Ripser

Efficient Computation of Vietoris–Rips Persistence Barcodes

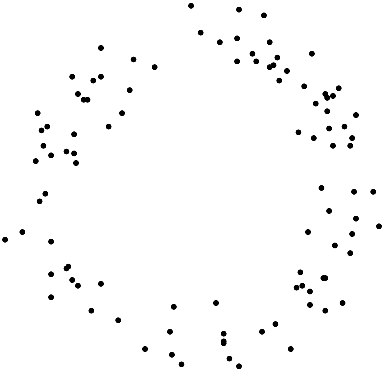
Ulrich Bauer

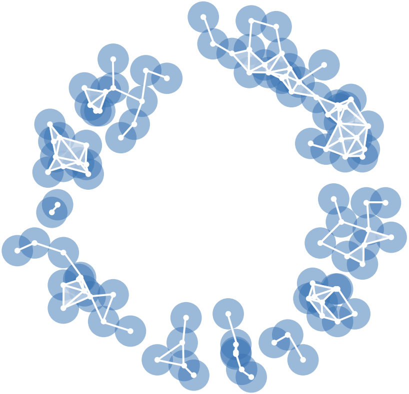
TUM

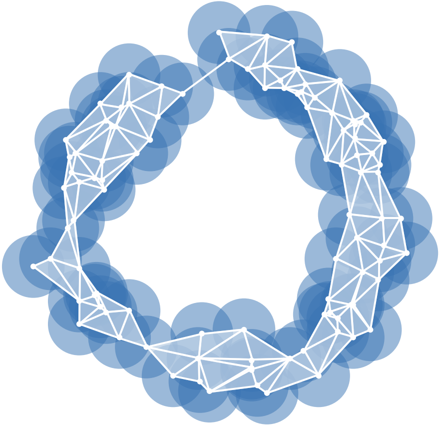
May 2, 2017

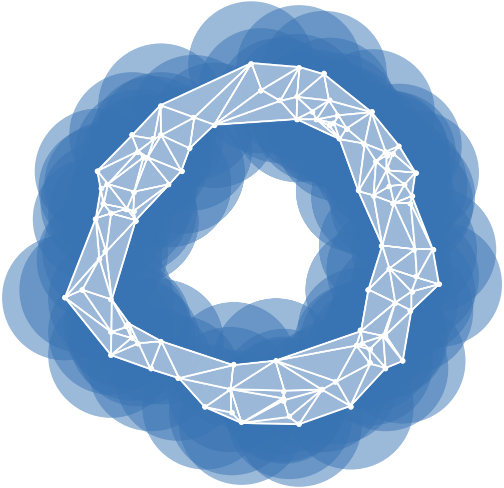
Special Hausdorff Program *Applied and Computational Algebraic Topology*
Hausdorff Research Institute for Mathematics, Bonn

