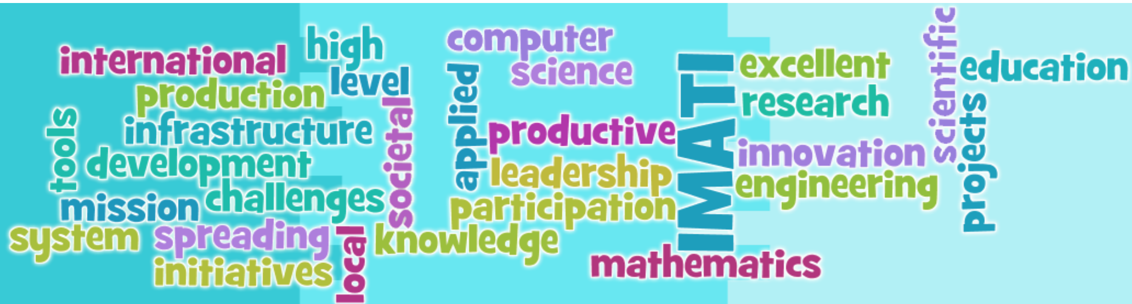
Istituto di Matematica Applicata e Tecnologie Informatiche «Enrico Magenes»

Extracting Geometrical Features From Data

Topological Data Analysis

Ulderico Fugacci

CNR - IMATI



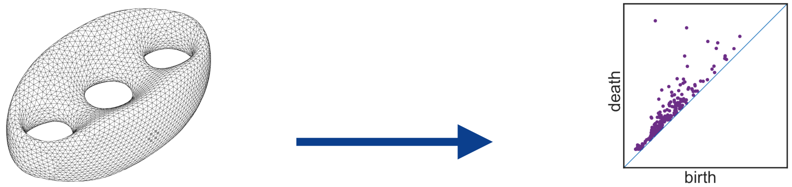
Topological Data Analysis

Outline:

Computing Persistence

Persistent Homology Computation

Topological Data Analysis allows for assigning to (almost) ***any dataset*** a collection of features representing a ***topological summary*** of the input data



Shape

Features

Goal:

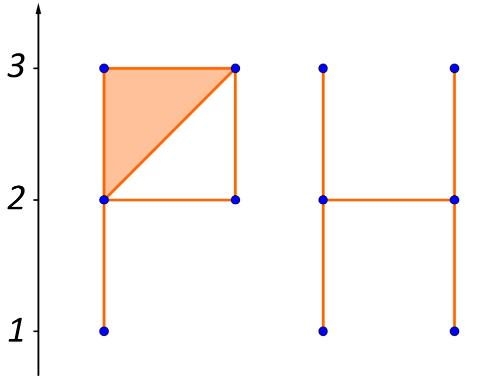
- ***How to efficiently compute (persistent) homology?***
- ***How to compactly encode simplicial complexes of high dimension and large size?***

Persistent Homology Computation

Standard Algorithm:

[Zomorodian & Carlsson 2005]

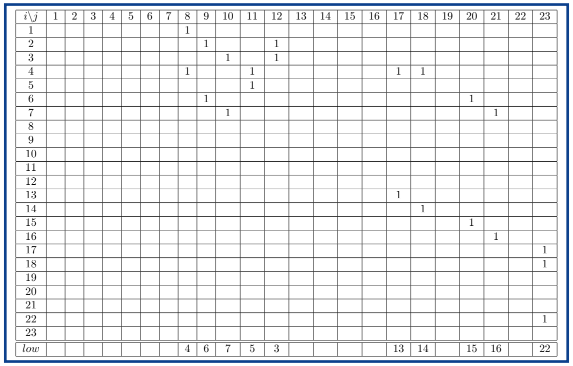
From:



To:

H_0 **[1, 2]**
[1, ∞)
[1, ∞)

H_1 **[3, ∞)**



$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1						
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16		22

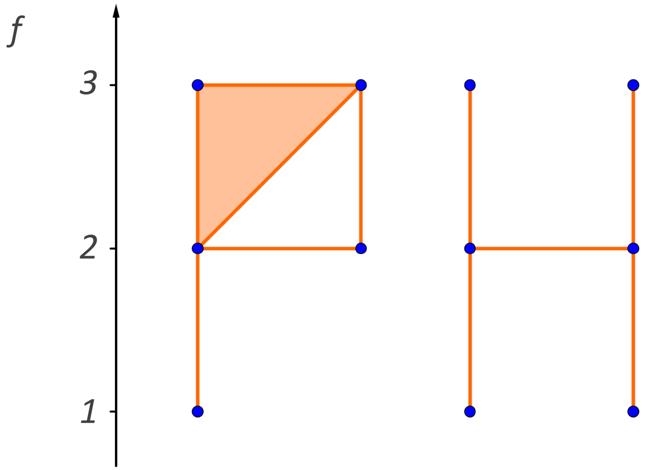
Compute a ***reduced boundary matrix*** for $\{ K^p \}_p$ from which easily read the persistence pairs

