j}.$$

In other words, an r-parameter persistence module is a functor $F\colon \mathbb{N}^r \to k\mathrm{Mod}$.

Interlude: representation theory of quivers

Interlude: representation theory of quivers

A *quiver* $Q = (Q_0, Q_1, s, t)$ consists of two non-empty sets Q_0, Q_1 and two maps $s, t \colon Q_1 \to Q_0$. A quiver is *finite* if both Q_0 and Q_1 are finite.

Whenever $s(u) = x$ and $t(u) = y$ we write $x \xrightarrow{u} y$. For example, the following are finite quivers:

$$\begin{tikzcd} x \arrow[loop left, "u"] \arrow[loop right, "v"] \end{tikzcd} \qquad \begin{tikzcd}[column sep=large] y \arrow[r, bend left, "s"] \arrow[r, bend right, "h"'] & w \arrow[l, "t"'] \end{tikzcd}$$

Representations of quivers

Let k be a field. A *representation of a quiver* (V, ϕ) consists of a family of k-vector spaces $V = \{V_i\}_{i \in Q_0}$ together with a family of k-linear maps $\phi = \{\phi_e \colon V_{s(e) \to V_{t(e)} | e \in Q_1}\}$. A representation (V, ϕ) is *finite-dimensional* if for all $i \in Q_0$ the vector space V_i is finite-dimensional.

A *morphism of representations* $f \colon (V, \phi) \to (V', \phi')$ is given by k-linear maps $f_i \colon V_i \to V'_i$ for all $i \in Q_0$ such that the following diagram

$$\begin{tikzcd}
V_{s(e)} \arrow[r, "\phi_e"] \arrow[d, "f_{s(e)}"'] & V_{t(e)} \arrow[d, "f_{t(e)}"] \\
V'_{s(e)} \arrow[r, "\phi'_{t(e)}"'] & V'_{t(e)}
\end{tikzcd}$$

commutes for all $e \in Q_1$.

Examples of quiver representations

$$x \xrightarrow{u} y$$

Two finite-dimensional representations $\phi\colon V' \to V$ and $\psi\colon W' \to W$ are isomorphic iff $\dim V' = \dim W'$ and $\dim V = \dim W$ and $\mathrm{rank}\phi = \mathrm{rank}\psi$.

$$\begin{tikzcd} x \arrow[loop above, "u"] \end{tikzcd}$$

Two finite-dimensional representations $\phi\colon V \to V$ and $\psi\colon W \to W$ are isomorphic iff ϕ and ψ have the same Jordan normal form.

$$\begin{tikzcd} x \arrow[loop left, "u"] \arrow[loop right, "v"] \end{tikzcd}$$

Studying isomorphism classes of representations of this quiver amounts to studying pairs of quadratic matrices up to simultaneous conjugation.

Indecomposable representations

The *direct sum* of two representations (ϕ, V) and (ψ, W) is the representation $(\phi \oplus \psi, V \oplus W)$ where $V \oplus W = V_i \oplus W_i$ for all $i \in Q_0$ and $(\phi \oplus \psi)_e = \begin{pmatrix} \phi_e & 0 \\ 0 & \psi_e \end{pmatrix}$.

Indecomposable representations

The *direct sum* of two representations (ϕ, V) and (ψ, W) is the representation $(\phi \oplus \psi, V \oplus W)$ where $V \oplus W = V_i \oplus W_i$ for all $i \in Q_0$ and $(\phi \oplus \psi)_e = \begin{pmatrix} \phi_e & 0 \\ 0 & \psi_e \end{pmatrix}$.

We say that a representation (ϕ, V) is *indecomposable* if it is non-zero and not isomorphic to a direct sum of two non-zero representations.

Indecomposable representations

The *direct sum* of two representations (ϕ, V) and (ψ, W) is the representation $(\phi \oplus \psi, V \oplus W)$ where $V \oplus W = V_i \oplus W_i$ for all $i \in Q_0$ and $(\phi \oplus \psi)_e = \begin{pmatrix} \phi_e & 0 \\ 0 & \psi_e \end{pmatrix}$.

We say that a representation (ϕ, V) is *indecomposable* if it is non-zero and not isomorphic to a direct sum of two non-zero representations.

Example: indecomposable representations of the loop quiver are given by the Jordan blocks.

Indecomposable representations

The *direct sum* of two representations (ϕ, V) and (ψ, W) is the representation $(\phi \oplus \psi, V \oplus W)$ where $V \oplus W = V_i \oplus W_i$ for all $i \in Q_0$ and $(\phi \oplus \psi)_e = \begin{pmatrix} \phi_e & 0 \\ 0 & \psi_e \end{pmatrix}$.

We say that a representation (ϕ, V) is *indecomposable* if it is non-zero and not isomorphic to a direct sum of two non-zero representations.

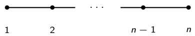
Example: indecomposable representations of the loop quiver are given by the Jordan blocks.

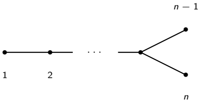
Theorem (Krull, Remak, Schmidt) Assume that Q is finite, then any finite-dimensional representation (V, ϕ) of Q can be written as a direct sum $(V, \phi) = (V_1, \phi_1) \oplus \cdots \oplus (V_r, \phi_r)$ where each (V_i, ϕ_i) is indecomposable, and the decomposition is unique up to isomorphism and permutation of the terms.