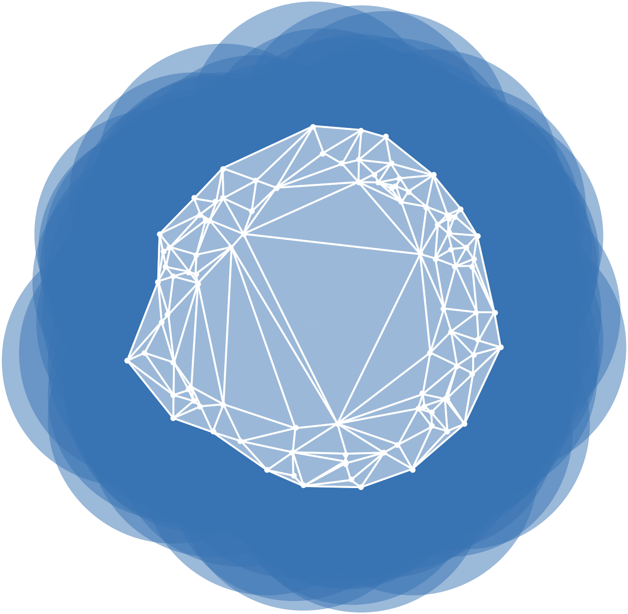
Persistent homology

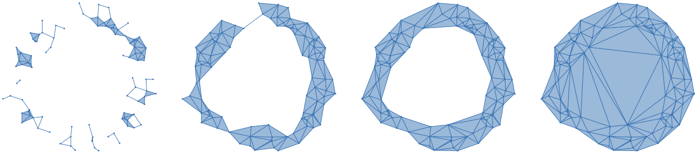


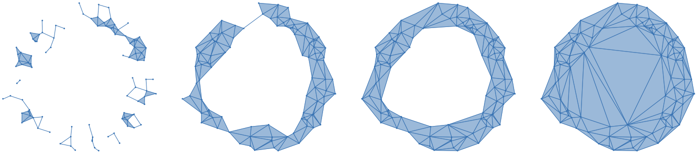


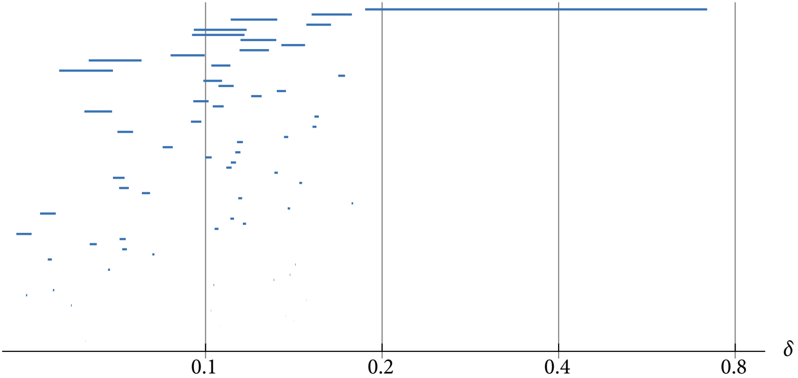




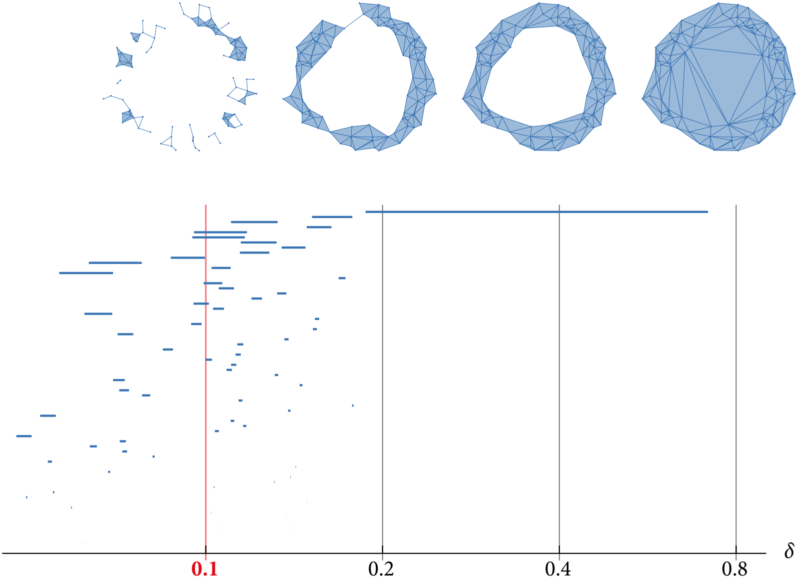




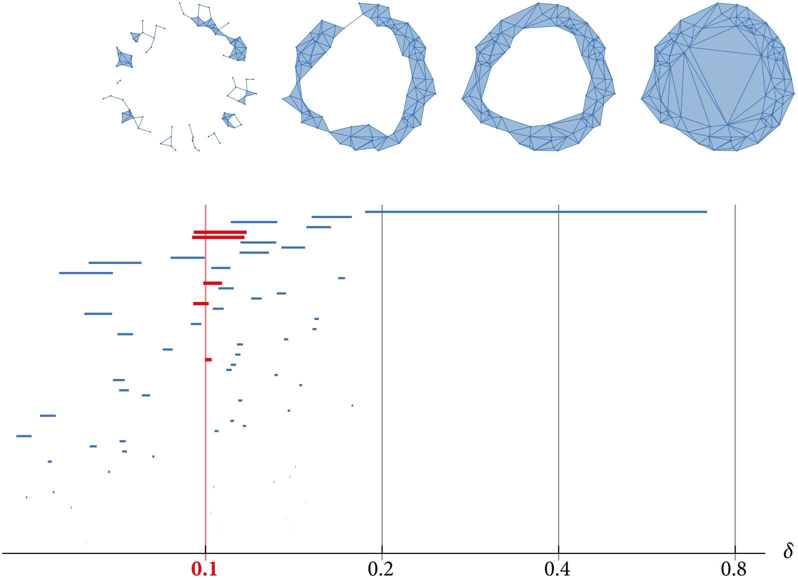


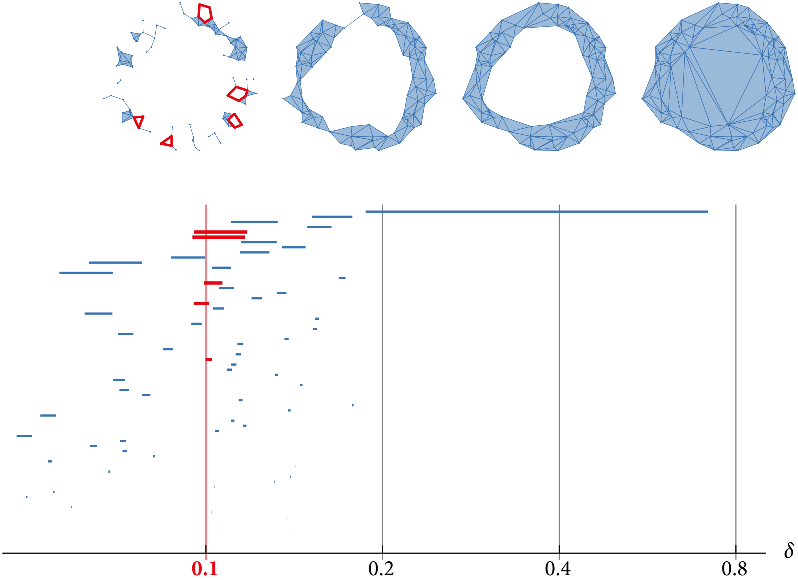


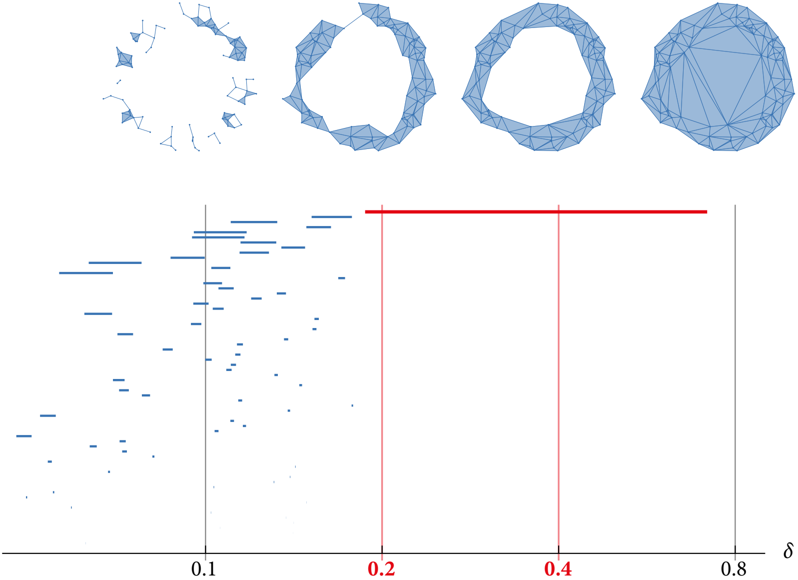
0.1
0.2
0.4
0.8
δ



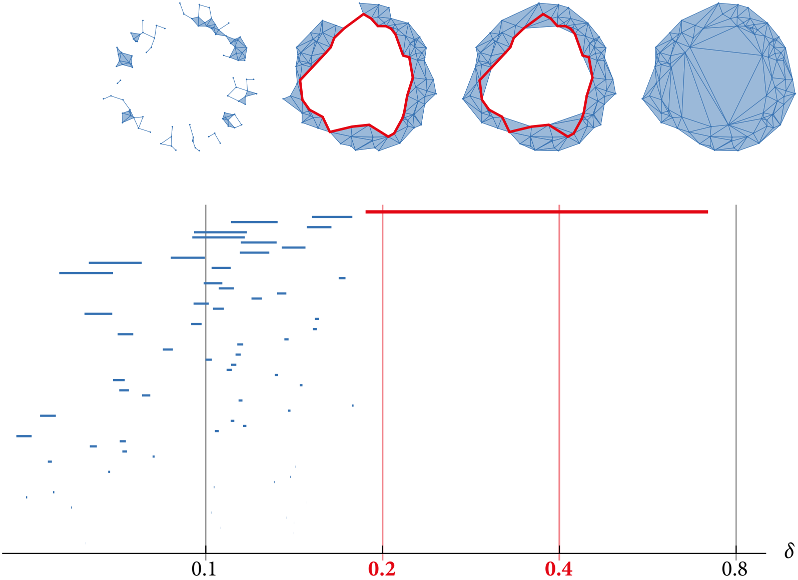
0.1
0.2
0.4
0.8
δ


0.1
0.2
0.4
0.8
δ


δ
0.1
0.2
0.4
0.8



0.1
0.2
0.4
0.8
δ



0.1
0.2
0.4
0.8
δ

Vietoris–Rips persistence

Vietoris–Rips filtrations

Consider a finite metric space (X, d).
The *Vietoris–Rips complex* is the simplicial complex

$$\mathrm{Rips}_t(X) = \{S \subseteq X \mid \operatorname{diam} S \leq t\}$$

- 1-skeleton: all edges with pairwise distance $\leq t$
- all possible higher simplices (flag complex)

Vietoris–Rips filtrations

Consider a finite metric space (X, d).
The *Vietoris–Rips complex* is the simplicial complex

$$\mathrm{Rips}_t(X) = \{S \subseteq X \mid \operatorname{diam} S \leq t\}$$

- 1-skeleton: all edges with pairwise distance $\leq t$
- all possible higher simplices (flag complex)

Goal:

- compute persistence barcodes for $H_d(\mathrm{Rips}_t(X))$ (in dimensions $0 \leq d \leq k$)

Demo: Ripser

Example data set:

- 192 points on $\mathbb{S}^2$
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Demo: Ripser

Example data set:

- 192 points on $\mathbb{S}^2$
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Comparison with other software:

- javaplex: 3200 seconds, 12 GB
- Dionysus: 533 seconds, 3.4 GB
- GUDHI: 75 seconds, 2.9 GB
- DIPHA: 50 seconds, 6 GB
- Eirene: 12 seconds, 1.5 GB