Persistent Homology Computation

Given a filtered simplicial complex, let us consider its *filtering function* ***f*:**

$$f(\sigma) := \min \{ p \mid \sigma \in K^p \}$$

Conversely, $K^p := \{ \sigma \in K \mid f(\sigma) \leq p \}$



Total Ordering on $\{ K^p \}_p$:

A sequence $\sigma_1, \sigma_2, \ldots, \sigma_n$ of the simplices of K such that:

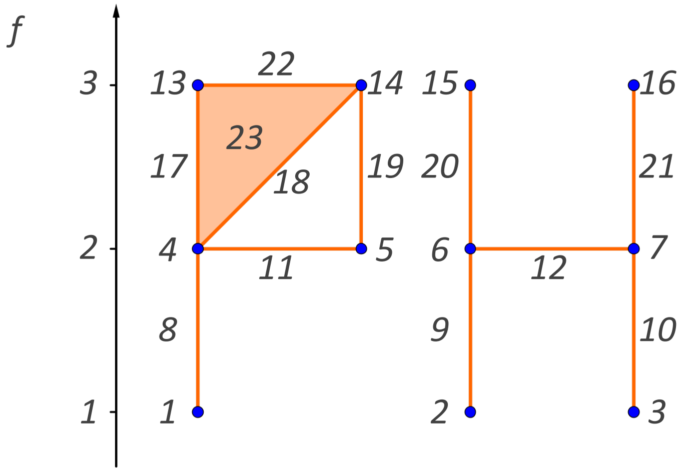
- *if $f(\sigma_i) < f(\sigma_j)$, then $i < j$*
- *if σ_i is a proper face of σ_j, then $i < j$*

Persistent Homology Computation

Given a filtered simplicial complex, let us consider its *filtering function* ***f*:**

$$f(\sigma) := \min \{ p \mid \sigma \in K^p \}$$

Conversely, $K^p := \{ \sigma \in K \mid f(\sigma) \leq p \}$



A Possible Choice:

Set $\sigma < \sigma'$ *if:*

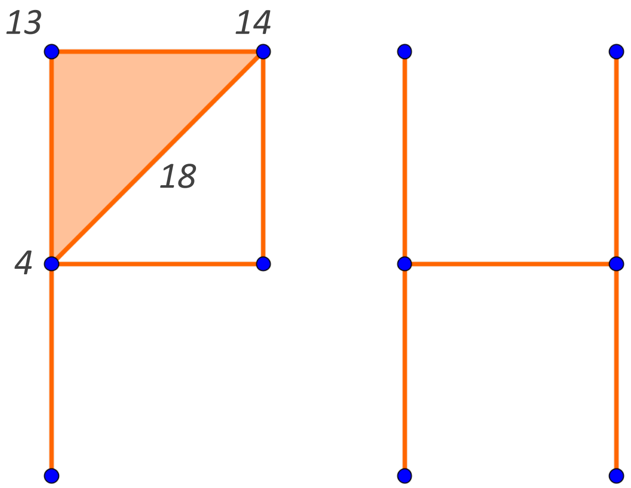
- $f(\sigma) < f(\sigma')$
- $f(\sigma) = f(\sigma')$ *and* $\dim(\sigma) < \dim(\sigma')$
- $f(\sigma) = f(\sigma')$, $\dim(\sigma) = \dim(\sigma')$, *and σ precedes σ' w.r.t. the* ***lexicographic order*** *of their vertices*

Persistent Homology Computation

Boundary Matrix:

A square matrix ***D*** of size *n* x *n* defined by

$$D_{i,j} := \begin{cases} 1 & \text{if } \sigma_i \text{ is a face of } \sigma_j \text{ s.t. } \dim(\sigma_i) = \dim(\sigma_j) - 1 \\ 0 & \text{otherwise} \end{cases}$$



E.g.

- $D_{4,18} = 1$
- $D_{14,18} = 1$
- $D_{13,18} = 0$

Persistent Homology Computation

Reduced Matrix:

Given a non-null column *j* of a boundary matrix *D*,

$$low(j) := \max \{ i \mid D_{i,j} \neq 0 \}$$

A matrix **R** *is called* ***reduced*** *if, for each pair of non-null columns j_1, j_2,*

$$low(j_1) \neq low(j_2)$$

Equivalently, if low function is ***injective*** on its domain of definition

Persistent Homology Computation

$i \backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1														
3										1													
4								1			1						1	1					
5											1								1				
6									1			1								1			
7										1		1									1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	7					13	14	14	15	16	14	22

low(10) = 7 = low(12) → *D* is **not** reduced

Persistent Homology Computation

Reduction Algorithm:

```
Matrix $R = D$
for $j = 1, ..., n$ do
  while $\exists$ j’’ < j with low(j’) = low(j) do
    R.column(j) = R.column(j) + R.column(j’)
  endwhile
endfor
return $R$
```

