Fantastic barcode algorithms and where to find them

Barbara Giunti

Topology of Data in Rome, 15-16 September 2022

TU Graz

SCIENCE
PASSION
TECHNOLOGY

Matrices and interval spheres

$$\begin{bmatrix} 1 & & 1 & 1 \\ & 1 & 1 & \\ 1 & & 1 & \\ & 1 & & 1 \\ 1 & & & \\ & & 1 & 1 \end{bmatrix}$$

Motivation

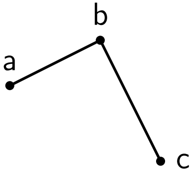
Motivation

http://www.zotero.org/groups/TDA-Applications

Boundary matrices

n-**simplex:** collection of $n + 1$ points

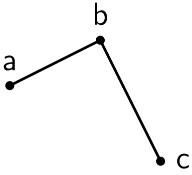
Boundary matrices



***n*-simplex:** collection of $n + 1$ points

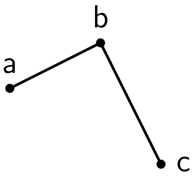
Simplicial complex: collection of simplices

Boundary matrices



n-simplex: collection of $n+1$ points
Simplicial complex: collection of simplices
Boundary of a n-simplex: all $(n-1)$-simplices without one of its points

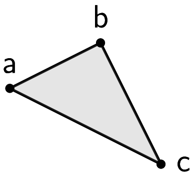
Boundary matrices



	ab	bc
a	1	
b	1	1
c		1

***n*-simplex:** collection of $n + 1$ points
Simplicial complex: collection of simplices
Boundary of a *n*-simplex: all $(n - 1)$-simplices without one of its points
(n)-boundary matrix: boundary of all the (n)-simplices

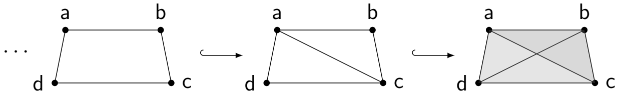
Boundary matrices



$$\begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cc} ab & bc \end{array} \\ \begin{bmatrix} 1 & \\ 1 & 1 \\ & 1 \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ ab \\ ac \\ bc \end{array} \begin{array}{c} abc \\ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \end{array}$$

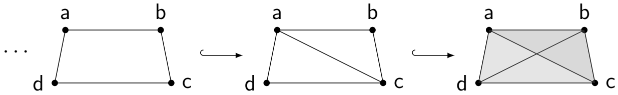
***n*-simplex:** collection of $n+1$ points
Simplicial complex: collection of simplices
Boundary of a *n*-simplex: all $(n-1)$-simplices without one of its points
(*n*)-boundary matrix: boundary of all the (n)-simplices

Filtrations



Filtration: nested sequence of simplicial complexes

Filtrations



	abc	acd	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Filtration: nested sequence of simplicial complexes

(Total) boundary matrix: boundary matrix of the final complex

Chain complexes and homology

Chain complex: sequence of (boundary) maps of vector spaces such that the composition of two consecutive maps is 0

Chain complexes and homology

Chain complex: sequence of (boundary) maps of vector spaces such that the composition of two consecutive maps is 0

degree n k

Chain complexes and homology

Chain complex: sequence of (boundary) maps of vector spaces such that the composition of two consecutive maps is 0

degree $n+1$ k

$$\downarrow$$

degree n k   k

Chain complexes and homology

Chain complex: sequence of (boundary) maps of vector spaces such that the composition of two consecutive maps is 0

$$\begin{array}{lcc} \text{degree } n+1 & & \mathsf{k} \\ & & \downarrow \\ \text{degree } n & \mathsf{k} & \mathsf{k} \end{array}$$

Homology: the quotient of the kernel of a map by the image of the previous one

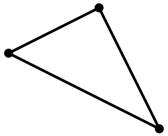
Chain complexes and homology

Chain complex: sequence of (boundary) maps of vector spaces such that the composition of two consecutive maps is 0

degree $n+1$ k

$\downarrow$

degree n k k



Homology: the quotient of the kernel of a map by the image of the previous one

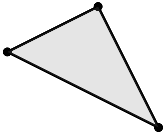
Chain complexes and homology

Chain complex: sequence of (boundary) maps of vector spaces such that the composition of two consecutive maps is 0

degree $n+1$ k

$\downarrow$

degree n k k



Homology: the quotient of the kernel of a map by the image of the previous one

Decomposition into interval spheres

Theorem (Chachólski, G., Landi, HHA 2020)

Every filtration decomposes uniquely into direct sum of interval spheres.