Demo: Ripser

Example data set:

- 192 points on $\mathbb{S}^2$
- persistent homology barcodes up to dimension 2
- over 56 mio. simplices in 3-skeleton

Comparison with other software:

- javaplex: 3200 seconds, 12 GB
- Dionysus: 533 seconds, 3.4 GB
- GUDHI: 75 seconds, 2.9 GB
- DIPHA: 50 seconds, 6 GB
- Eirene: 12 seconds, 1.5 GB

Ripser: 1.2 seconds, 152 MB

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies
- support for
  - coefficients in a prime field $\mathbb{F}_p$

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies
- support for
  - coefficients in a prime field $\mathbb{F}_p$
  - sparse distance matrices for distance threshold

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies
- support for
  - coefficients in a prime field $\mathbb{F}_p$
  - sparse distance matrices for distance threshold
- open source (`http://ripser.org`)
  - released in July 2016

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies
- support for
  - coefficients in a prime field $\mathbb{F}_p$
  - sparse distance matrices for distance threshold
- open source (`http://ripser.org`)
  - released in July 2016
- online version (`http://live.ripser.org`)
  - launched in August 2016

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies
- support for
  - coefficients in a prime field $\mathbb{F}_p$
  - sparse distance matrices for distance threshold
- open source (`http://ripser.org`)
  - released in July 2016
- online version (`http://live.ripser.org`)
  - launched in August 2016
- most efficient software for Vietoris–Rips persistence
  - computes H^2 barcode for $50\,000$ random points on a torus in 136 seconds / 9 GB (using distance threshold)

Ripser

A software for computing Vietoris–Rips persistence barcodes

- about 1000 lines of C++ code, no external dependencies
- support for
  - coefficients in a prime field $\mathbb{F}_p$
  - sparse distance matrices for distance threshold
- open source (`http://ripser.org`)
  - released in July 2016
- online version (`http://live.ripser.org`)
  - launched in August 2016
- most efficient software for Vietoris–Rips persistence
  - computes H^2 barcode for $50\,000$ random points on a torus in 136 seconds / 9 GB (using distance threshold)
- 2016 ATMCS Best New Software Award (jointly with RIVET)

Design goals

Goals for previous projects:

- PHAT [B, Kerber, Reininghaus, Wagner 2013]:
  fast persistence computation (matrix reduction only)
- DIPHA [B, Kerber, Reininghaus 2014]:
  distributed persistence computation

Design goals

Goals for previous projects:

- PHAT [B, Kerber, Reininghaus, Wagner 2013]:
  fast persistence computation (matrix reduction only)
- DIPHA [B, Kerber, Reininghaus 2014]:
  distributed persistence computation

Goals for Ripser:

- Use as little memory as possible
- Be reasonable about computation time

The four special ingredients

The improved performance is based on 4 insights:

- Clearing inessential columns [Chen, Kerber 2011]
- Computing cohomology [de Silva et al. 2011]
- Implicit matrix reduction
- Apparent and emergent pairs

The four special ingredients

The improved performance is based on 4 insights:

- Clearing inessential columns [Chen, Kerber 2011]
- Computing cohomology [de Silva et al. 2011]
- Implicit matrix reduction
- Apparent and emergent pairs

Lessons from PHAT:

- Clearing and cohomology yield considerable speedup,
- but only when *both* are used in conjuction!

Matrix reduction

Matrix reduction algorithm

Setting:

- finite metric space X, n points
- persistent homology $H_d(\mathrm{Rips}_t(X); \mathbb{F}_2)$ in dimensions $d \leq k$

Notation:

- D: boundary matrix of filtration
- R_i: ith column of R

Matrix reduction algorithm

Setting:

- finite metric space X, n points
- persistent homology $H_d(\operatorname{Rips}_t(X); \mathbb{F}_2)$ in dimensions $d \leq k$

Notation:

- D: boundary matrix of filtration
- R_i: ith column of R

Algorithm:

- $R = D$, $V = I$
- while $\exists i < j$ with $\operatorname{pivot} R_i = \operatorname{pivot} R_j$
  - add R_i to R_j, add V_i to V_j

Matrix reduction algorithm

Setting:

- finite metric space X, n points
- persistent homology $H_d(\mathrm{Rips}_t(X); \mathbb{F}_2)$ in dimensions $d \leq k$

Notation:

- D: boundary matrix of filtration
- R_i: ith column of R

Algorithm:

- $R = D$, $V = I$
- while $\exists i < j$ with $\operatorname{pivot} R_i = \operatorname{pivot} R_j$
  - add R_i to R_j, add V_i to V_j

Result:

- $R = D \cdot V$ is reduced (unique pivots)
- V is full rank upper triangular

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$P = \{i : R_i = 0\} \qquad \textit{positive indices,}$$

$$N = \{j : R_j \neq 0\} \qquad \textit{negative indices,}$$

$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential indices.}$$

Then

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$P = \{i : R_i = 0\}$ — *positive* indices,

$N = \{j : R_j \neq 0\}$ — *negative* indices,

$E = P \setminus \operatorname{pivots} R$ — *essential* indices.