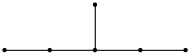
Classification of (representations of) quivers

Dynkin

($n \geq 2$)



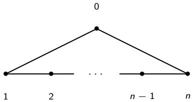


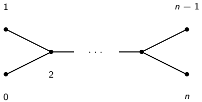


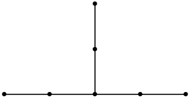
...

Extended Dynkin

($n \geq 2$)



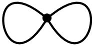




...

Wild

Everything else. For example:



Classification of representations of quivers

Suppose that k is algebraically closed. The number of isomorphism classes of indecomposable representations is:

Dynkin	**Extended Dynkin**	**Wild**
Finite.	Infinite; depends on one parameter.	Infinite; depends on $N > 1$ parameters, where N depends on the quiver.

G. Kac, *Infinite root systems, representations of graphs and invariant theory I - II*, 1980-2 and P. Gabriel, *Unzerlegbare darstellungen I*, 1972.

Classification of indecomposable representations of quivers: example

Conside again the loop quiver:

Classification of indecomposable representations of quivers: example

Conside again the loop quiver:

Recall that two finite-dimensional representations $\phi \colon V \to V$ and $\psi \colon W \to W$ are isomorphic iff ϕ and ψ have the same Jordan normal form, and the isomorphism classes of indecomposable representations of the loop quiver are given by the Jordan blocks.

Each Jordan block depends on a continuous parameter given by the eigenvalue.

Back to multi-parameter persistent homology

A multi-parameter persistence module is a representation of a quiver of the following form:

$$
\begin{tikzcd}
\cdots & \cdots & \cdots & \\
\bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \cdots \\
\bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \cdots \\
\bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \cdots
\end{tikzcd}
$$

Back to multi-parameter persistent homology

A multi-parameter persistence module is a representation of a quiver of the following form:

$$\begin{tikzcd}
\cdots & \cdots & \cdots & \\
\bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \cdots \\
\bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \cdots \\
\bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \bullet \arrow[r] \arrow[u] & \cdots
\end{tikzcd}$$

Such quivers are wild.

Back to multi-parameter persistent homology

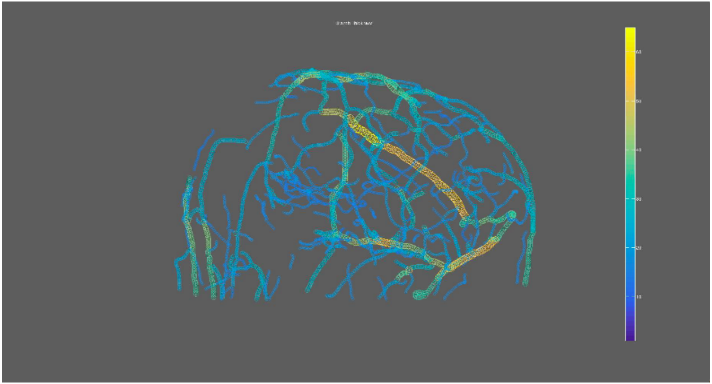
A multi-parameter persistence module is a representation of a quiver of the following form:

$$\begin{tikzcd}
\cdots & \cdots & \cdots & \\
\bullet \arrow[u] \arrow[r] & \bullet \arrow[u] \arrow[r] & \bullet \arrow[u] \arrow[r] & \cdots \\
\bullet \arrow[u] \arrow[r] & \bullet \arrow[u] \arrow[r] & \bullet \arrow[u] \arrow[r] & \cdots \\
\bullet \arrow[u] \arrow[r] & \bullet \arrow[u] \arrow[r] & \bullet \arrow[u] \arrow[r] & \cdots
\end{tikzcd}$$

Such quivers are wild.

Our motivation/goal: find computable invariants for applications.

Application: Time evolution of blood vessel growth in presence of tumors



Roche, Oxford Oncology (B. Markelc), Mathematical Biology, University of Oxford (B. Stolz, H. Byrne, J. Grogan)

Persistence modules are modules

Persistence modules are modules

- Recall that an $\mathbb{N}^r$-graded (or multi-graded) ring is a ring A together with a collection $\{A_u\}_{u \in \mathbb{N}^r}$ of subgroups of the underlying abelian group of A such that $A = \oplus_{u \in \mathbb{N}^r} A_u$ and for all $a \in A_m$ and $b \in A_n$ we have $ab \in A_{m+n}$.

