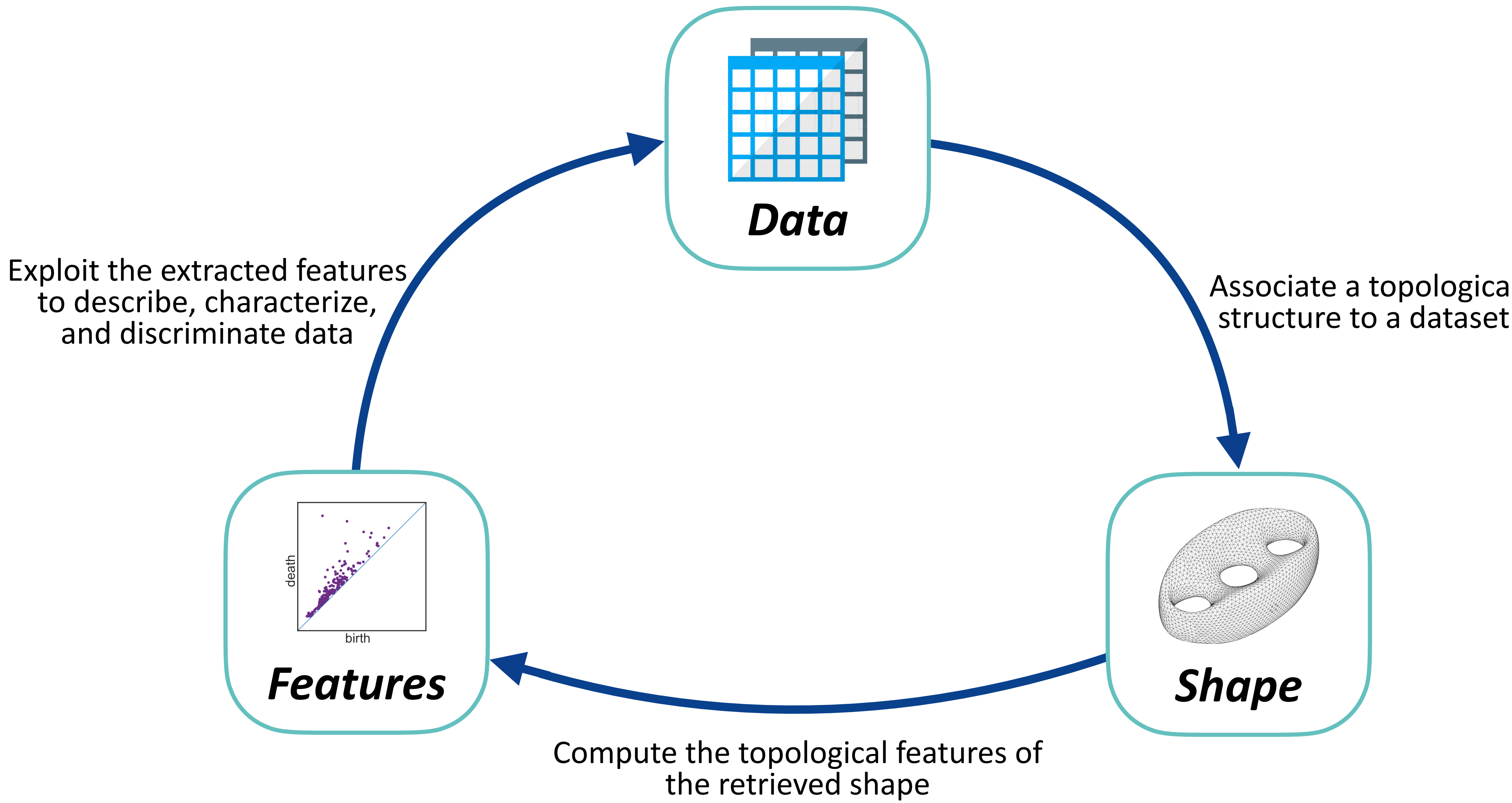
Topological Data Analysis

Algorithms & Structures

Ulderico Fugacci

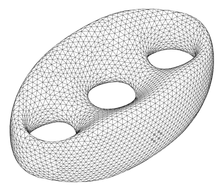
CNR - IMATI

Topological Data Analysis

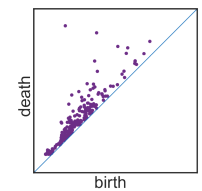


Algorithms & Structures

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:

Today, we address two main questions:

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

Algorithms & Structures

- *Computation of Persistent Homology*
- *Data Structures for Arbitrary Simplicial Complexes*

Algorithms & Structures

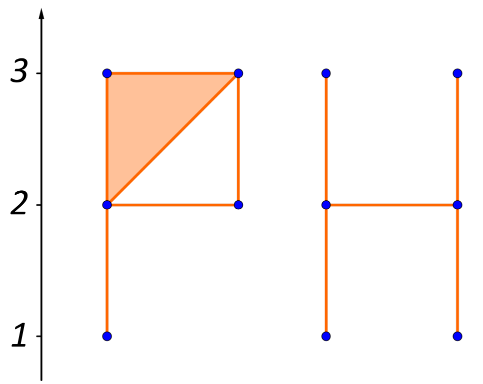
- ***Computation of Persistent Homology***
- *Data Structures for Arbitrary Simplicial Complexes*