Decomposition into interval spheres

Theorem (Chachólski, G., Landi, HHA 2020)

Every filtration decomposes uniquely into direct sum of interval spheres.

Remark (Chachólski, G., Jin, Landi, *CGTA*, 2022)

The generators of the interval spheres give the birth and death times of the barcode. Any barcode algorithm is actually computing the interval sphere decomposition.

Decomposition into interval spheres

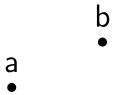
Theorem (Chachólski, G., Landi, HHA 2020)

Every filtration decomposes uniquely into direct sum of interval spheres.

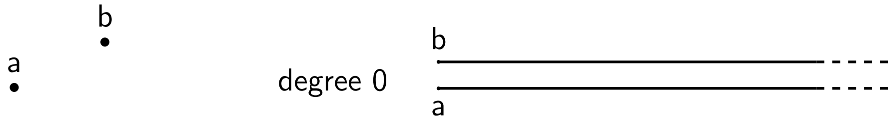
Remark (Chachólski, G., Jin, Landi, *CGTA*, 2022)

The generators of the interval spheres give the birth and death times of the barcode. Any barcode algorithm is actually computing the interval sphere decomposition.

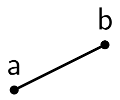
Interval spheres

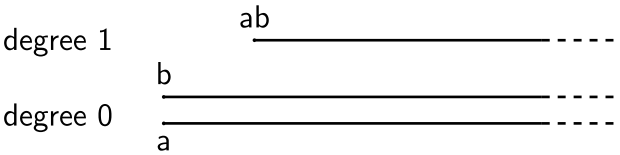


Interval spheres

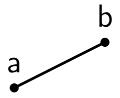


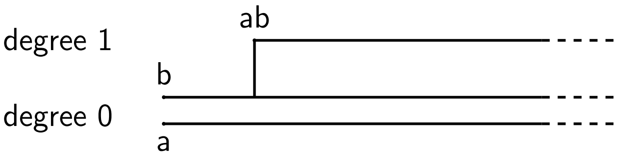
Interval spheres



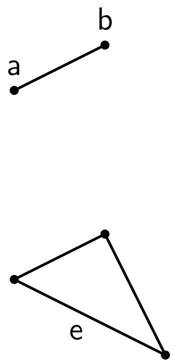


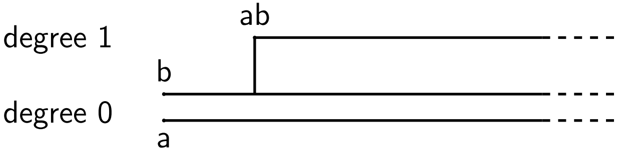
Interval spheres



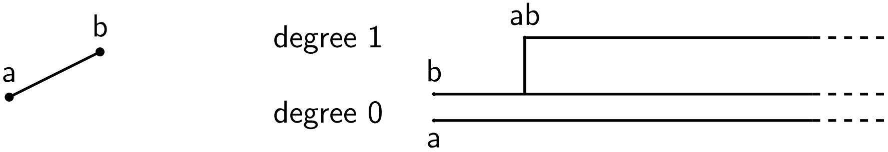


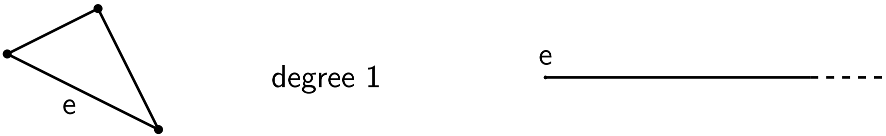
Interval spheres