Persistence modules are modules

- Recall that an $\mathbb{N}^r$-graded (or multi-graded) ring is a ring A together with a collection $\{A_u\}_{u \in \mathbb{N}^r}$ of subgroups of the underlying abelian group of A such that $A = \oplus_{u \in \mathbb{N}^r} A_u$ and for all $a \in A_m$ and $b \in A_n$ we have $ab \in A_{m+n}$.
- Make the ring $A = k[x_1, \ldots, x_r]$ into an $\mathbb{N}^r$-graded ring by setting

$$A_u = k x_1^{u_1} \ldots x_r^{u_r} \text{ for all } u = (u_1, \ldots, u_r) \in \mathbb{N}^r.$$

Persistence modules are modules

- Recall that an $\mathbb{N}^r$-graded (or multi-graded) ring is a ring A together with a collection $\{A_u\}_{u \in \mathbb{N}^r}$ of subgroups of the underlying abelian group of A such that $A = \oplus_{u \in \mathbb{N}^r} A_u$ and for all $a \in A_m$ and $b \in A_n$ we have $ab \in A_{m+n}$.
- Make the ring $A = k[x_1, \ldots, x_r]$ into an $\mathbb{N}^r$-graded ring by setting
$$A_u = kx_1^{u_1} \ldots x_r^{u_r} \text{ for all } u = (u_1, \ldots, u_r) \in \mathbb{N}^r.$$
- A module M over an $\mathbb{N}^r$-graded ring A is *graded* if there is a collection $\{M_i\}_{i \in \mathbb{N}^r}$ of subgroups of the underlying abelian group of M such that $M = \oplus_{i \in \mathbb{N}^r} M_i$ and for all $a \in A_j$ we have $aM_i \subset M_{i+j}$.

Persistence modules are modules

- Recall that an $\mathbb{N}^r$-graded (or multi-graded) ring is a ring A together with a collection $\{A_u\}_{u \in \mathbb{N}^r}$ of subgroups of the underlying abelian group of A such that $A = \oplus_{u \in \mathbb{N}^r} A_u$ and for all $a \in A_m$ and $b \in A_n$ we have $ab \in A_{m+n}$.
- Make the ring $A = k[x_1, \ldots, x_r]$ into an $\mathbb{N}^r$-graded ring by setting

$$A_u = k x_1^{u_1} \ldots x_r^{u_r} \text{ for all } u = (u_1, \ldots, u_r) \in \mathbb{N}^r.$$

- A module M over an $\mathbb{N}^r$-graded ring A is *graded* if there is a collection $\{M_i\}_{i \in \mathbb{N}^r}$ of subgroups of the underlying abelian group of M such that $M = \oplus_{i \in \mathbb{N}^r} M_i$ and for all $a \in A_j$ we have $aM_i \subset M_{i+j}$.

Correspondence Theorem of Persistent Homology (Carlsson, Zomorodian '09)
The functor category of r-parameter persistence modules is isomorphic to the category of graded $k[x_1, \ldots, x_r]$-modules and module homomorphisms respecting the grading.

Any persistence module is the homology of a filtered space

The homology of a multi-filtered space is a persistence module.

Any persistence module is the homology of a filtered space

The homology of a multi-filtered space is a persistence module.

On the other hand:

Theorem (Carlsson, Zomorodian, 2009)
For any finite persistence module M there exists a multi-filtered space K and a positive natural number i such that M is the homology in degree i of K.

Any persistence module is the homology of a filtered space

The homology of a multi-filtered space is a persistence module.

On the other hand:

Theorem (Carlsson, Zomorodian, 2009)
For any finite persistence module M there exists a multi-filtered space K and a positive natural number i such that M is the homology in degree i of K.

Therefore, studying the homology of r-filtered spaces amounts to studying graded modules over $k[x_1, \ldots, x_r]$.

Free resolutions and presentations

Let M be a finitely generated graded $k[x_1, \ldots, x_r]$-module. By the Hilbert Syzygy Theorem there is a free resolution by finitely generated $\mathbb{N}^r$-graded free $k[x_1, \ldots, x_r]$-modules of length at most r:

$$0 \longrightarrow F_m \xrightarrow{\phi_m} F_{m-1} \longrightarrow \ldots \longrightarrow F_1 \xrightarrow{\phi_1} F_0 \longrightarrow M \longrightarrow 0$$

with $\mathrm{image}(\phi_i) = \mathrm{kernel}(\phi_{i-1})$ and each F_i is a finitely generated graded free $k[x_1, \ldots, x_r]$-module and $m \leq r$.

Free resolutions and presentations

Let M be a finitely generated graded $k[x_1, \ldots, x_r]$-module. By the Hilbert Syzygy Theorem there is a free resolution by finitely generated $\mathbb{N}^r$-graded free $k[x_1, \ldots, x_r]$-modules of length at most r:

$$0 \longrightarrow F_m \xrightarrow{\phi_m} F_{m-1} \longrightarrow \ldots \longrightarrow F_1 \xrightarrow{\phi_1} F_0 \longrightarrow M \longrightarrow 0$$

with $\mathrm{image}(\phi_i) = \mathrm{kernel}(\phi_{i-1})$ and each F_i is a finitely generated graded free $k[x_1, \ldots, x_r]$-module and $m \leq r$.

The first part

$$F_1 \xrightarrow{\phi_1} F_0 \longrightarrow M \longrightarrow 0$$

of a free resolution of a module is called *presentation*. If we are given a presentation of M, we can then explicitly write M as the quotient $F_0/\mathrm{im}\phi_1$.

Minimal presentations and resolutions

Resolutions and presentations are in general not unique.

Minimal presentations and resolutions

Resolutions and presentations are in general not unique.