Persistent Homology Computation

Standard Algorithm:

[Zomorodian & Carlsson 2005]

From:



To:

H_0 $[1, 2]$ $[1, \infty)$ $[1, \infty)$

H_1 $[3, \infty)$

$i\backslash j$	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

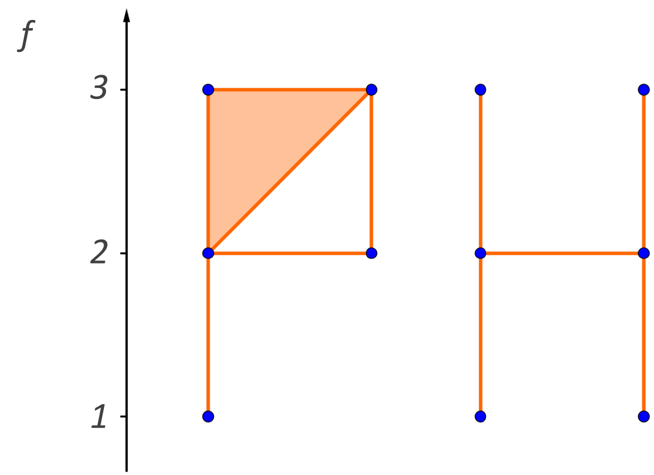
Compute a ***reduced boundary matrix*** for K^f from which easily read the persistence pairs

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^f:

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

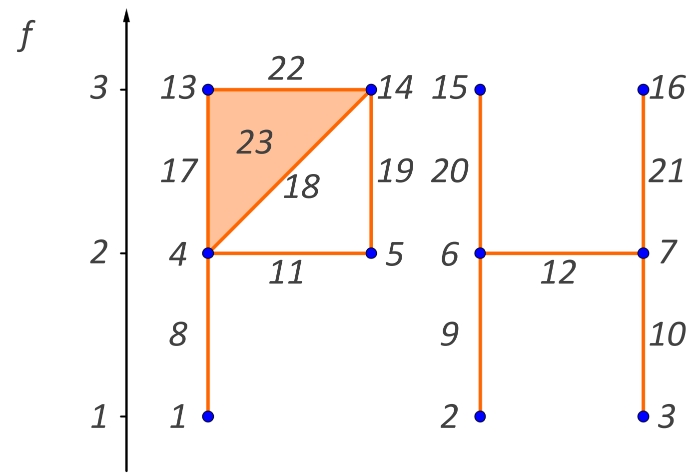
- *if $f(\sigma_i) < f(\sigma_j)$, then $i < j$*
- *if σ_i is a proper face of σ_j, then $i < j$*

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:

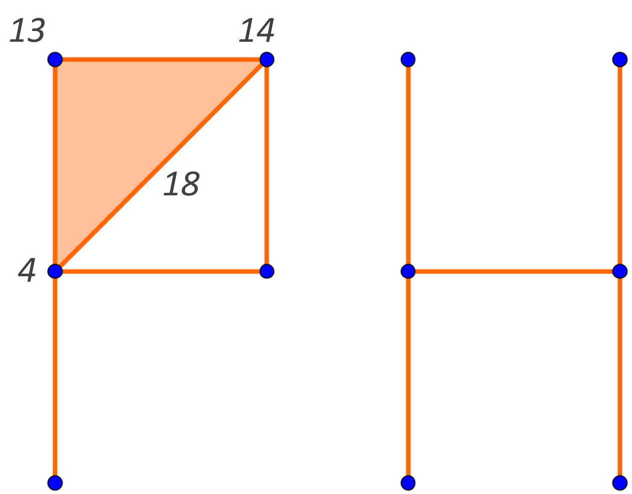
- $f(\sigma) < f(\sigma')$
- $f(\sigma) = f(\sigma')$ and $\dim(\sigma) < \dim(\sigma')$
- $f(\sigma) = f(\sigma')$, $\dim(\sigma) = \dim(\sigma')$, *and σ precedes σ' w.r.t. the* ***lexicographic order*** *of their vertices*

Persistent Homology Computation

Boundary Matrix:

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

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



E.g.

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

Persistent Homology Computation

Reduced Matrix:

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

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

*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)