Time Complexity:

At most n^2 column additions → *$O(n^3)$ in the worst case*

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1														
3										1													
4								1			1						1	1					
5											1								1				
6									1			1								1			
7										1		1									1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	7					13	14	14	15	16	14	22

Initialize ***R*** to ***D*,** where

D is the ***boundary matrix*** of *K*

expressed according with a ***total ordering*** of its simplices

j < 12

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1														
3										1													
4								1			1						1	1					
5											1								1				
6									1			1								1			
7										1		1									1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	7					13	14	14	15	16	14	22

For each ***j* < *12***,

there is **no** ***j’* < *j*** such that

low(j’) = low(j)

So, **increase *j* by *1***

j’ *j*

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1														
3										1													
4								1			1						1	1					
5											1								1				
6									1			1								1			
7										1		1									1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	7					13	14	14	15	16	14	22

For ***j = 12*, *low(12) = 7***

column j’=10 is such that ***low(j’) = low(j) = 7***

So, set

column 12 := ***column 12*** + ***column 10***

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1														
3										1		1											
4								1			1						1	1					
5											1								1				
6									1			1								1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	6					13	14	14	15	16	14	22

For ***j = 12***, ***low(12) = 7***

column j'=10 is such that ***low(j') = low(j) = 7***

So, set

column 12 := ***column 12*** + ***column 10*** ⟶ ***low(12) = 6***

									j’			j											
$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1														
3										1		1											
4								1			1						1	1					
5											1								1				
6									1			1								1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	6					13	14	14	15	16	14	22

For ***j* = 12**, ***low*(12) = 6**

***column j’* = 9** is such that ***low*(*j’*) = *low*(*j*) = 6**

So, set

***column* 12 := *column* 12 + *column* 9**

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1								1				
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14	14	15	16	14	22

For ***j* = 12**, ***low*(12) = 6**

column j’* = 9** is such that ***low(j’) = low(j) = 6

So, set

column 12** := **column 12** + **column 9 $\longrightarrow$ ***low*(12) = 3**

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1								1				
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14	14	15	16	14	22

For each ***j = 12***,

there is **no *j’ < j*** such that

low(j’) = low(j) = 3

So, **increase *j* by *1***

12 < j < 19

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1								1				
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14	14	15	16	14	22

For each ***12 < j < 19***,

there is **no *j' < j*** such that

low(j') = low(j)

So, **increase *j* by *1***

j' *j*

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1								1				
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1	1			1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14	14	15	16	14	22

For ***j* = 19, *low*(19) = 14**

***column j'*= 18** is such that ***low(j')* = *low(j)* = 14**

So, set

column 19* := *column 19* + *column 18

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1	1				
5											1								1				
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1				1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14	5	15	16	14	22

For ***j* = 19, *low*(19) = 14**

column j'= 18 is such that ***low(j') = low(j) = 14***

So, set

column 19* := *column 19* + *column 18 ⟶ ***low(19) = 5***

										j'								j				
$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1	1				
5											1								1				
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1				1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14	5	15	16	14	22

For ***j = 19, low(19) = 5***

column j'= 11 is such that ***low(j') = low(j) = 5***

So, set

column 19 **:=** ***column 19*** **+** ***column 11***

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1				1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16	14	22

For ***j = 19, low(19) = 5***

column j'= 11 is such that ***low(j') = low(j) = 5***

So, set

column 19 := ***column 19*** + ***column 11*** ⟶ ***low(19) undefined***

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1				1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							

low								4	6	7	5	3					13	14		15	16	14	22

For each ***j* = *19***,

there is **no *j' < j*** such that

low(j') = low(j)

So, **increase *j* by *1***

19 < j < 22

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1				1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16	14	22

For each ***19 < j < 22***,

there is **no *j' < j*** such that

low(j') = low(j)

So, **increase *j* by *1***

																	j’				j	
$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1				1	
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16	14	22

For ***j = 22***, ***low(22) = 14***

column j’= 18 is such that ***low(j’) = low(j) = 14***

So, set

column 22 := ***column 22*** + ***column 18***

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1				1	
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16	13	22

For ***j* = 22**, ***low*(22) = 14**

column j'= 18 is such that ***low(j') = low(j) = 14***

So, set

column 22 := *column 22* + *column 18* → ***low(22) = 13***

																	j’					*j*	
$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1				1	
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1					1	
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16	13	22

For ***j* = 22**, ***low*(22) = 13**

column j’= 17 is such that ***low(j’) = low(j) = 13***

So, set

column 22 := ***column 22*** + ***column 17***

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1						
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16		22

For ***j* = 22**, ***low(22) = 13***

column j'= 17 is such that ***low(j') = low(j) = 13***

So, set

column 22 := *column 22* + *column 17* $\longrightarrow$ ***low(22) undefined***

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1						
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							

low								4	6	7	5	3					13	14		15	16		22

For each ***j = 22***,

there is **no *j’ < j*** such that

low(j’) = low(j)