Example:[1] Let $M = (x_1x_2, x_1x_3) \subset k[x_1, x_2, x_3] = S$. The following are two free resolutions of M:

$$0 \longrightarrow S \xrightarrow{\begin{pmatrix} x_3 \\ -x_2 \end{pmatrix}} S^2 \xrightarrow{\begin{pmatrix} x_1x_2 & x_1x_3 \end{pmatrix}} M \longrightarrow 0$$

$$0 \xrightarrow{\begin{pmatrix} -x_2 \\ 1 \end{pmatrix}} S^2 \xrightarrow{\begin{pmatrix} x_3 & x_2x_3 \\ -x_2 & -x_2^2 \end{pmatrix}} S^2 \xrightarrow{\begin{pmatrix} x_1x_2 & x_1x_3 \end{pmatrix}} M \longrightarrow 0$$

However, minimal presentations of modules over local or graded rings are unique up to isomorphism.

[1] Bulletin of the AMS, July 2016

Invariants from resolutions and presentations

Minimal presentations are invariants of a module, and one can compute many invariants from minimal presentations and resolutions, such as:

- Betti numbers
- (Multi-graded) Hilbert series
- ...

Presentation of a persistence module: naïve Algorithm

Since the ith homology of the ith chain complex of a multi-filtered simplicial complex is defined as

$$H_i = \mathrm{kernel}(d_i)/\mathrm{image}(d_{i+1}),$$

an algorithm to compute a presentation of H_i is given by the following steps:

1. Compute a presentation of $\mathrm{image}(d_{i+1})$.

[1] G. Carlsson, G. Singh, A. Zomorodian, *Computing multidimensional persistence*, 2010.

Presentation of a persistence module: naïve Algorithm

Since the ith homology of the ith chain complex of a multi-filtered simplicial complex is defined as

$$H_i = \mathrm{kernel}(d_i)/\mathrm{image}(d_{i+1}),$$

an algorithm to compute a presentation of H_i is given by the following steps:

1. Compute a presentation of $\mathrm{image}(d_{i+1})$.
2. Compute a presentation of $\mathrm{kernel}(d_i)$.

[1] G. Carlsson, G. Singh, A. Zomorodian, *Computing multidimensional persistence*, 2010.

Presentation of a persistence module: naïve Algorithm

Since the ith homology of the ith chain complex of a multi-filtered simplicial complex is defined as

$$H_i = \mathrm{kernel}(d_i)/\mathrm{image}(d_{i+1}),$$

an algorithm to compute a presentation of H_i is given by the following steps:

1. Compute a presentation of $\mathrm{image}(d_{i+1})$.
2. Compute a presentation of $\mathrm{kernel}(d_i)$.
3. Compute a presentation of the quotient H_i.

Problem: the known algorithms to compute $\mathrm{image}(d_{i+1})$ are exponential in time and space[1].

[1] G. Carlsson, G. Singh, A. Zomorodian, *Computing multidimensional persistence*, 2010.

Presentation of a module: algorithm by Carlsson, Singh and Zomorodian

In 2010 Carlsson, Singh and Zomorodian[2] put forward an optimisation of the algorithm to compute a presentation of homology of 'one-critical' multi-filtered complexes.

[2] G. Carlsson, G. Singh, and A. Zomorodian, *Computing multidimensional persistence*, 2010.

Presentation of a module: algorithm by Carlsson, Singh and Zomorodian

In 2010 Carlsson, Singh and Zomorodian[2] put forward an optimisation of the algorithm to compute a presentation of homology of 'one-critical' multi-filtered complexes.

Let $K = \{K_u\}_{u \in \mathbb{N}^r}$ be a multi-filtered simplicial complex. We assume that there exists $v \in \mathbb{N}^r$ such that K_v is a finite simplicial complex, and $K_u = K_v$ for all $u \geq v$. We denote the simplicial complex K_v by K_{tot}.

[2] G. Carlsson, G. Singh, and A. Zomorodian, *Computing multidimensional persistence*, 2010.

Presentation of a module: algorithm by Carlsson, Singh and Zomorodian

In 2010 Carlsson, Singh and Zomorodian[2] put forward an optimisation of the algorithm to compute a presentation of homology of 'one-critical' multi-filtered complexes.

Let $K = \{K_u\}_{u \in \mathbb{N}^r}$ be a multi-filtered simplicial complex. We assume that there exists $v \in \mathbb{N}^r$ such that K_v is a finite simplicial complex, and $K_u = K_v$ for all $u \geq v$. We denote the simplicial complex K_v by K_{tot}.

For any $\sigma \in K_{\mathrm{tot}}$ define the set of *generators of* σ to be

$$\mathrm{gen}(\sigma) = \min\{v \in \mathbb{N}^r \mid \sigma \in K(v)\}.$$

[2] G. Carlsson, G. Singh, and A. Zomorodian, *Computing multidimensional persistence*, 2010.

Presentation of a module: algorithm by Carlsson, Singh and Zomorodian

In 2010 Carlsson, Singh and Zomorodian[2] put forward an optimisation of the algorithm to compute a presentation of homology of 'one-critical' multi-filtered complexes.

Let $K = \{K_u\}_{u \in \mathbb{N}^r}$ be a multi-filtered simplicial complex. We assume that there exists $v \in \mathbb{N}^r$ such that K_v is a finite simplicial complex, and $K_u = K_v$ for all $u \geq v$. We denote the simplicial complex K_v by K_{tot}.