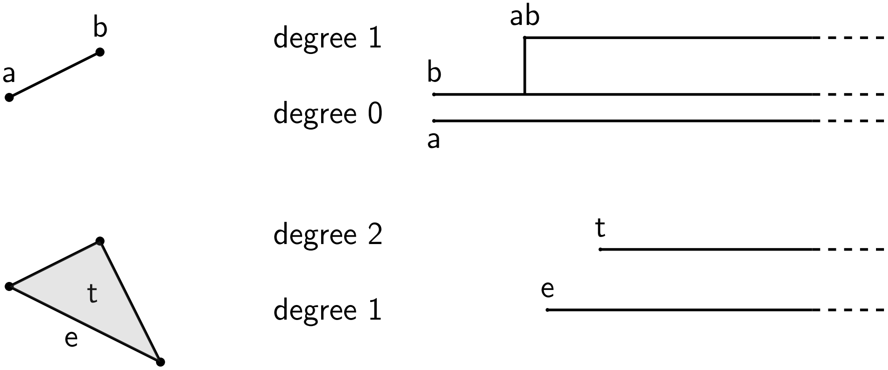


Interval spheres

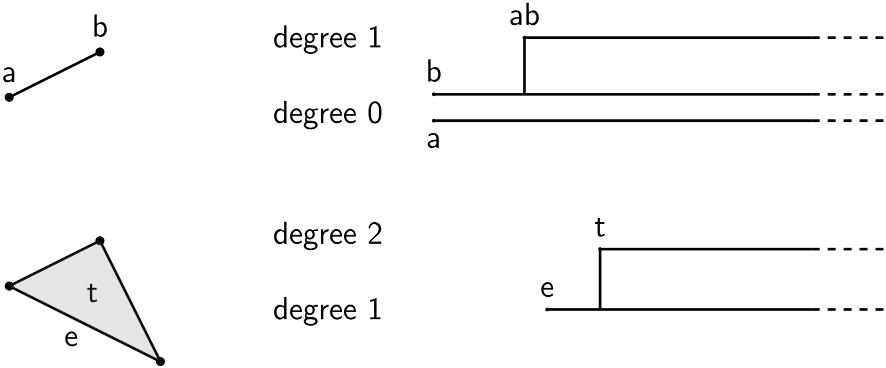




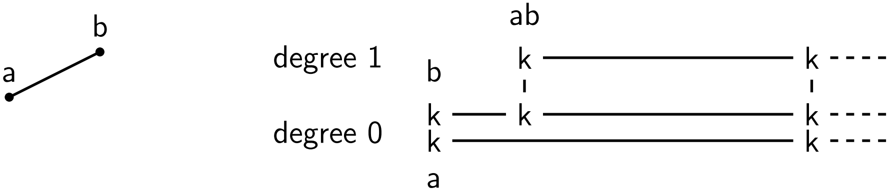
Interval spheres

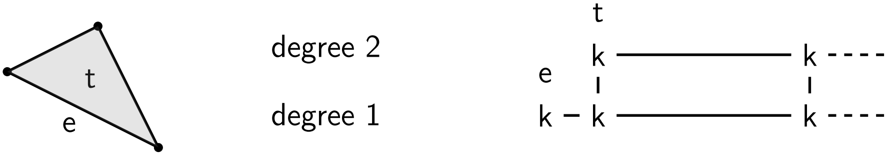


Interval spheres



Interval spheres





Pairing

	abc	**acd**	abd	bcd
bc				
ad				
ab				
cd				
ac				
bd				

Pairing

	abc	**acd**	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Pairing

	abc	**acd**	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Pairing

	abc	**acd**	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Pairing

	abc	**acd**	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Pairing

	abc	**acd**	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Pairing

	abc	**acd**	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Two elements are paired if and only if the sum of the differences of the ranks is 1

Topological persistence and simplification, Edelsbrunner, Letscher, Zomorodian, *FOCS*, 2000

Pairing

	abc	**acd**	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Two elements are paired if and only if the sum of the differences of the ranks is 1

$\Longrightarrow$ Any reduction that preserves these ranks is legit

Topological persistence and simplification, Edelsbrunner, Letscher, Zomorodian, *FOCS*, 2000