M is **not** reduced

Persistent Homology Computation

Reduction Algorithm:

```
Matrix R = M
for j = 1, …, n do
  while ∃ 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 ***M*,** where

M is the ***boundary matrix*** of K^f

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*** $\longrightarrow$ ***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* $\longrightarrow$ ***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 $i = 1, \ldots, n$,

  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^f**:

$[i, j]$ 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

$[1, \infty)$

$[2, \infty)$

$[3, 12]$

$[4, 8]$

$[5, 11]$

$[6, 9]$

$[7, 10]$

$[13, 17]$

$[14, 18]$

$[15, 20]$

$[16, 21]$

H_1

$[19, \infty)$

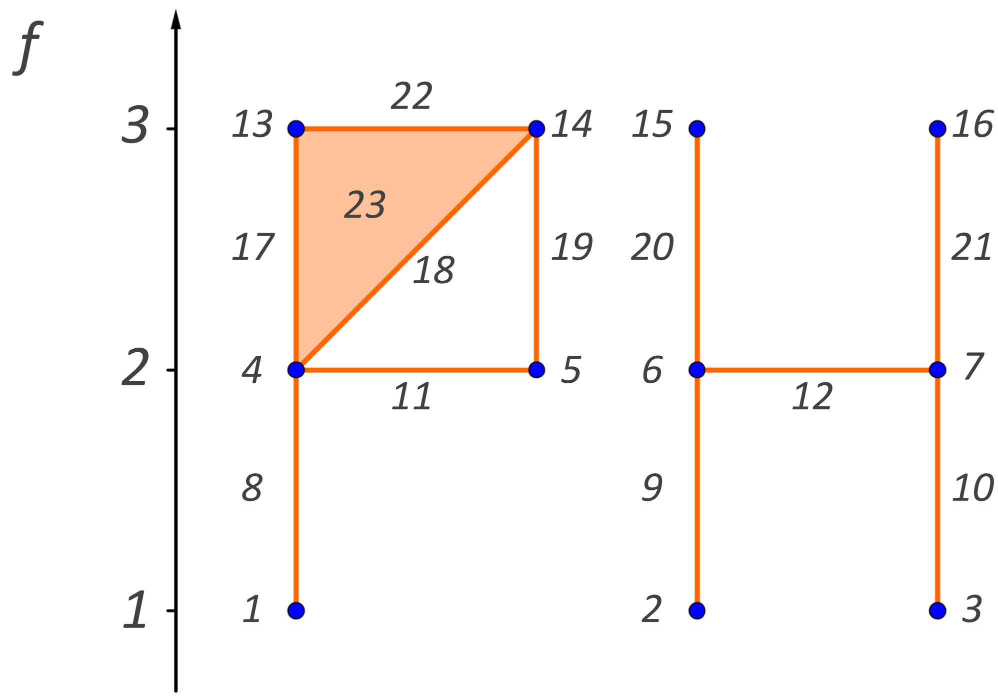
$[22, 23]$

$i \backslash j$	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

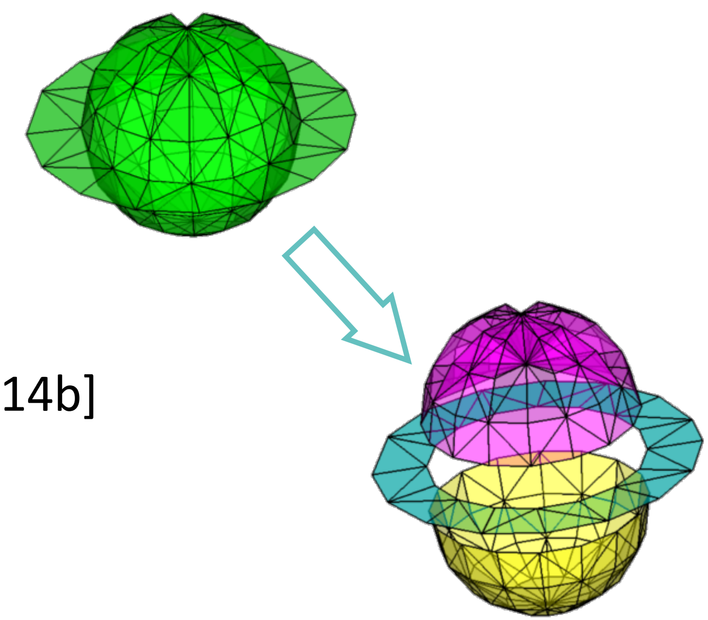