So, **increase *j* by *1***

j

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1						
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16		22

For each ***j* = 23**,

there is **no *j'* < *j*** such that

low(j') = low(j) = 22

So, **matrix *R* is reduced**

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1						
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16		22

The algorithm returns the above **reduced matrix *R***

Persistent Homology Computation

Retrieving Persistence Pairs:

- For each $i = 1, \dots, n$,

  if there exists j such that $low(j) = i$ ⇨ **$[i, j]$ is a pair for R**

- Once every i has been parsed,

  if i is an **unpaired** value ⇨ **$[i, \infty)$ is a pair for R**

From pairs of R to the **"actual" persistence pairs of $\{K^p\}_p$**:

$[i, j]$ corresponds to $[f(\sigma_i), f(\sigma_j)]$

$[i, \infty)$ corresponds to $[f(\sigma_i), \infty)$

(homological degree = $\dim(\sigma_i)$)

Persistent Homology Computation

H_0

[1, ∞)

[2, ∞)

[3, 12]

[4, 8]

[5, 11]

[6, 9]

[7, 10]

[13, 17]

[14, 18]

[15, 20]

[16, 21]

H_1

[19, ∞)

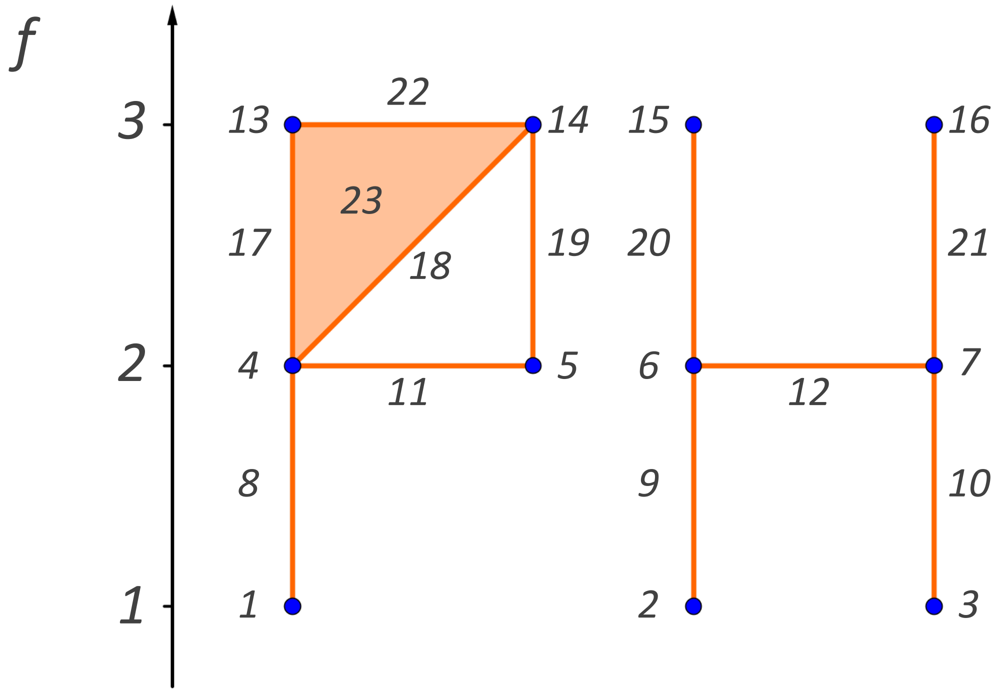
[22, 23]

$i\backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1								1															
2									1			1											
3										1		1											
4								1			1						1	1					
5											1												
6									1											1			
7										1											1		
8																							
9																							
10																							
11																							
12																							
13																	1						
14																		1					
15																				1			
16																					1		
17																							1
18																							1
19																							
20																							
21																							
22																							1
23																							
low								4	6	7	5	3					13	14		15	16		22

Persistent Homology Computation

H_0

$[1, \infty)$	**$[1, \infty)$**
$[2, \infty)$	**$[1, \infty)$**
$[3, 12]$	**$[1, 2]$**
$[4, 8]$	$[2, 2]$
$[5, 11]$	$[2, 2]$
$[6, 9]$	$[2, 2]$
$[7, 10]$	$[2, 2]$
$[13, 17]$	$[3, 3]$
$[14, 18]$	$[3, 3]$
$[15, 20]$	$[3, 3]$
$[16, 21]$	$[3, 3]$



H_1

$[19, \infty)$	**$[3, \infty)$**
$[22, 23]$	$[3, 3]$

Persistent Homology Computation

Standard algorithm to compute (persistent) homology [Zomorodian & Carlsson 2005]:

- Based on a ***matrix reduction***
- ***Linear complexity*** in practical cases
- ***Cubic complexity*** in the worst case