Standard barcode algorithm

Pivot $low(j)$: index of lowest nonzero element of column j

Input: Boundary matrix D
Output: Reduced matrix R

$R = D$
for $j = 1, \ldots, \#$ *of simplices* **do**
  while $low(j) = low(i) \neq 0$ *for* $i < j$ **do**
    add column i to column j

Topological persistence and simplification, Edelsbrunner, Letscher, Zomorodian, *FOCS*, 2000

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
 & 1 & & & & & & & \\
1 & & & & & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & 1 & & & &
\end{array}\right]
$$

Example

$$
\begin{bmatrix}
\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
 & 1 & & & & & & & \\
1 & & & & 1 & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & 1 & & & & \\
1 & & & & & & & &
\end{array}
\end{bmatrix}
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
 & 1 & & & 1 & & & & \\
1 & & & & 1 & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & 1 & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & 1 & & & & \\
 & & 1 & & 1 & & & & \\
 & 1 & & & 1 & & & & \\
1 & & & & 1 & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & 1 & 1 & 1 & 1 & 1 \\
 & & 1 & & 1 & 1 & 1 & 1 & 1 \\
 & 1 & & & 1 & 1 & 1 & 1 & 1 \\
1 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & & & & \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Standard barcode algorithm with clear

Input: Boundary matrix D
Output: Reduced matrix R

```
R = D
for j = 1, ..., # of simplices do
    while low(j) = low(i) ≠ 0 for i < j do
        add column i to column j
    if the column j is nonzero then
        Set column i to 0 for i = low(j)
```

Each interval sphere has two generators:

- If we find the “upper”, we can remove the “lower”
- If we find the “lower”, we can remove the “upper”

Persistent homology computation with a twist, Chen, Kerber, *EuroCG*, 2011

Standard barcode algorithm with clear

Input: Boundary matrix D

Output: Reduced matrix R

$R = D$

for $j = 1, \dots, \#$ *of simplices* **do**

  while $low(j) = low(i) \neq 0$ *for* $i < j$ **do**

    add column i to column j

  if *the column* j *is nonzero* **then**

    Set column i to 0 for $i = low(j)$

Each interval sphere has two generators:

- If we find the "upper", we can remove the "lower" CLEAR
- If we find the "lower", we can remove the "upper"

Persistent homology computation with a twist, Chen, Kerber, *EuroCG*, 2011

Standard barcode algorithm with clear

Input: Boundary matrix D
Output: Reduced matrix R

$R = D$
for $j = 1, \ldots, \#$ *of simplices* **do**
  while $low(j) = low(i) \neq 0$ *for* $i < j$ **do**
    add column i to column j
  if *the column* j *is nonzero* **then**
    Set column i to 0 for $i = low(j)$

Each interval sphere has two generators:

- If we find the "upper", we can remove the "lower" CLEAR
- If we find the "lower", we can remove the "upper"

Persistent homology computation with a twist, Chen, Kerber, *EuroCG*, 2011

Standard barcode algorithm with clear

Input: Boundary matrix D

Output: Reduced matrix R

$R = D$

for $j = 1, \ldots, \#$ *of simplices* **do**

  while $low(j) = low(i) \neq 0$ *for* $i < j$ **do**

    add column i to column j

  if *the column* j *is nonzero* **then**

    Set column i to 0 for $i = low(j)$

Each interval sphere has two generators:

- If we find the "upper", we can remove the "lower" CLEAR
- If we find the "lower", we can remove the "upper" COMPRESS

Persistent homology computation with a twist, Chen, Kerber, *EuroCG*, 2011

Cohomology

The coboundary of a $(n-1)$-simplex s is the collection of all n-simplices that have s in their boundary.

Cohomology

The coboundary of a $(n-1)$-simplex s is the collection of all n-simplices that have s in their boundary.

The anti-transpose of a boundary matrix is (almost) the coboundary matrix of the filtration and its pairing is in bijection with the pairing of the boundary matrix[1].

[1] Dualities in persistent (co)homology, de Silva, Morozov, Vejdemo-Johansson, *Inverse Problems*, 2011

Cohomology

The coboundary of a $(n-1)$-simplex s is the collection of all n-simplices that have s in their boundary.