Direct Approaches:

- ***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:

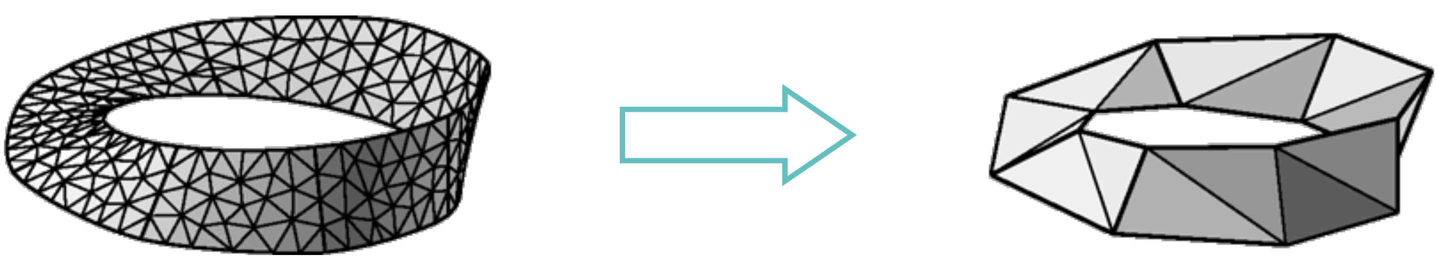
- ***Spectral sequences*** [Edelsbrunner, Harer ’08; Lipsky et al. ’11]
- ***Constructive Mayer-Vietoris*** [Boltcheva et al. ’11]
- ***Multicore coreductions*** [Murty et al. ’13]
- ***Multicore homology*** [Lewis, Zomorodian ’14]
- ***Persistent homology in chunks*** [Bauer et al. ‘14a]
- ***Distributed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



Persistent Homology Computation

Coarsening Approaches:

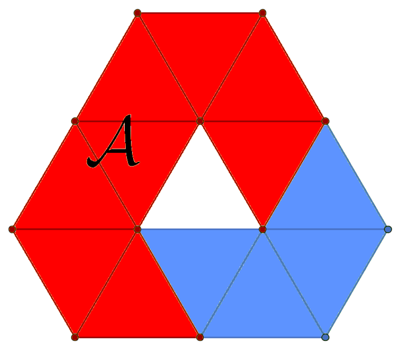
- ***Topological operators and simplifications*** [Dłotko, Wagner ’14]
  - Acyclic subcomplexes [Mrozek et al. ‘08]
  - Reductions and coreductions [Mrozek et al. ‘10]
  - Edge contractions [Attali et al. ’11]
- ***Morse-based approaches*** [Robins et al. ’11; Harker et al. ’14; Fugacci et al. ’14]