Then

$\widetilde{\Sigma}_Z = \{\, V_i \mid i \in P\}$ is a basis of Z_*,

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$P = \{i : R_i = 0\}$ — *positive* indices,

$N = \{j : R_j \neq 0\}$ — *negative* indices,

$E = P \setminus \operatorname{pivots} R$ — *essential* indices.

Then

$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\}$ is a basis of Z_*,

$\Sigma_B = \{R_j \mid j \in N\}$ is a basis of B_*,

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$P = \{i : R_i = 0\} \qquad \textit{positive} \text{ indices,}$$

$$N = \{j : R_j \neq 0\} \qquad \textit{negative} \text{ indices,}$$

$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential} \text{ indices.}$$

Then

$$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\} \qquad \text{is a basis of } Z_*,$$

$$\Sigma_B = \{R_j \mid j \in N\} \qquad \text{is a basis of } B_*,$$

$$\Sigma_Z = \Sigma_B \cup \{V_i \mid i \in E\} \qquad \text{is } \textit{another} \text{ basis of } Z_*.$$

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$$P = \{i : R_i = 0\} \qquad \textit{positive indices},$$

$$N = \{j : R_j \neq 0\} \qquad \textit{negative indices},$$

$$E = P \setminus \operatorname{pivots} R \qquad \textit{essential indices}.$$

Then

$$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\} \qquad \text{is a basis of } Z_*,$$

$$\Sigma_B = \{R_j \mid j \in N\} \qquad \text{is a basis of } B_*,$$

$$\Sigma_Z = \Sigma_B \cup \{V_i \mid i \in E\} \qquad \text{is \textit{another} basis of } Z_*.$$

Persistent homology is generated by the basis cycles Σ_Z.

Compatible basis cycles

For a reduced boundary matrix $R = D \cdot V$, call

$P = \{i : R_i = 0\}$	*positive* indices,
$N = \{j : R_j \neq 0\}$	*negative* indices,
$E = P \smallsetminus \operatorname{pivots} R$	*essential* indices.

Then

$\widetilde{\Sigma}_Z = \{V_i \mid i \in P\}$	is a basis of Z_*,
$\Sigma_B = \{R_j \mid j \in N\}$	is a basis of B_*,
$\Sigma_Z = \Sigma_B \cup \{V_i \mid i \in E\}$	is *another* basis of Z_*.

Persistent homology is generated by the basis cycles Σ_Z.

- Persistence intervals: $\{[i, j) \mid i = \operatorname{pivot} R_j\} \cup \{[i, \infty) \mid i \in E\}$
- Columns with non-essential positive indices never used!

Clearing

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential positive indices
- Reduce boundary matrices of $\partial_d : C_d \to C_{d-1}$ in decreasing dimension $d = k + 1, \ldots, 1$
- Whenever $i = \operatorname{pivot} R_j$ (in matrix for ∂_d)
    - Set R_i to 0 (in matrix for ∂_{d-1})

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential positive indices
- Reduce boundary matrices of $\partial_d : C_d \to C_{d-1}$ in decreasing dimension $d = k+1, \dots, 1$
- Whenever $i = \operatorname{pivot} R_j$ (in matrix for ∂_d)
  - Set R_i to 0 (in matrix for ∂_{d-1})
  - Set V_i to R_j

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential positive indices
- Reduce boundary matrices of $\partial_d : C_d \to C_{d-1}$ in decreasing dimension $d = k+1, \dots, 1$
- Whenever $i = \operatorname{pivot} R_j$ (in matrix for ∂_d)
  - Set R_i to 0 (in matrix for ∂_{d-1})
  - Set V_i to R_j
- Still yields $R = D \cdot V$ reduced, V full rank upper triangular

Clearing non-essential positive columns

Idea [Chen, Kerber 2011]:

- Don't reduce at non-essential positive indices
- Reduce boundary matrices of $\partial_d : C_d \to C_{d-1}$
  in decreasing dimension $d = k + 1, \ldots, 1$
- Whenever $i = \operatorname{pivot} R_j$ (in matrix for ∂_d)
    - Set R_i to 0 (in matrix for ∂_{d-1})
    - Set V_i to R_j
- Still yields $R = D \cdot V$ reduced, V full rank upper triangular

Note:

- reducing *positive* columns typically harder than negative
- with clearing: need only reduce *essential* positive columns

Cohomology

Persistent cohomology

We have seen: many columns of $R = D \cdot V$ are not needed

- Skip those inessential columns in matrix reduction

Persistent cohomology

We have seen: many columns of $R = D \cdot V$ are not needed

- Skip those inessential columns in matrix reduction

For persistence barcodes in low dimensions $d \leq k$:

- Number of skipped indices for reducing D^T (cohomology) is much larger than for D (homology)
  - reducing boundary matrix produces basis for $H_{k+1}(K_{k+1})$, which is not needed
- The resulting persistence barcode is the same [de Silva et al. 2011]