The anti-transpose of a boundary matrix is (almost) the coboundary matrix of the filtration and its pairing is in bijection with the pairing of the boundary matrix[1].

On some inputs, the standard barcode algorithm with clear is much more efficient on the coboundary than on the boundary matrix.

[1] Dualities in persistent (co)homology, de Silva, Morozov, Vejdemo-Johansson, *Inverse Problems*, 2011

Cohomology

The coboundary of a $(n-1)$-simplex s is the collection of all n-simplices that have s in their boundary.

The anti-transpose of a boundary matrix is (almost) the coboundary matrix of the filtration and its pairing is in bijection with the pairing of the boundary matrix[1].

On some inputs, the standard barcode algorithm with clear is much more efficient on the coboundary than on the boundary matrix. Nothing comparable happens for the compress optimisation.

[1]Dualities in persistent (co)homology, de Silva, Morozov, Vejdemo-Johansson, *Inverse Problems*, 2011

Row barcode algorithm

Pivot *left* (i): index of leftmost nonzero element of row i

```
Input: Boundary matrix D
Output: Reduced matrix R
R = D
for i = # of simplices, ..., 1 do
    while left(i) = left(j) ≠ 0 for j > i do
        add row j to row i
```

Tripartitions and bases of an ordered complex, Edelsbrunner, Ölsböck, *Discrete & Computational Geometry*, 2020

Row barcode algorithm with compress

Pivot $\mathit{left}(i)$: index of leftmost nonzero element of row i

Input: Boundary matrix D
Output: Reduced matrix R

$R = D$
for $i = \#$ *of simplices*$, \ldots, 1$ **do**
  while $\mathit{left}(i) = \mathit{left}(j) \neq 0$ *for* $j > i$ **do**
    add row j to row i
  if *the row* i *is nonzero* **then**
    Set row j to 0 for $j = \mathit{left}(i)$

Tripartitions and bases of an ordered complex, Edelsbrunner, Ölsböck, *Discrete & Computational Geometry*, 2020 / Notes on pivot pairings, G., *EuroCG*, 2021

Row barcode algorithm with compress

Pivot $\mathit{left}\,(i)$: index of leftmost nonzero element of row i

Input: Boundary matrix D
Output: Reduced matrix R

$R = D$
for $i = \#$ *of simplices*$, \ldots, 1$ **do**
    while $\mathit{left}\,(i) = \mathit{left}\,(j) \neq 0$ *for* $j > i$ **do**
        add row j to row i
    if *the row* i *is nonzero* **then**
        Set row j to 0 for $j = \mathit{left}\,(i)$

If the filtration is full, then # of column operations with clear on the coboundary matrix is equal to # of row operations with the compress on the boundary matrix.

Implementations

- JavaPlex
- Dionysus
- Gudhi
- Phat
- Ripser
- Giotto-PH

A roadmap for the computation of persistent homology, Otter, Porter, Tillmann, Grindrod, Harrington, *EPJ Data Science*, 2017

Memory and data type

The choice of how to store the boundary matrix has big impact on the performances, as boundary matrices are initially sparse[1].

[1] PHAT-persistent homology algorithms toolbox, Bauer, Kerber, Reininghaus, Wagner, *Journal of symbolic computation*, 2017

Memory and data type

The choice of how to store the boundary matrix has big impact on the performances, as boundary matrices are initially sparse[1].
$\Longrightarrow$ use data types whose size is proportional to the number of nonzero entries in a column

[1] PHAT-persistent homology algorithms toolbox, Bauer, Kerber, Reininghaus, Wagner, *Journal of symbolic computation*, 2017

Memory and data type

The choice of how to store the boundary matrix has big impact on the performances, as boundary matrices are initially sparse[1].
$\Longrightarrow$ use data types whose size is proportional to the number of nonzero entries in a column

- The cost of adding column i to column j is $\#i$;
- The **cost of a matrix reduction** is the added cost of all column additions;
- The **fill-up** of is the number of entries in the reduced matrix.