Persistent Homology Computation

Coarsening Approaches:

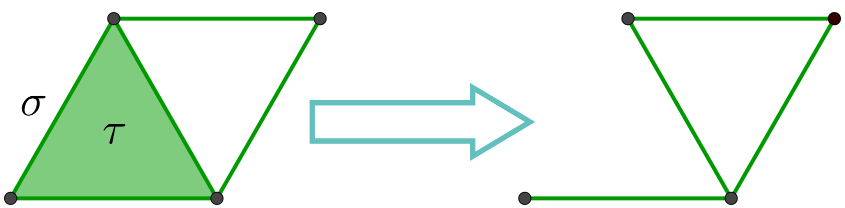
- ***Topological operators and simplifications*** [Dłotko, Wagner '14]
  - *Acyclic subcomplexes* [Mrozek et al. '08]
  - Reductions and coreductions [Mrozek et al. '10]
  - Edge contractions [Attali et al. '11]
- ***Morse-based approaches*** [Robins et al. '11; Harker et al. '14; Fugacci et al. '14]

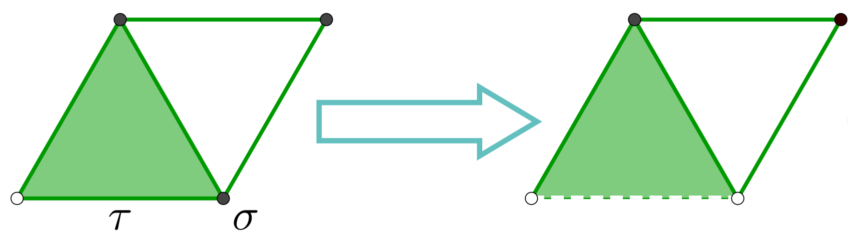


Persistent Homology Computation

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]

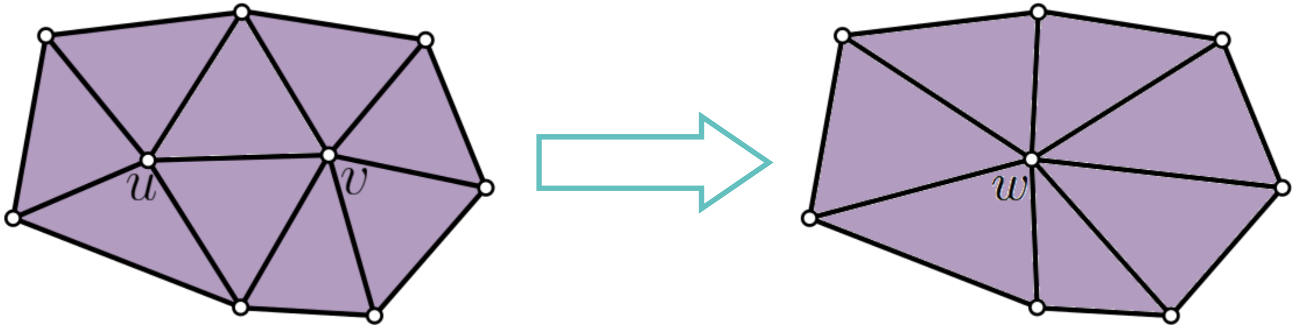




Persistent Homology Computation

Coarsening Approaches:

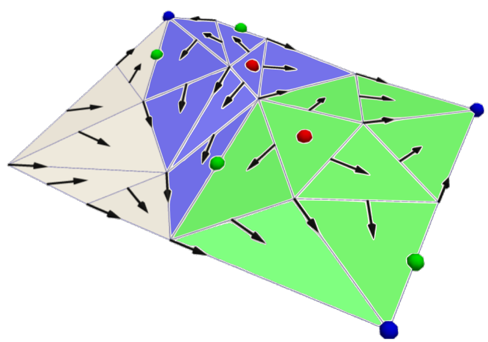
- ***Topological operators and simplifications*** [Dłotko, Wagner '14]
  - Acyclic subcomplexes [Mrozek et al. '08]
  - Reductions and coreductions [Mrozek et al. '10]
  - *Edge contractions* [Attali et al. '11]