Counting homology column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k+1} \underbrace{\binom{n}{d+1}}_{\dim C_d(K)} = \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d+1}}_{\dim Z_d(K)}$$

Counting homology column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k+1} \underbrace{\binom{n}{d+1}}_{\dim C_d(K)} = \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d+1}}_{\dim Z_d(K)}$$

$k = 2, n = 192$: $\quad 56\,050\,096 = 1\,161\,471 + 54\,888\,625$

Counting homology column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k+1} \underbrace{\binom{n}{d+1}}_{\dim C_d(K)} = \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d+1}}_{\dim Z_d(K)}$$

$k = 2, n = 192: \quad 56\,050\,096 = 1\,161\,471 + 54\,888\,625$

- using clearing:

$$\sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \underbrace{\binom{n-1}{k+2}}_{\dim H_{k+1}(K)} = \sum_{d=1}^{k+2} \binom{n-1}{d}$$

Counting homology column reductions

- standard matrix reduction:

$$\sum_{d=1}^{k+1} \underbrace{\binom{n}{d+1}}_{\dim C_d(K)} = \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d+1}}_{\dim Z_d(K)}$$

$$k = 2, n = 192: \quad 56\,050\,096 = 1\,161\,471 + 54\,888\,625$$

- using clearing:

$$\sum_{d=1}^{k+1} \underbrace{\binom{n-1}{d}}_{\dim B_{d-1}(K)} + \underbrace{\binom{n-1}{k+2}}_{\dim H_{k+1}(K)} = \sum_{d=1}^{k+2} \binom{n-1}{d}$$

$$k = 2, n = 192: \quad 54\,888\,816 = 1\,161\,471 + 53\,727\,345$$

Counting cohomology column reductions

- standard matrix reduction:

$$\sum_{d=0}^{k} \underbrace{\binom{n}{d+1}}_{\dim C^d(K)} = \sum_{d=0}^{k} \underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \sum_{d=0}^{k} \underbrace{\binom{n-1}{d}}_{\dim Z^d(K)}$$

Counting cohomology column reductions

- standard matrix reduction:

$$\sum_{d=0}^{k} \underbrace{\binom{n}{d+1}}_{\dim C^d(K)} = \sum_{d=0}^{k} \underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \sum_{d=0}^{k} \underbrace{\binom{n-1}{d}}_{\dim Z^d(K)}$$

$k = 2, n = 192: \quad 1\,179\,808 = 18\,337 + 1\,161\,471$

Counting cohomology column reductions

- standard matrix reduction:

$$\sum_{d=0}^{k} \underbrace{\binom{n}{d+1}}_{\dim C^d(K)} = \sum_{d=0}^{k} \underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \sum_{d=0}^{k} \underbrace{\binom{n-1}{d}}_{\dim Z^d(K)}$$

$k = 2, n = 192$: $\quad 1\,179\,808 = 18\,337 + 1\,161\,471$

- using clearing:

$$\sum_{d=0}^{k} \underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \underbrace{\binom{n-1}{0}}_{\dim H^0(K)} + = \sum_{d=0}^{k+1} \binom{n-1}{d}$$

Counting cohomology column reductions

- standard matrix reduction:

$$\underbrace{\sum_{d=0}^{k}\binom{n}{d+1}}_{\dim C^d(K)} = \sum_{d=0}^{k}\underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \sum_{d=0}^{k}\underbrace{\binom{n-1}{d}}_{\dim Z^d(K)}$$

$k = 2, n = 192$: $\quad 1\,179\,808 = 18\,337 + 1\,161\,471$

- using clearing:

$$\sum_{d=0}^{k}\underbrace{\binom{n-1}{d+1}}_{\dim B^{d+1}(K)} + \underbrace{\binom{n-1}{0}}_{\dim H^0(K)} + = \sum_{d=0}^{k+1}\binom{n-1}{d}$$

$k = 2, n = 192$: $\quad 1\,161\,472 = 1 + 1\,161\,471$

Observations

For a typical input:

- V has very few off-diagonal entries
- most negative columns of D are already reduced

Observations

For a typical input:

- V has very few off-diagonal entries
- most negative columns of D are already reduced

Previous example ($k = 2, n = 192$):

- Only 845 out of $1\,161\,471$ columns have to be reduced

Implicit matrix reduction

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
  - Explicitly generated and stored in memory

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
  - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
  - transform D into R by column operations

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
  - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
  - transform D into R by column operations

Approach for Ripser:

- Boundary matrix D for lexicographically ordered basis
  - Implicitly defined and recomputed when needed

Implicit matrix reduction

Standard approach:

- Boundary matrix D for filtration-ordered basis
    - Explicitly generated and stored in memory
- Matrix reduction: store only reduced matrix R
    - transform D into R by column operations

Approach for Ripser:

- Boundary matrix D for lexicographically ordered basis
    - Implicitly defined and recomputed when needed
- Matrix reduction in Ripser: store only coefficient matrix V
    - recompute previous columns of $R = D \cdot V$ when needed
    - Typically, V is much sparser and smaller than R