For any $\sigma \in K_{\mathrm{tot}}$ define the set of *generators of* σ to be

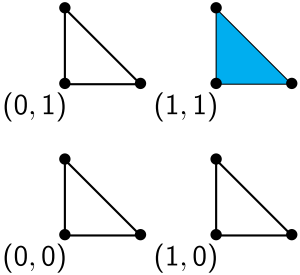
$$\mathrm{gen}(\sigma) = \min\{v \in \mathbb{N}^r \mid \sigma \in K(v)\}.$$

A multi-filtered simplicial complex is *one-critical* if the set of generators of every simplex has cardinality one.

[2] G. Carlsson, G. Singh, and A. Zomorodian, *Computing multidimensional persistence*, 2010.

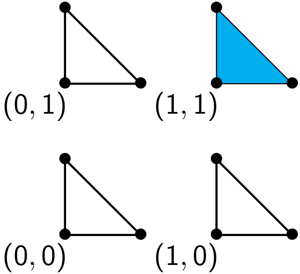
One-critical multi-filtered simplicial complex: example

One-critical:

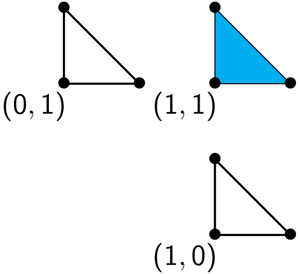


One-critical multi-filtered simplicial complex: example

One-critical:



Not one-critical:



Optimization

For a one-critical multifiltered simplicial complex K:

- the chain modules are free modules, hence one can choose bases for them
- The standard basis is the basis of simplices in degree given by their generator
- The boundary maps can be written as homogeneous matrices with monomial entries
- Carlsson, Singh and Zomorodian show that this gives a polynomial bound on complexity.
- The resulting presentation is not an invariant, as it depends on a choice of basis.

Presentation of a module: algorithm by Chacholski-Scolamiero-Vaccarino (CSV)

In 2014 Chacholski, Scolamiero and Vaccarino[3] put forward a polynomial-time algorithm to compute a presentation of homology of arbitrary multi-filtered simplicial complexes.

[3] W. Chacholski, M. Scolamiero, and F. Vaccarino, *Combinatorial presentation of multidimensional persistent homology*, 2014.

Presentation of a module: algorithm by Chacholski-Scolamiero-Vaccarino (CSV)

In 2014 Chacholski, Scolamiero and Vaccarino[3] put forward a polynomial-time algorithm to compute a presentation of homology of arbitrary multi-filtered simplicial complexes.

- For any $u \in \mathbb{N}^r$, denote by $K_{n,u}$ the set of n-simplices in K_u; the assignment $u \mapsto K_{n,u}$ induces a functor $K_n \colon \mathbb{N}^r \to \mathrm{Sets}$, where Sets is the category of sets.
- For any $v \in \mathbb{N}^r$ and any $i \in \{0, \ldots, n+1\}$ define the following map

$$d_i \colon K_{n+1,v} \longrightarrow K_{n,v} \colon \{x_0, \ldots, x_{n+1}\} \mapsto \{x_0, \ldots, \hat{x}_i, \ldots, x_{n+1}\}$$

  where $\hat{x}_i$ means that we omit the vertex x_i. The maps d_i give natural transformations $K_{n+1} \to K_n$.

[3] W. Chacholski, M. Scolamiero, and F. Vaccarino, *Combinatorial presentation of multidimensional persistent homology*, 2014.

CSV algorithm

Let $S = k[x_1, \ldots, x_r]$.There exists a sequence of free graded S-modules

$$\mathcal{RK}K_n \oplus \mathcal{RG}K_{n+1} \xrightarrow{\pi \oplus d} \mathcal{RG}K_n \xrightarrow{\alpha} \mathcal{R}D_{n-1}$$

such that $\alpha \circ (\pi \oplus d)$ is trivial, and the $k[x_1, \ldots, x_r]$-module $\mathrm{kernel}(\alpha)/\mathrm{im}(\pi \oplus d)$ is isomorphic to the homology in degree n of the multi-filtered simplicial complex K,

CSV algorithm

Let $S = k[x_1, \ldots, x_r]$.There exists a sequence of free graded S-modules

$$\mathcal{RK}K_n \oplus \mathcal{RG}K_{n+1} \xrightarrow{\pi \oplus d} \mathcal{RG}K_n \xrightarrow{\alpha} \mathcal{R}D_{n-1}$$

such that $\alpha \circ (\pi \oplus d)$ is trivial, and the $k[x_1, \ldots, x_r]$-module $\mathrm{kernel}(\alpha)/\mathrm{im}(\pi \oplus d)$ is isomorphic to the homology in degree n of the multi-filtered simplicial complex K, where

- $\mathcal{RK}K_n = \bigoplus_{\sigma \in K_{\mathrm{tot},n}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$,
- $\mathcal{RG}K_n = \bigoplus_{\sigma \in K_{\mathrm{tot},n}} \bigoplus_{v \in \mathrm{gen}(\sigma)} x^{v} S$, and
- $\mathcal{R}D_{n-1} = \bigoplus_{\sigma \in K_{\mathrm{tot},n-1}} S.$

with the notation $x^u := x_1^{u_1} \ldots x_r^{u_r}$.