- ***Morse-based approaches*** [Robins et al. '11; Harker et al. '14; Fugacci et al. '14]



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]



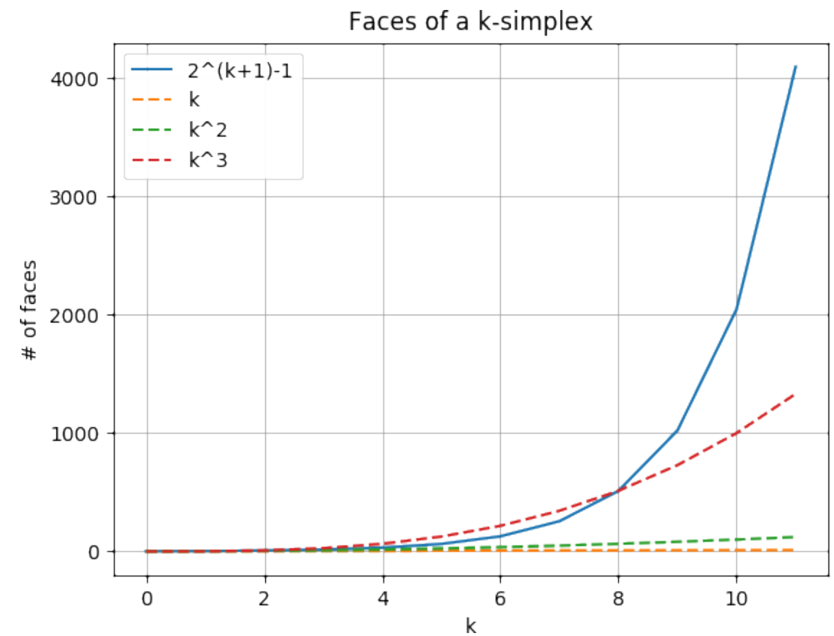
Algorithms & Structures

- *Computation of Persistent Homology*
- *Data Structures for Arbitrary Simplicial Complexes*

Encoding Simplicial Complexes

Issue:

It is enough to have a point cloud consisting of at least ***30 points*** *for having to deal with an associated filtered simplicial complex of more than a* ***billion*** *of simplices*



Solution:

Development of ***compact*** *and* ***efficient data structures*** *for encoding arbitrary simplicial complexes*

Encoding Simplicial Complexes

Outline:

- ***Which info to be stored?***
- ***Data Structures***
  - *Simplex-based representations*
  - *Top-based representations*
  - *Operator-driven representations*