Oblivious matrix reduction

Algorithm variant:

- $R = D$
- for $j = 1, \ldots, n$
  - while $\exists i < j$ with $\operatorname{pivot} R_i = \operatorname{pivot} R_j$
    - add D_i to R_j

Oblivious matrix reduction

Algorithm variant:

- $R = D$
- for $j = 1, \ldots, n$
  - while $\exists i < j$ with $\operatorname{pivot} R_i = \operatorname{pivot} R_j$
    - add D_i to R_j

Computing the persistence intervals requires only:

- current column R_j
- pivots of previous columns R_i

Oblivious matrix reduction

Algorithm variant:

- $R = D$
- for $j = 1, \ldots, n$
    - while $\exists i < j$ with $\operatorname{pivot} R_i = \operatorname{pivot} R_j$
        - add D_i to R_j

Computing the persistence intervals requires only:

- current column R_j
- pivots of previous columns R_i

Corollary

The rank of an $m \times n$ matrix can be computed in $O(n)$ memory.

Apparent and emergent pairs

Natural filtration settings

Typical assumptions on the filtration:

• general filtration	persistence (in theory)
• filtration by singletons or pairs	discrete Morse theory
• simplexwise filtration	persistence (computation)

Natural filtration settings

Typical assumptions on the filtration:

• general filtration	persistence (in theory)
• filtration by singletons or pairs	discrete Morse theory
• simplexwise filtration	persistence (computation)

Conclusion:

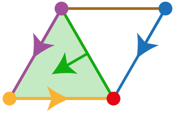
- Discrete Morse theory sits in the middle
  between persistence and persistence (!)

Discrete Morse theory

Definition (Forman 1998)

A *discrete vector field* on a cell complex is a partition of the set of cells into

- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.

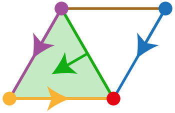


Discrete Morse theory

Definition (Forman 1998)

A *discrete vector field* on a cell complex is a partition of the set of cells into

- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.



A function $f : K \to \mathbb{R}$ on a cell complex is a *discrete Morse function* if

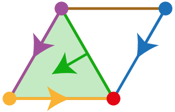
- sublevel sets are subcomplexes, and

Discrete Morse theory

Definition (Forman 1998)

A *discrete vector field* on a cell complex is a partition of the set of cells into

- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.



A function $f : K \to \mathbb{R}$ on a cell complex is a *discrete Morse function* if

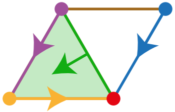
- sublevel sets are subcomplexes, and

Discrete Morse theory

Definition (Forman 1998)

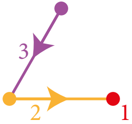
A *discrete vector field* on a cell complex is a partition of the set of cells into

- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.



A function $f : K \to \mathbb{R}$ on a cell complex is a *discrete Morse function* if

- sublevel sets are subcomplexes, and

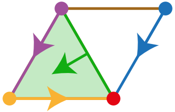


Discrete Morse theory

Definition (Forman 1998)

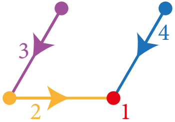
A *discrete vector field* on a cell complex is a partition of the set of cells into

- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.



A function $f : K \to \mathbb{R}$ on a cell complex is a *discrete Morse function* if

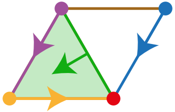
- sublevel sets are subcomplexes, and



Discrete Morse theory

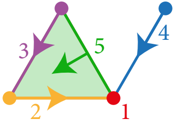
Definition (Forman 1998)

A *discrete vector field* on a cell complex is a partition of the set of cells into



- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.

A function $f : K \to \mathbb{R}$ on a cell complex is a *discrete Morse function* if



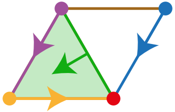
- sublevel sets are subcomplexes, and

Discrete Morse theory

Definition (Forman 1998)

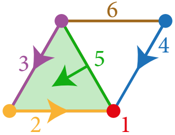
A *discrete vector field* on a cell complex is a partition of the set of cells into

- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.



A function $f : K \to \mathbb{R}$ on a cell complex is a *discrete Morse function* if

- sublevel sets are subcomplexes, and

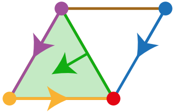


Discrete Morse theory

Definition (Forman 1998)

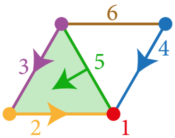
A *discrete vector field* on a cell complex is a partition of the set of cells into

- singleton sets $\{\phi\}$ *(critical cells)*, and
- pairs $\{\sigma, \tau\}$, where σ is a facet of τ.



A function $f : K \to \mathbb{R}$ on a cell complex is a *discrete Morse function* if

- sublevel sets are subcomplexes, and
- level sets form a discrete vector field.