CSV algorithm

The homomorphisms are defined as follows:

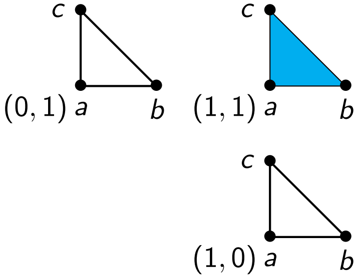
(π) For any $\sigma \in K_n$ and $v_0 \neq v_1 \in \mathrm{gen}(\sigma)$, the homomorphism $\pi\colon \mathcal{R}\mathcal{K}K_n \to \mathcal{R}\mathcal{G}K_n$ sends $x^{\max\{v_0,v_1\}}$ to $x^{v_0} - x^{v_1}$.

(d) For any $\sigma \in K_{n+1}$ and $v \in \mathrm{gen}(\sigma)$, the homomorphism $d\colon \mathcal{R}\mathcal{G}K_{n+1} \to \mathcal{R}\mathcal{G}K_n$ sends x^v to $\sum_{i=0}^{n+1}(-1)^i x^{\widetilde{d_i(\sigma)}}$, where $\widetilde{d_i(\sigma)}$ is the minimal element in the set $\{w \in \mathrm{gen}(d_i(\sigma)) \mid w \leq v\}$ with respect to the lexicographical order.

(α) For any $\sigma \in K_n$ and $v \in \mathrm{gen}(\sigma)$, the homomorphism $\alpha\colon \mathcal{R}\mathcal{G}K_n \to \mathcal{R}D_{n-1}$ sends x^v to $\sum_{i=0}^{n}(-1)^i d_i(\sigma)$.

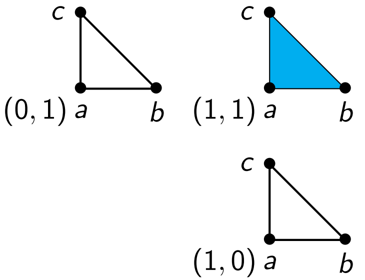
CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:



CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:

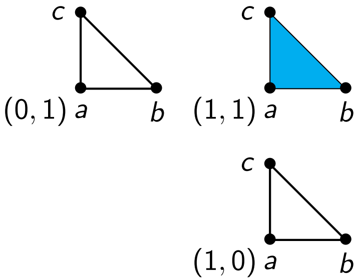


We have:

$$\mathrm{gen}(\{a\}) = \{(0,1),(1,0)\}$$

CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:

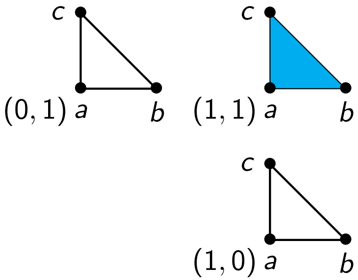


We have:

$\mathrm{gen}(\{a\}) = \{(0,1),(1,0)\}$
$\mathrm{gen}(\{b\}) = \{(0,1),(1,0)\}$

CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:

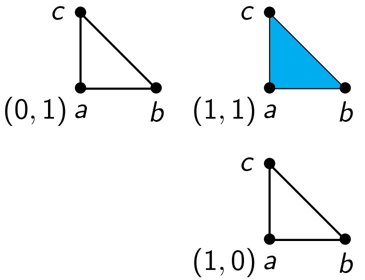


We have:

$$\mathrm{gen}(\{a\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{c\}) = \{(0,1),(1,0)\}$$

CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:

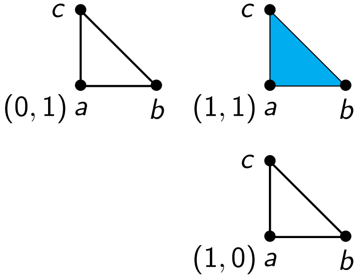


We have:

$$\mathrm{gen}(\{a\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{c\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{a,b\}) = \{(0,1),(1,0)\}$$

CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:

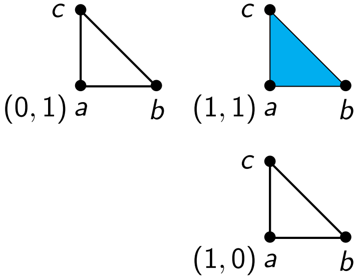


We have:

$$\mathrm{gen}(\{a\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{c\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{a,b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b,c\}) = \{(0,1),(1,0)\}$$

CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:

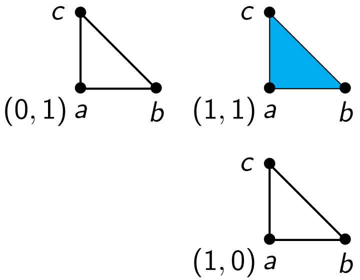


We have:

$$\mathrm{gen}(\{a\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{c\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{a,b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b,c\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{a,c\}) = \{(0,1),(1,0)\}$$

CSV algorithm: example

We illustrate the algorithm for the computation of H_1 of the following 2-filtered simplicial complex K:



We have:

$$\mathrm{gen}(\{a\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{c\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{a,b\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{b,c\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{a,c\}) = \{(0,1),(1,0)\}$$
$$\mathrm{gen}(\{a,b,c\}) = \{(1,1)\}$$

CSV algorithm: example

Let $S = k[x_1, x_2]$. Then:

- $\mathcal{RK}K_1 = \bigoplus_{\sigma \in K_{\mathrm{tot},1}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$

CSV algorithm: example

Let $S = k[x_1, x_2]$. Then:

- $\mathcal{RK}K_1 = \bigoplus_{\sigma \in K_{\mathrm{tot},1}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$, so
  $\mathcal{RK}K_1 = x_1x_2S \oplus x_1x_2S \oplus x_1x_2S$

CSV algorithm: example

Let $S = k[x_1, x_2]$. Then:

- $\mathcal{RK}K_1 = \bigoplus_{\sigma \in K_{\mathrm{tot},1}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$, so
  $\mathcal{RK}K_1 = x_1x_2S \oplus x_1x_2S \oplus x_1x_2S$
- $\mathcal{RG}K_n = \bigoplus_{\sigma \in K_{\mathrm{tot},n}} \bigoplus_{v \in \mathrm{gen}(\sigma)} x^v S$

CSV algorithm: example

Let $S = k[x_1, x_2]$. Then:

- $\mathcal{RK}K_1 = \bigoplus_{\sigma \in K_{\mathrm{tot},1}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$, so

  $\mathcal{RK}K_1 = x_1 x_2 S \oplus x_1 x_2 S \oplus x_1 x_2 S$

- $\mathcal{RG}K_n = \bigoplus_{\sigma \in K_{\mathrm{tot},n}} \bigoplus_{v \in \mathrm{gen}(\sigma)} x^v S$, so

  $\mathcal{RG}K_2 = x_1 x_2 S$

CSV algorithm: example

Let $S = k[x_1, x_2]$. Then:

- $\mathcal{RK}K_1 = \bigoplus_{\sigma \in K_{\mathrm{tot},1}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$, so
  $\mathcal{RK}K_1 = x_1x_2S \oplus x_1x_2S \oplus x_1x_2S$
- $\mathcal{RG}K_n = \bigoplus_{\sigma \in K_{\mathrm{tot},n}} \bigoplus_{v \in \mathrm{gen}(\sigma)} x^v S$, so
  $\mathcal{RG}K_2 = x_1x_2S$
  $\mathcal{RG}K_1 = x_1S \oplus x_2S \oplus x_1S \oplus x_2S \oplus x_1S \oplus x_2S$

CSV algorithm: example

Let $S = k[x_1, x_2]$. Then:

- $\mathcal{R}\mathcal{K}K_1 = \bigoplus_{\sigma \in K_{\mathrm{tot},1}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$, so

  $\mathcal{R}\mathcal{K}K_1 = x_1x_2S \oplus x_1x_2S \oplus x_1x_2S$
- $\mathcal{R}\mathcal{G}K_n = \bigoplus_{\sigma \in K_{\mathrm{tot},n}} \bigoplus_{v \in \mathrm{gen}(\sigma)} x^v S$, so

  $\mathcal{R}\mathcal{G}K_2 = x_1x_2S$

  $\mathcal{R}\mathcal{G}K_1 = x_1S \oplus x_2S \oplus x_1S \oplus x_2S \oplus x_1S \oplus x_2S$
- $\mathcal{R}D_0 = \bigoplus_{\sigma \in K_{\mathrm{tot},0}} k[x_1, \ldots, x_r]$, so

CSV algorithm: example

Let $S = k[x_1, x_2]$. Then:

- $\mathcal{RK}K_1 = \bigoplus_{\sigma \in K_{\mathrm{tot},1}} \bigoplus_{v_0 \neq v_1 \in \mathrm{gen}(\sigma)} x^{\max\{v_0, v_1\}} S$, so

  $\mathcal{RK}K_1 = x_1 x_2 S \oplus x_1 x_2 S \oplus x_1 x_2 S$
- $\mathcal{RG}K_n = \bigoplus_{\sigma \in K_{\mathrm{tot},n}} \bigoplus_{v \in \mathrm{gen}(\sigma)} x^v S$, so

  $\mathcal{RG}K_2 = x_1 x_2 S$

  $\mathcal{RG}K_1 = x_1 S \oplus x_2 S \oplus x_1 S \oplus x_2 S \oplus x_1 S \oplus x_2 S$
- $\mathcal{R}D_0 = \bigoplus_{\sigma \in K_{\mathrm{tot},0}} k[x_1, \ldots, x_r]$, so

  $\mathcal{R}DK_0 = S \oplus S \oplus S$

CSV algorithm: example

- $\pi\colon \mathcal{RK}K_1 \to \mathcal{RG}K_1$ sends $x^{\max\{v_0,v_1\}}$ to $x^{v_0} - x^{v_1}$, so

CSV algorithm: example

- $\pi\colon \mathcal{RK}K_1 \to \mathcal{RG}K_1$ sends $x^{\max\{v_0,v_1\}}$ to $x^{v_0} - x^{v_1}$, so

$$\pi = \begin{pmatrix} x_1 & 0 & 0 \\ -x_2 & 0 & 0 \\ 0 & x_1 & 0 \\ 0 & -x_2 & 0 \\ 0 & 0 & x_1 \\ 0 & 0 & -x_2 \end{pmatrix}$$