Several different strategies:

Direct approaches:

- ***Zigzag persistent homology*** [Milosavljević et al. ’05]
- ***Computation with a twist*** [Chen, Kerber ’11]
- ***Dual algorithm*** [De Silvia et al. ’11]
- ***Output-sensitive algorithm*** [Chen, Kerber ’13]
- ***Multi-field algorithm*** [Boissonnat, Maria ’14]
- ***Annotation-based methods*** [Boissonnat et al. ’13; Dey et al. ’14]

Distributed approaches:

- ***Spectral sequences*** [Edelsbrunner, Harer ’08; Lipsky et al. ’11]
- ***Constructive Mayer-Vietoris*** [Boltcheva et al. ’11]
- ***Multicore coreductions*** [Murty et al. ’13]
- ***Multicore homology*** [Lewis, Zomorodian ’14]
- ***Persistent homology in chunks*** [Bauer et al. ‘14a]
- ***Distributed persistent computation*** [Bauer et al. ‘14b]

Coarsening approaches:

- ***Topological operators and simplifications*** [Mrozek, Wanner ’10; Dłotko, Wagner ’14]
- ***Morse-based approaches*** [Robins et al. ’11; Harker et al. ’14; Fugacci et al. ’14]

Persistent Homology Computation

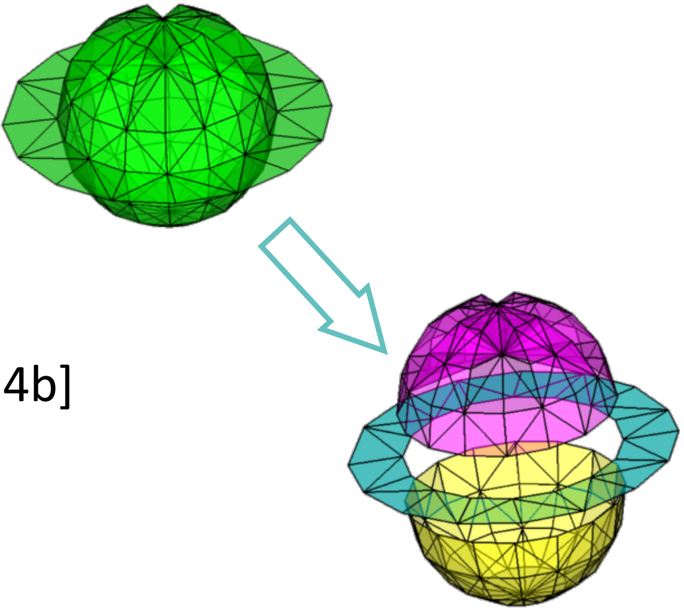
Direct Approaches:

- ***Zigzag persistent homology*** [Milosavljević et al. ’05]
- ***Computation with a twist*** [Chen, Kerber ’11]
- ***Dual algorithm*** [De Silvia et al. ’11]
- ***Output-sensitive algorithm*** [Chen, Kerber ’13]
- ***Multi-field algorithm*** [Boissonnat, Maria ’14]
- ***Annotation-based methods*** [Boissonnat et al. ’13; Dey et al. ’14]

Persistent Homology Computation

Distributed Approaches:

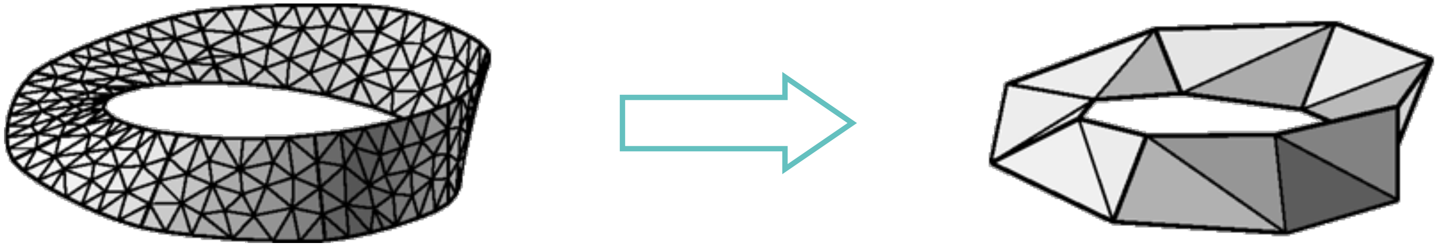
- ***Spectral sequences*** [Edelsbrunner, Harer ’08; Lipsky et al. ’11]
- ***Constructive Mayer-Vietoris*** [Boltcheva et al. ’11]
- ***Multicore coreductions*** [Murty et al. ’13]
- ***Multicore homology*** [Lewis, Zomorodian ’14]
- ***Persistent homology in chunks*** [Bauer et al. ‘14a]
- ***Distributed persistent computation*** [Bauer et al. ‘14b]



Persistent Homology Computation

Coarsening Approaches:

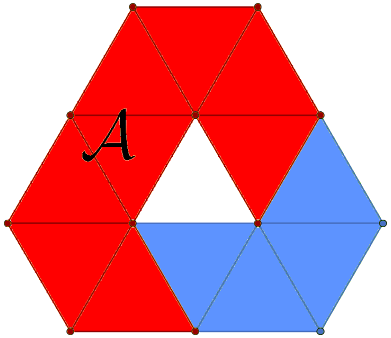
- ***Topological operators and simplifications*** [Dłotko, Wagner ’14]
  - Acyclic subcomplexes [Mrozek et al. ‘08]
  - Reductions and coreductions [Mrozek et al. ‘10]
  - Edge contractions [Attali et al. ’11]
- ***Morse-based approaches*** [Robins et al. ’11; Harker et al. ’14; Fugacci et al. ’14]



Persistent Homology Computation

Coarsening Approaches:

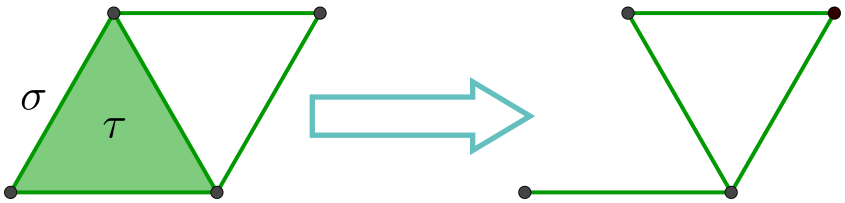
- ***Topological operators and simplifications*** [Dłotko, Wagner ’14]
  - *Acyclic subcomplexes* [Mrozek et al. ‘08]
  - Reductions and coreductions [Mrozek et al. ‘10]
  - Edge contractions [Attali et al. ’11]
- ***Morse-based approaches*** [Robins et al. ’11; Harker et al. ’14; Fugacci et al. ’14]

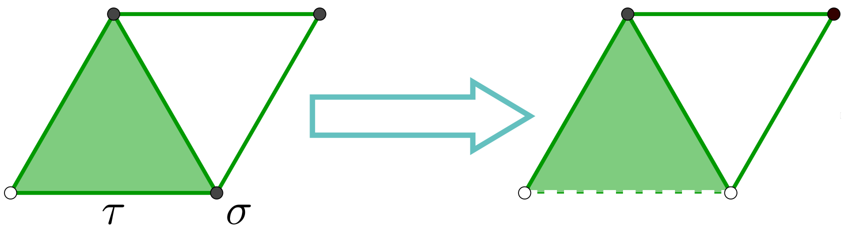


Persistent Homology Computation

Coarsening Approaches:

- ***Topological operators and simplifications*** [Dłotko, Wagner ’14]
  - Acyclic subcomplexes [Mrozek et al. ‘08]
  - ***Reductions and coreductions*** [Mrozek et al. ‘10]
  - Edge contractions [Attali et al. ’11]
- ***Morse-based approaches*** [Robins et al. ’11; Harker et al. ’14; Fugacci et al. ’14]

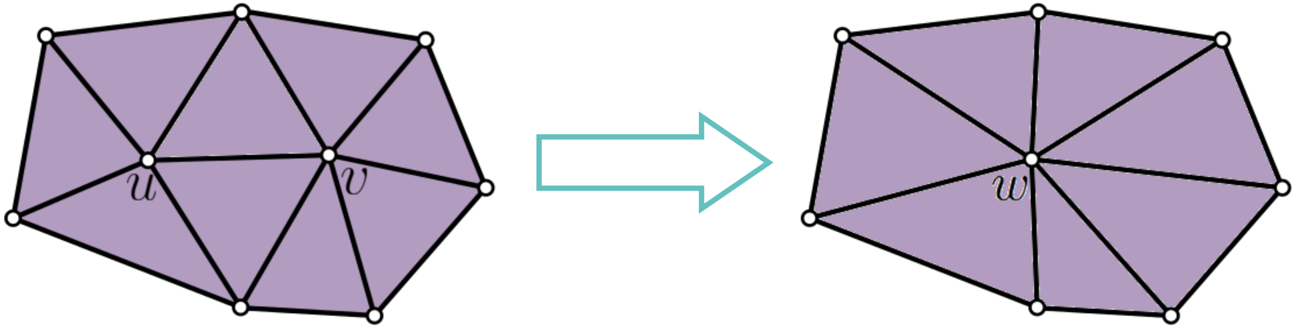




Persistent Homology Computation

Coarsening Approaches:

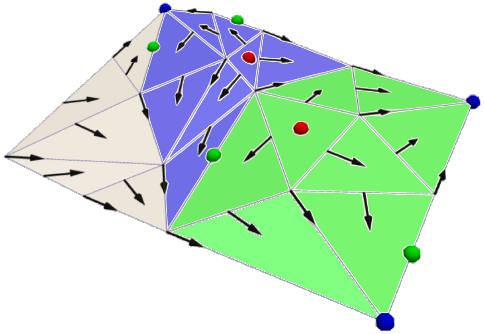
- ***Topological operators and simplifications*** [Dłotko, Wagner '14]
  - Acyclic subcomplexes [Mrozek et al. '08]
  - Reductions and coreductions [Mrozek et al. '10]
  - *Edge contractions* [Attali et al. '11]
- ***Morse-based approaches*** [Robins et al. '11; Harker et al. '14; Fugacci et al. '14]



Persistent Homology Computation

Coarsening Approaches:

- ***Topological operators and simplifications*** [Dłotko, Wagner ’14]
  - Acyclic subcomplexes [Mrozek et al. ‘08]
  - Reductions and coreductions [Mrozek et al. ‘10]
  - Edge contractions [Attali et al. ’11]
- ***Morse-based approaches*** [Robins et al. ’11; Harker et al. ’14; Fugacci et al. ’14]



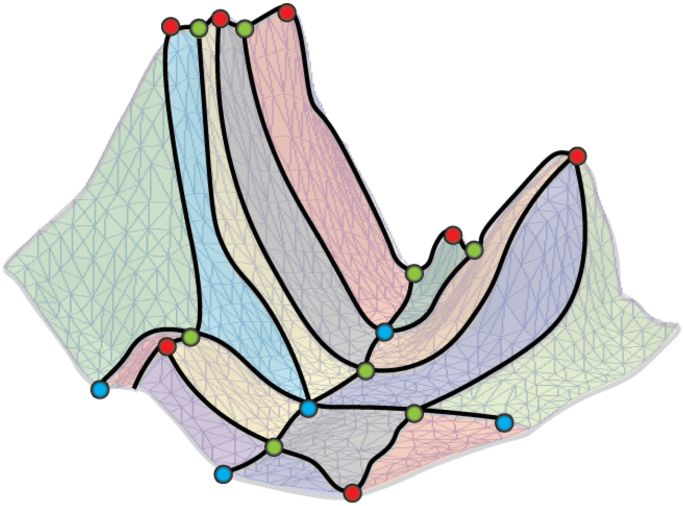
Bibliography

Some References:

- ***Persistent Homology Computation:***

  - A. Zomorodian, G. Carlsson. ***Computing persistent homology.*** Discrete & Computational Geometry, 33.2, pages 249-274, 2005.
  - N. Otter, M.A. Porter, U. Tillmann, P. Grindrod, H.A. Harrington. ***A roadmap for the computation of persistent homology.*** EPJ Data Science, 6.1, 2017.

Possible Topics for Seminars



Discrete Morse Theory

Study the shape of a space by studying the behavior of a function defined on it

Possible Topics for Seminars

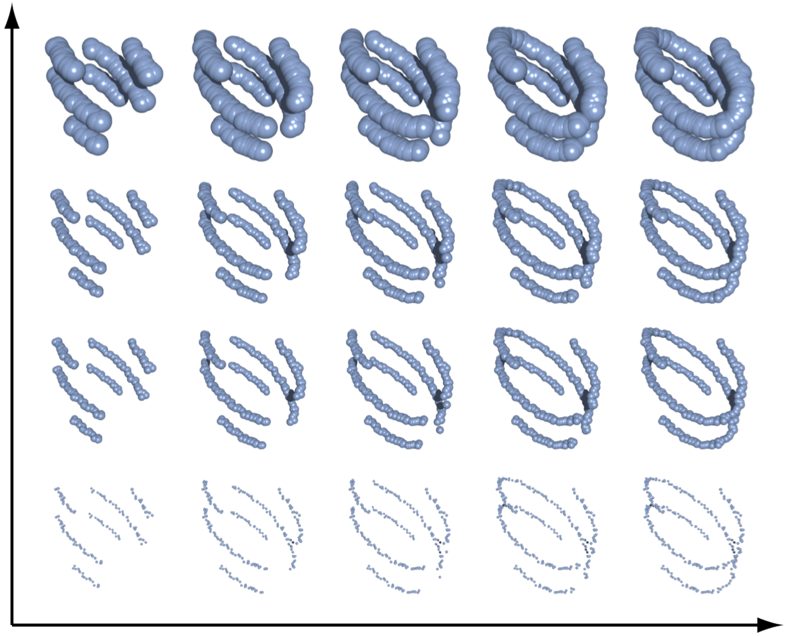
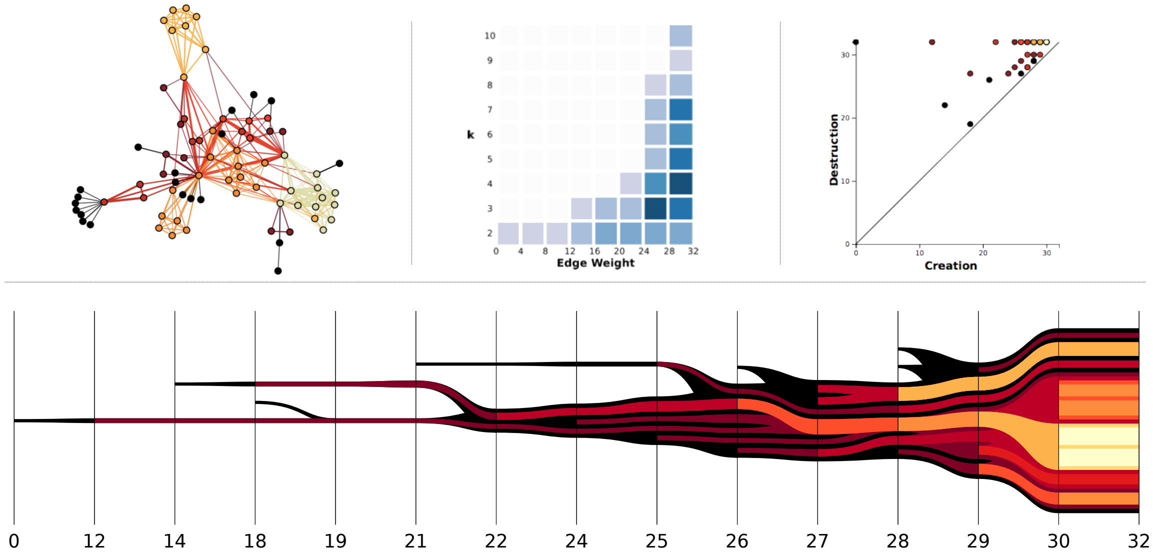