- ***Comparisons***
- ***Issues and solutions in adopting top-based representations***

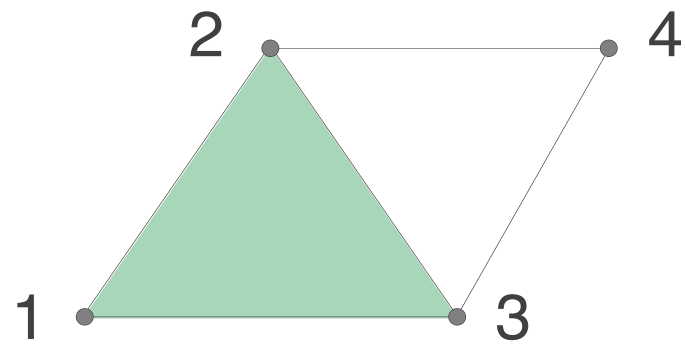
Out Of Scope:

- ***Data structures for specific classes of complexes***
  - *E.g. manifold or complexes of low dimension*

Encoding Simplicial Complexes

Data Structure:

The *entities* which a simplicial complex consists of are:



- its ***simplices***

$$K = K_0 \cup K_1 \cup \ldots \cup K_d$$

where K_i is the collection of the i-simplices of K

- the ***topological relations***

$$R_{i,j} \subseteq K_i \times K_j$$

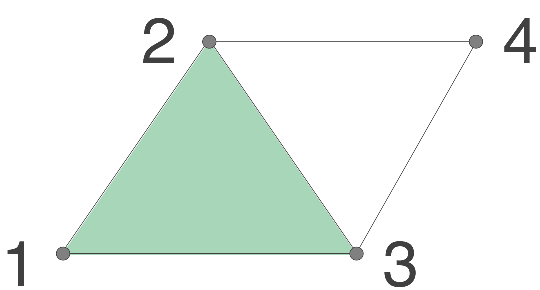
between the simplices of K encoding the (co-)boundary of each simplex

*A **data structure** for K has to explicitly **store** a portion of the above information and to (efficiently) **retrieve** the remaining part*

Encoding Simplicial Complexes

Topological Relations: Given an i-simplex σ and a j-simplex τ of K,