Fundamental theorem of discrete Morse theory

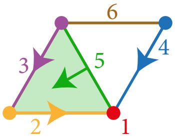
Let f be a discrete Morse function on a cell complex K.

Fundamental theorem of discrete Morse theory

Let f be a discrete Morse function on a cell complex K.

Theorem (Forman 1998)

If $(s, t]$ *contains no critical value of* f*, then the sublevel set* K_t collapses to K_s .

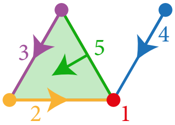


Fundamental theorem of discrete Morse theory

Let f be a discrete Morse function on a cell complex K.

Theorem (Forman 1998)

If $(s, t]$ *contains no critical value of* f*, then the sublevel set* K_t collapses to K_s.

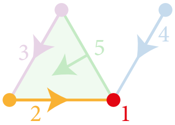


Fundamental theorem of discrete Morse theory

Let f be a discrete Morse function on a cell complex K.

Theorem (Forman 1998)

If $(s, t]$ contains no critical value of f, then the sublevel set K_t collapses to K_s.

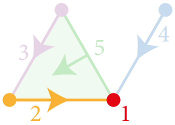


Fundamental theorem of discrete Morse theory

Let f be a discrete Morse function on a cell complex K.

Theorem (Forman 1998)

If $(s, t]$ *contains no critical value of* f*, then the sublevel set* K_t collapses to K_s.



Corollary

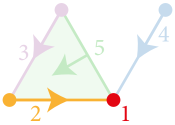
$K \simeq M$ *for some cell complex* M *built from the critical cells of* f.

Fundamental theorem of discrete Morse theory

Let f be a discrete Morse function on a cell complex K.

Theorem (Forman 1998)

If $(s, t]$ contains no critical value of f, then the sublevel set K_t collapses to K_s .



Corollary

$K \simeq M$ for some cell complex M built from the critical cells of f.

This homotopy equivalence is compatible with the filtration.

Corollary

K and M have isomorphic persistent homology (with regard to the sublevel sets of f).

Morse pairs and persistence pairs

Consider a *Morse filtration* (one or two simplices at a time).

Morse pair (σ, τ):

- inserting σ and τ simultaneously does not change the *homotopy type*

Morse pairs and persistence pairs

Consider a *Morse filtration* (one or two simplices at a time).
Morse pair (σ, τ):

- inserting σ and τ simultaneously does not change the *homotopy type*

Consider a *simplexwise filtration* (one simplex at a time).
Persistence pair (σ, τ):

- inserting simplex σ creates a new *homological* feature
- inserting τ destroys that feature again

Apparent pairs

Definition

In a simplexwise filtration, (σ, τ) is an *apparent* pair if

- σ is the youngest face of τ
- τ is the oldest coface of σ

Apparent pairs

Definition

In a simplexwise filtration, (σ, τ) is an *apparent* pair if

- σ is the youngest face of τ
- τ is the oldest coface of σ

Lemma

Any apparent pairs is a persistence pair.

Lemma

The apparent pairs form a discrete gradient.

- Generalizes a construction proposed by [Kahle 2011] for the study of random Rips filtrations

From Morse theory to persistence and back

Proposition (from Morse to persistence)

The pairs of a Morse filtration are apparent 0*-persistence pairs for the canonical simplexwise refinement of the filtration.*

From Morse theory to persistence and back

Proposition (from Morse to persistence)

The pairs of a Morse filtration are apparent 0-persistence pairs for the canonical simplexwise refinement of the filtration.

Proposition (from persistence to Morse)

Consider an arbitrary filtration with a simplexwise refinement.
The apparent 0-persistence pairs yield a Morse filtration

- *refining the original one, and*
- *refined by the simplexwise one.*

Emergent persistent pairs

Consider the *lexicographically refined Rips filtration:*

- increasing diameter, refined by
- lexicographic order

This is the simplexwise filtration for computations in Ripser.

Emergent persistent pairs

Consider the *lexicographically refined Rips filtration:*

- increasing diameter, refined by
- lexicographic order

This is the simplexwise filtration for computations in Ripser.

Lemma

Assume that

- τ *is the lexicographically minimal proper coface of* σ *with* $\operatorname{diam}(\tau) = \operatorname{diam}(\sigma)$*,*
- *and there is no persistence pair* (ρ, τ) *with* $\sigma \prec \rho$*.*

Then (σ, τ) *is an* emergent *persistence pair.*

Emergent persistent pairs

Consider the *lexicographically refined Rips filtration:*

- increasing diameter, refined by
- lexicographic order

This is the simplexwise filtration for computations in Ripser.

Lemma

Assume that

- τ *is the lexicographically minimal proper coface of* σ *with* $\operatorname{diam}(\tau) = \operatorname{diam}(\sigma)$,
- *and there is no persistence pair* (ρ, τ) *with* $\sigma \prec \rho$.

Then (σ, τ) *is an* emergent *persistence pair.*

- Includes all apparent pairs with persistence 0
- Can be identified *without* enumerating all cofaces of σ
  - Provides a shortcut for computation

Ripser Live: users from 156 different cities