[1] PHAT-persistent homology algorithms toolbox, Bauer, Kerber, Reininghaus, Wagner, *Journal of symbolic computation*, 2017

Memory and data type

The choice of how to store the boundary matrix has big impact on the performances, as boundary matrices are initially sparse[1].
$\Longrightarrow$ use data types whose size is proportional to the number of nonzero entries in a column

- The cost of adding column i to column j is $\#i$;
- The **cost of a matrix reduction** is the added cost of all column additions;
- The **fill-up** of is the number of entries in the reduced matrix.

[1] PHAT-persistent homology algorithms toolbox, Bauer, Kerber, Reininghaus, Wagner, *Journal of symbolic computation*, 2017

Memory and data type

The choice of how to store the boundary matrix has big impact on the performances, as boundary matrices are initially sparse[1].
$\implies$ use data types whose size is proportional to the number of nonzero entries in a column

- The cost of adding column i to column j is $\# i$;
- The **cost of a matrix reduction** is the added cost of all column additions;
- The **fill-up** of is the number of entries in the reduced matrix.

[1] PHAT-persistent homology algorithms toolbox, Bauer, Kerber, Reininghaus, Wagner, *Journal of symbolic computation*, 2017

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & 1 & 1 & 1 & 1 & 1 \\
 & & 1 & & 1 & 1 & 1 & 1 & 1 \\
 & 1 & & & 1 & 1 & 1 & 1 & 1 \\
1 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & & & & \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & 1 & 1 & 1 & 1 & 1 \\
 & & 1 & & 1 & 1 & 1 & 1 & 1 \\
 & 1 & & & 1 & 1 & 1 & 1 & 1 \\
1 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & & & & \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

What if we try to keep the matrix sparse during the reduction?

Swap barcode algorithm

Input: Boundary matrix D

Output: Reduced matrix R

$R = D$

for $j = 1, \ldots, \#$ *of simplices* **do**

  while $low(j) = low(i) \neq 0$ *for* $i < j$ **do**

    if $\#j < \#i$ **then**

      swap j and i

    add column i to j

  if *column* j *is nonzero* **then**

    Set i to 0 for $i = low(j)$

Keeping it sparse: Computing Persistent Homology revised, Bauer, Bin Masood, G., Houry, Kerber, Rathod, *to appear*

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
 & 1 & & & & & & & \\
1 & & & & & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & 1 & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & 1 & & & & \\
 & & 1 & & 1 & & & & \\
 & 1 & & & 1 & & & & \\
1 & & & & 1 & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & 1 & & & \\
 & & 1 & & & 1 & & & \\
 & 1 & & & & 1 & & & \\
1 & & & & & 1 & & & \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & & & & \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
\qquad
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & 1 & 1 & 1 & 1 & 1 \\
 & & 1 & & 1 & 1 & 1 & 1 & 1 \\
 & 1 & & & 1 & 1 & 1 & 1 & 1 \\
1 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & & & & \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Retrospective barcode algorithm

Input: Boundary matrix D
Output: Reduced boundary matrix R

$R = D$
for $j = 1, \ldots, \#$ *of simplices* **do**
    Remove the negative entries from j
    while *column* j *is nonzero and* $R_j^{low(i)} \neq 0$ *for* $i < j$ **do**
        add column i to column j
    for *every column* $i < j$ *with* $R_i^{low(j)} \neq 0$ **do**
        add column j to column i

Keeping it sparse: Computing Persistent Homology revised, Bauer, Bin Masood, G., Houry, Kerber, Rathod, *to appear*

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
 & 1 & & & & & & & \\
1 & & & & & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & & & & & \\
1 & & & & 1 & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
 & 1 & & & & & & & \\
1 & & & & 1 & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
 & 1 & 1 & & & & & & \\
1 & 1 & & & 1 & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
1 & 1 & & & 1 & & & & \\
1 & & & & 1 & & & & \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & 1 & 1 & & & & & \\
1 & 1 & 1 & & 1 & & & & \\
 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Example

$$
\left[\begin{array}{cccc|c|cccc}
* & * & * & * & * & * & * & * & * \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & & & 1 & 1 & 1 & 1 & 1 \\
1 & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
 & & & & & 1 & & & \\
 & & & & & & 1 & & \\
 & & & & & & & 1 & \\
 & & & & & & & & 1 \\
\hline
 & & & & 1 & & & & \\
\hline
 & & & 1 & & & & & \\
 & & 1 & & & & & & \\
 & 1 & & & & & & & \\
1 & & & & & & & &
\end{array}\right]
$$