CSV algorithm: example

- $\pi \colon \mathcal{RKK}_1 \to \mathcal{RGK}_1$ sends $x^{\max\{v_0,v_1\}}$ to $x^{v_0} - x^{v_1}$, so

$$\pi = \begin{pmatrix} x_1 & 0 & 0 \\ -x_2 & 0 & 0 \\ 0 & x_1 & 0 \\ 0 & -x_2 & 0 \\ 0 & 0 & x_1 \\ 0 & 0 & -x_2 \end{pmatrix}$$

- $d \colon \mathcal{RGK}_2 \to \mathcal{RGK}_1$ sends x^v to $\sum_{i=0}^{n+1} (-1)^i x^{\widetilde{d_i(\sigma)}}$

CSV algorithm: example

- $\pi\colon \mathcal{R}\mathcal{K}K_1 \to \mathcal{R}\mathcal{G}K_1$ sends $x^{\max\{v_0,v_1\}}$ to $x^{v_0} - x^{v_1}$, so

$$\pi = \begin{pmatrix} x_1 & 0 & 0 \\ -x_2 & 0 & 0 \\ 0 & x_1 & 0 \\ 0 & -x_2 & 0 \\ 0 & 0 & x_1 \\ 0 & 0 & -x_2 \end{pmatrix}$$

- $d\colon \mathcal{R}\mathcal{G}K_2 \to \mathcal{R}\mathcal{G}K_1$ sends x^v to $\sum_{i=0}^{n+1}(-1)^i x^{\widetilde{d_i(\sigma)}}$, so

$$d = \begin{pmatrix} 0 & -x_2 & 0 & -x_2 & 0 & -x_2 \end{pmatrix}^t$$

CSV algorithm: example

- $\pi\colon \mathcal{RK}K_1 \to \mathcal{RG}K_1$ sends $x^{\max\{v_0,v_1\}}$ to $x^{v_0} - x^{v_1}$, so

$$\pi = \begin{pmatrix} x_1 & 0 & 0 \\ -x_2 & 0 & 0 \\ 0 & x_1 & 0 \\ 0 & -x_2 & 0 \\ 0 & 0 & x_1 \\ 0 & 0 & -x_2 \end{pmatrix}$$

- $d\colon \mathcal{RG}K_2 \to \mathcal{RG}K_1$ sends x^v to $\sum_{i=0}^{n+1}(-1)^i x^{\widetilde{d_i(\sigma)}}$, so

$$d = \begin{pmatrix} 0 & -x_2 & 0 & -x_2 & 0 & -x_2 \end{pmatrix}^t$$

- $\alpha\colon \mathcal{RG}K_1 \to \mathcal{R}D_0$ sends x^v to $\sum_{i=0}^{n}(-1)^i d_i(\sigma)$

CSV algorithm: example

- $\pi\colon \mathcal{RKK}_1 \to \mathcal{RGK}_1$ sends $x^{\max\{v_0,v_1\}}$ to $x^{v_0} - x^{v_1}$, so

$$\pi = \begin{pmatrix} x_1 & 0 & 0 \\ -x_2 & 0 & 0 \\ 0 & x_1 & 0 \\ 0 & -x_2 & 0 \\ 0 & 0 & x_1 \\ 0 & 0 & -x_2 \end{pmatrix}$$

- $d\colon \mathcal{RGK}_2 \to \mathcal{RGK}_1$ sends x^v to $\sum_{i=0}^{n+1}(-1)^i x^{\widetilde{d_i(\sigma)}}$, so

$$d = \begin{pmatrix} 0 & -x_2 & 0 & -x_2 & 0 & -x_2 \end{pmatrix}^t$$

- $\alpha\colon \mathcal{RGK}_1 \to \mathcal{R}D_0$ sends x^v to $\sum_{i=0}^{n}(-1)^i d_i(\sigma)$, so

$$\alpha = \begin{pmatrix} -1 & -1 & -1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

CSV algorithm: example

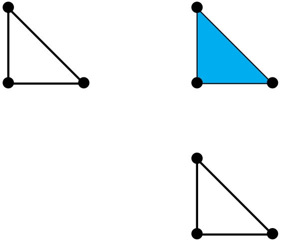
By using a computational algebra software package one can then compute the following minimal presentation:

$$0 \longrightarrow S^2 \xrightarrow{\begin{pmatrix} x_1 & 0 \\ 0 & x_2 \end{pmatrix}} S^2 \xrightarrow{\begin{pmatrix} x_2 & x_1 \end{pmatrix}} H_1(K) \longrightarrow 0$$

and thus

$$H_1(K) = \frac{x_1 S}{(x_1 x_2)} \oplus \frac{x_2 S}{(x_1 x_2)}.$$

CSV algorithm: example



$$H_1(K) = \frac{x_1 S}{(x_1 x_2)} \oplus \frac{x_2 S}{(x_1 x_2)}.$$

Conclusions

Conclusions

- Need efficient implementation of algorithm by Chacholski, Scolamiero and Vaccarino.
- Computational algebra libraries are not efficient.
- How complex is the problem in practice?
- Insight from geometric invariant theory?