Image courtesy of
[Carlsson & Zomorodian 2009]

Multi-Parameter Persistent Homology

What if we consider multiple filtering functions?

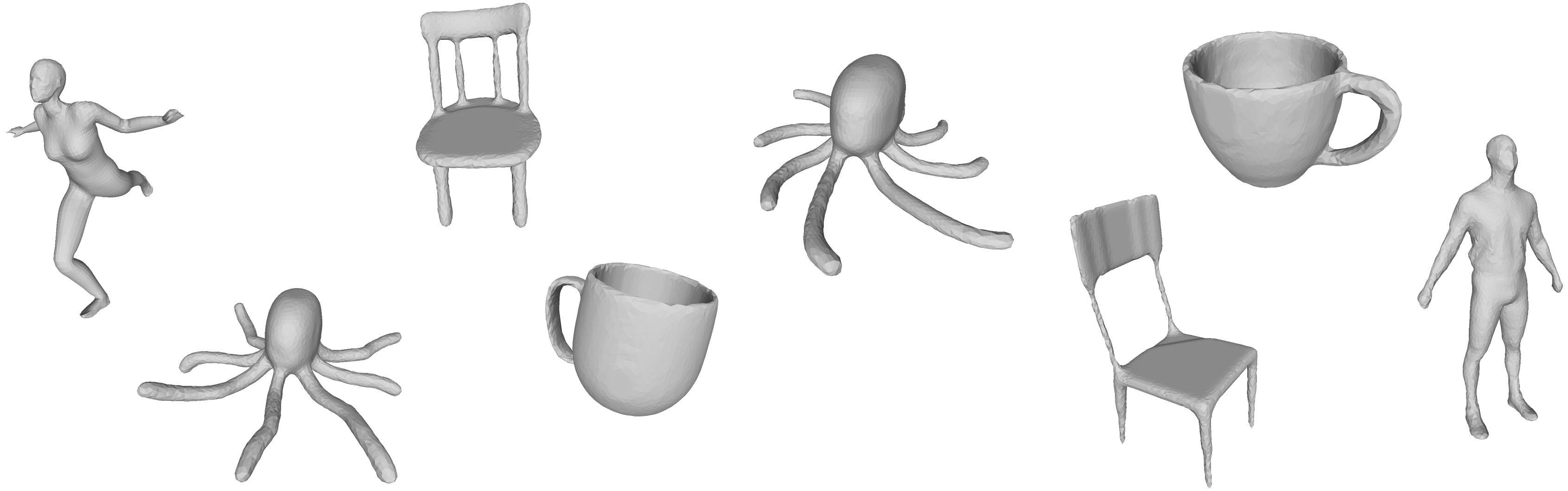
Possible Topics for Seminars



Persistent Homology & Networks

- ***Homological Scaffolds:*** *Topological summaries of weighted graphs*
- ***Clique Community Persistence:*** *Tracking the evolution of network communities*

Possible Topics for Seminars



Algorithms & Implementation

- *Efficient computation of* ***Vietoris-Rips complexes*** *and other data-to-complex strategies*
- *Focus on a specific* ***algorithm for speed-up persistent homology computation***
- *Use of available* ***software tools*** *for testing persistent homology on various datasets*