Experiments

	Alpha shape			Lower star			Vietoris–Rips		
Algorithm	Fill-up	Col.ops	Bitflips	Fill-up	Col.ops	Bitflips	Fill-up	Col.ops	Bitflips
clear	6.35M	2.17M	62.30M	34.70M	4.90M	29.43M	14,975	5.35M	19.02M
clear*	6.90M	1.41M	84.09M	33.18M	4.91M	30.06M	0.51M	222	38,342
swap	1.56M	1.10M	7.37M	33.61M	4.83M	26.79M	14,887	5.19M	17.29M
swap*	2.04M	1.54M	20.72M	31.59M	4.86M	27.15M	0.51M	224	33,830
retro	1.14M	2.34M	19.93M	8.13M	21.51M	34.70M	5,049	0.48M	0.49M
retro*	7.94M	3.51M	40.87M	7.35M	21.94M	31.73M	6.41M	9,944	14.32M
mix	1.04M	15.04M	140.34M	10.85M	120.02M	239.17M	5,172	0.52M	0.57M
mix*	1.41M	14.03M	71.37M	10.79M	76.24M	150.20M	0.49M	0.22M	21.84M

Table: "M" stands for millions. The $*$ means via row barcode algorithm.

Experiments

	Alpha shape			Lower star			Vietoris–Rips			Shuffled		
Algorithm	40K	80K	160K	Tooth	Lobster	Skull	104	297	445	50	75	100
clear	*6.5	*18.6	*49.8	2.0	25.8	23.9	*0.0	*0.1	*0.1	*0.1	*1.3	*11.2
swap	*9.8	*28.2	*74.2	2.4	38.6	25.4	*0.0	*0.1	*0.2	0.1	0.6	2.9
retro	4.3	11.8	30.9	*4.9	*29.0	*61.6	0.1	1.4	7.2	0.0	0.1	0.3

Table: Best running times (in seconds) on various data sets. The $*$ means via row barcode algorithm.

Experiments

	List	Vector	Set	Heap	P-Heap	P-Set	P-Full	P-Bit-Tree
clear	58.3	1.9	7.9	7.1	6.5	7.5	2.2	**0.9**
clear*	144.6	2.8	11.9	9.5	8.9	9.8	3.2	0.9
swap	45.9	1.2	1.1	68.8	63.7	1.1	**0.5**	27.7
swap*	+5m	3.0	4.0	275.6	213.6	4.1	1.6	122.8
retro	2.8	**0.6**	2.9	6.5	6.3	9.4	4.6	3.3
retro*	72.7	2.6	20.3	128.0	167.5	182.3	103.8	54.6
mix	40.1	**4.0**	17.7	44.5	38.0	14.5	6.5	22.1
mix*	+5m	14.0	15.5	+5m	+5m	12.5	6.5	268.2

Table: Alpha filtration on 10000 points on a torus. All timings are in seconds but for the timeout (minutes). The $*$ means via row barcode algorithm.

Step columns and critical pivots

A **clique filtration** is a filtration where the n-simplices are added as soon as their boundary $(n-1)$-simplices are added.

Step columns and critical pivots

A **clique filtration** is a filtration where the n-simplices are added as soon as their boundary $(n-1)$-simplices are added.

A **step column** is a column that is not modified during the reduction.

	abc	acd	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Step columns and critical pivots

A **clique filtration** is a filtration where the n-simplices are added as soon as their boundary $(n-1)$-simplices are added.

A **step column** is a column that is not modified during the reduction.
A **critical pivot** is a pivot that is in the reduced matrix but was not in the initial one.

	abc	acd	abd	bcd
bc	1			1
ad		1	1	
ab	1		1	
cd		1		1
ac	1	1		
bd			1	1

Step columns and critical pivots

A **clique filtration** is a filtration where the n-simplices are added as soon as their boundary $(n-1)$-simplices are added.

A **step column** is a column that is not modified during the reduction.
A **critical pivot** is a pivot that is in the reduced matrix but was not in the initial one.

	abc	acd	abd	bcd
bc	1	1		
ad		1	1	
ab	1	1	1	
cd		1		
ac	1			
bd			1	

Step columns and critical pivots

A **clique filtration** is a filtration where the n-simplices are added as soon as their boundary $(n-1)$-simplices are added.

A **step column** is a column that is not modified during the reduction.
A **critical pivot** is a pivot that is in the reduced matrix but was not in the initial one.

	abc	acd	abd	bcd
bc	1	1		
ad		1	1	
ab	1	1	1	
cd		1		
ac	1			
bd			1	

Lemma (G., Houry, Kerber, *ISSAC*, 2022)

In a clique filtration, there is an critical pivot if and only if at the corresponding simplex is a "lower" generator of a new interval sphere.

Average complexity

Lemma (G., Houry, Kerber, *ISSAC*, 2022)

The cost of a matrix reduction is bounded by $\#$ of columns $\times$ the fill-up of the reduced matrix.

Average complexity

Lemma (G., Houry, Kerber, *ISSAC*, 2022)

The cost of a matrix reduction is bounded by $\#$ of columns $\times$ the fill-up of the reduced matrix.

The fill-up is bounded by $\Theta(\#\text{ of rows}) + \sum_{i=1}^{\#\text{of rows}} \mathbb{P}(\text{new ``lower'' generator})$.

Average complexity

Lemma (G., Houry, Kerber, *ISSAC*, 2022)

The cost of a matrix reduction is bounded by $\#$ of columns $\times$ the fill-up of the reduced matrix.

The fill-up is bounded by $\Theta(\#\text{ of rows}) + \sum_{i=1}^{\#\text{of rows}} \mathbb{P}(\text{new "lower" generator})$.

For every random clique filtration model for which we can bound the probability of obtaining a new interval sphere, we have a bound on the average complexity of the barcode algorithm.

Average complexity

Theorem (G., Houry, Kerber, *ISSAC*, 2022)

Let R be the reduced 1-boundary matrix of a Vietoris–Rips filtration. Then $\mathbb{E}[\textit{fill-up of } R] = O(n^2 \log^2 n)$ and $\mathbb{E}[\textit{cost of matrix reduction}] = O(n^5 \log^2 n)$.

Average complexity

Theorem (G., Houry, Kerber, *ISSAC*, 2022)

Let R be the reduced 1-boundary matrix of a Vietoris–Rips filtration. Then $\mathbb{E}[\text{fill-up of } R] = O(n^2 \log^2 n)$ and $\mathbb{E}[\text{cost of matrix reduction}] = O(n^5 \log^2 n)$.

Let R be the reduced 1-boundary matrix of an Erdős–Rényi filtration. Then $\mathbb{E}[\text{fill-up of } R] = O(n^3 \log n)$ and $\mathbb{E}[\text{cost of matrix reduction}] = O(n^6 \log n)$.

Average complexity

Theorem (G., Houry, Kerber, *ISSAC*, 2022)

Let R be the reduced 1-boundary matrix of a Vietoris–Rips filtration. Then $\mathbb{E}[\text{fill-up of } R] = O(n^2 \log^2 n)$ and $\mathbb{E}[\text{cost of matrix reduction}] = O(n^5 \log^2 n)$. Let R be the reduced 1-boundary matrix of an Erdős–Rényi filtration. Then $\mathbb{E}[\text{fill-up of } R] = O(n^3 \log n)$ and $\mathbb{E}[\text{cost of matrix reduction}] = O(n^6 \log n)$.

The worst-case complexities are, respectively, $O(n^4)$ and $O(n^7)$, and they are realized for the Erdős–Rényi filtration.

Take home

- The pairs links the matrices (combinatorial) to the barcode (homological/geometrical)
- Any reduction that maintains the pairs is valid
- Things can go (sort of) bad but they usually don't

Take home

- The pairs links the matrices (combinatorial) to the barcode (homological/geometrical)
- Any reduction that maintains the pairs is valid
- Things can go (sort of) bad but they usually don't

Thank you for your attention!