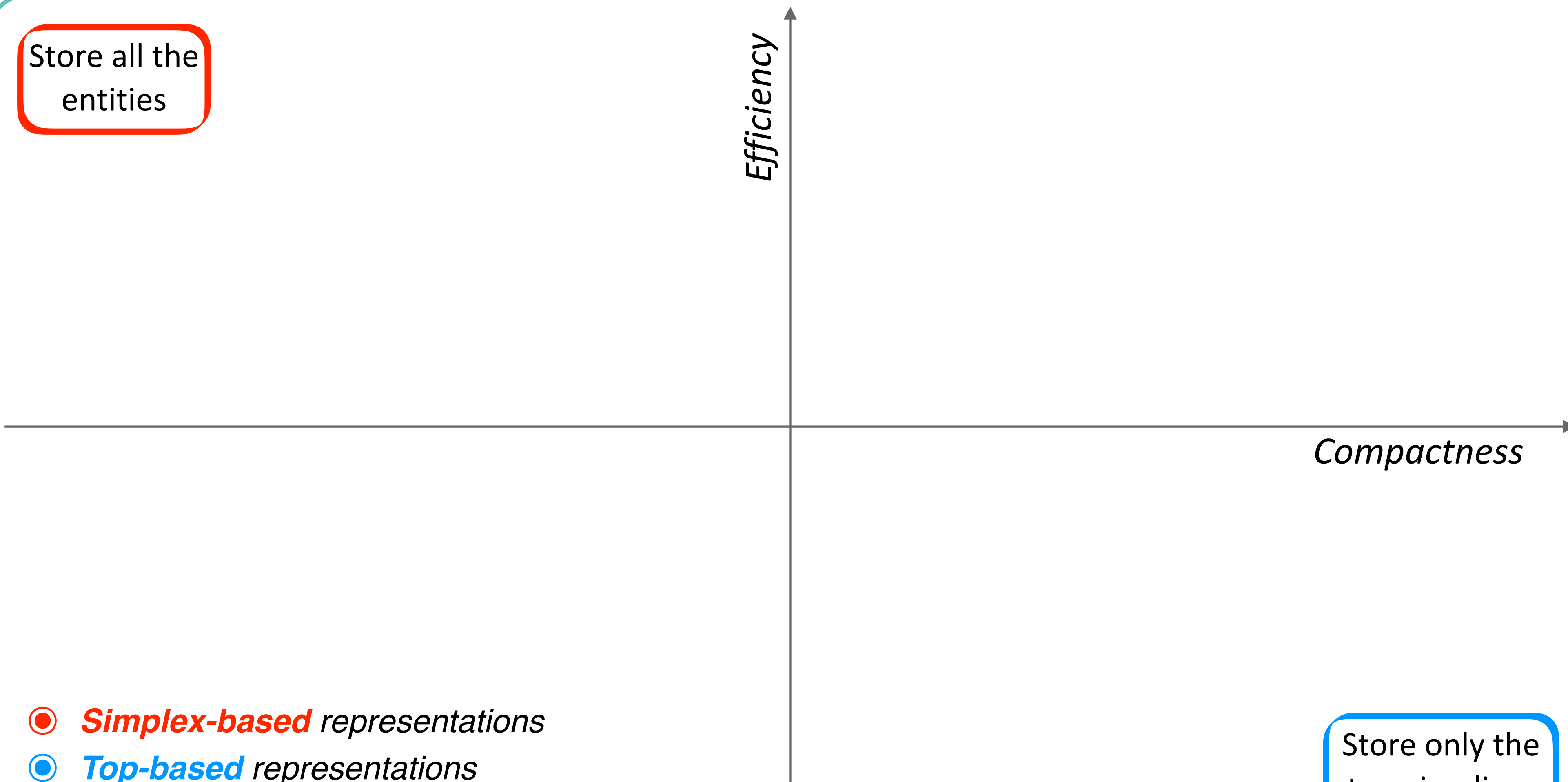
$$(\sigma, \tau) \in R_{i,j} \iff \begin{cases} \sigma \subseteq \tau & \text{for } i < j \\ |\sigma \cap \tau| = i \ \ (\text{equivalently, } \sigma \cap \tau \in K_{i-1}) & \text{for } i = j \\ \tau \subseteq \sigma & \text{for } i > j \end{cases}$$



$(12, 123) \in R_{1,2}$ $(12, 24) \in R_{1,1}$ $(12, 1) \in R_{1,0}$

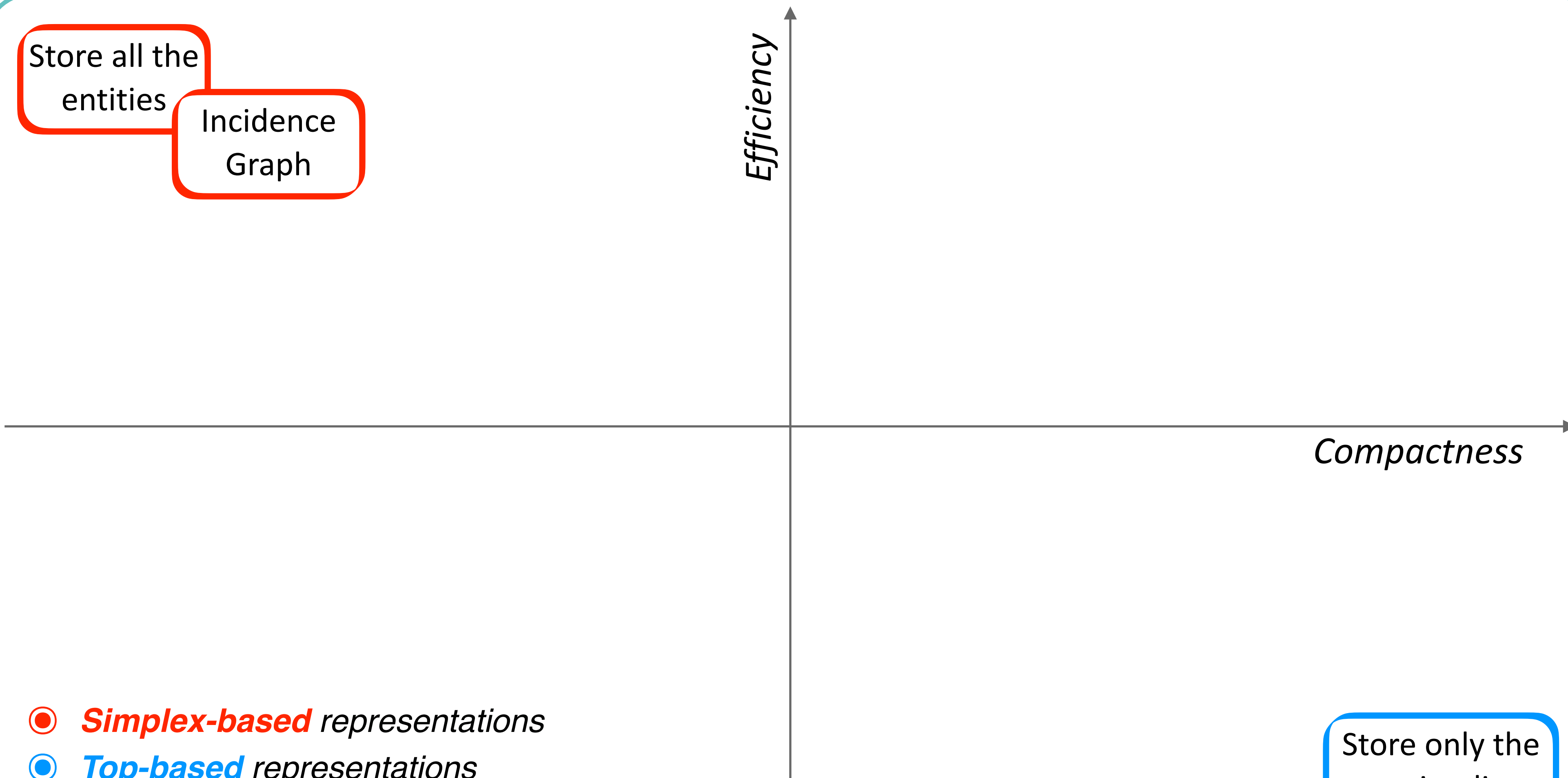
An i-simplex σ is called a ***top simplex*** *of K if there is no simplex τ of K such that $(\sigma, \tau) \in R_{i,i+1}$*

Encoding Simplicial Complexes



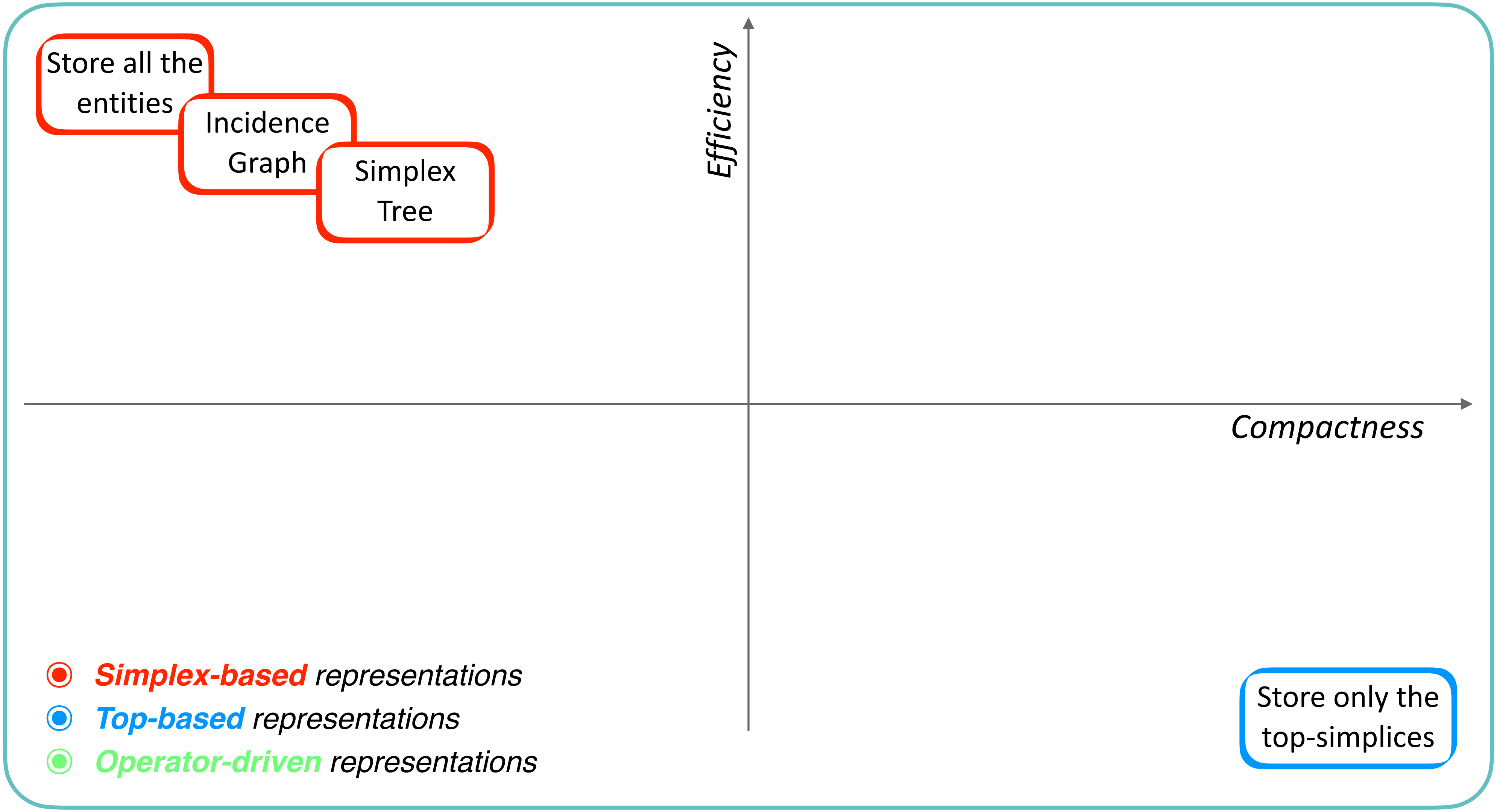
- ***Simplex-based*** *representations*
- ***Top-based*** *representations*
- ***Operator-driven*** *representations*

Encoding Simplicial Complexes

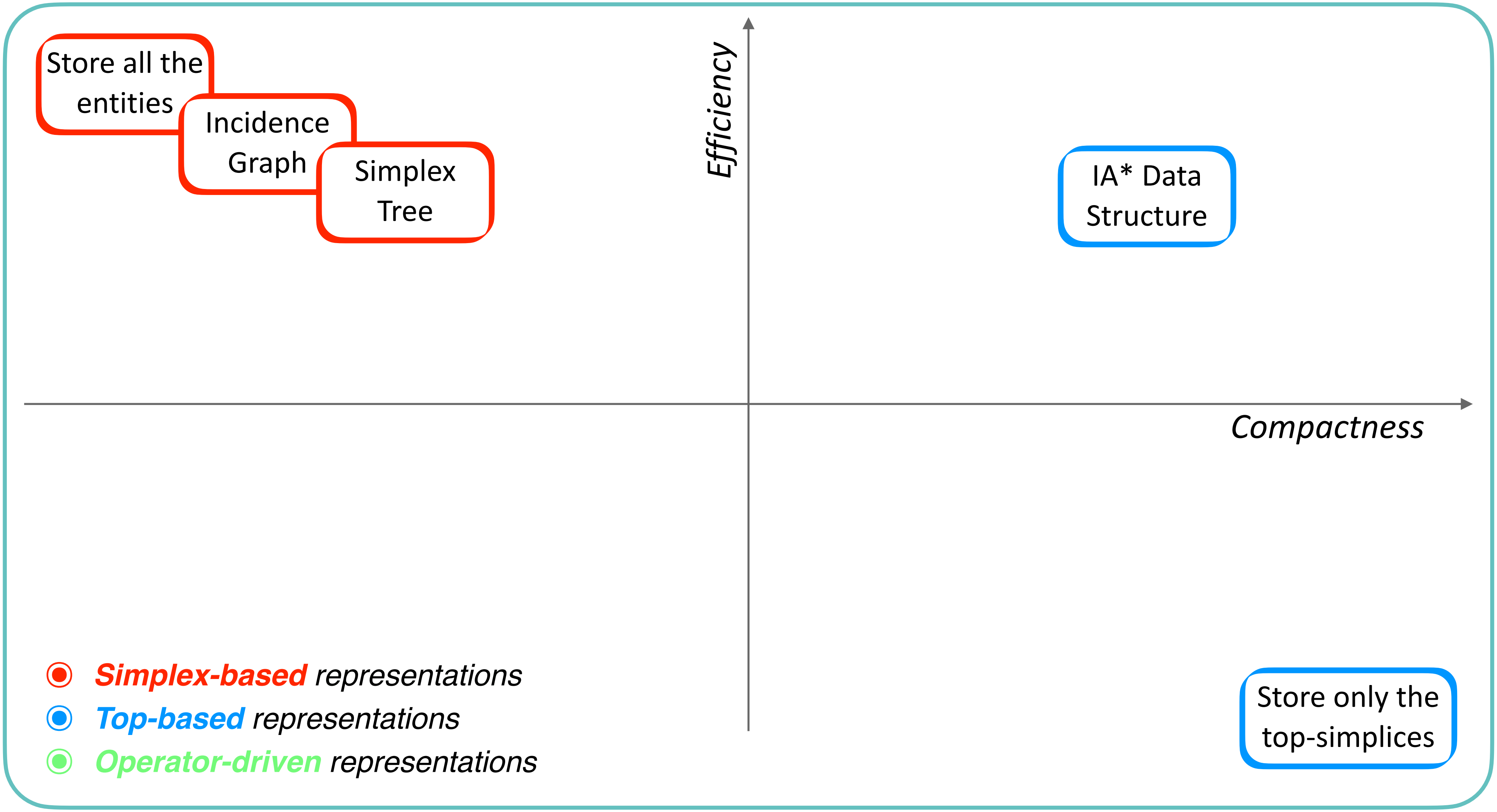


- ***Simplex-based*** *representations*
- ***Top-based*** *representations*
- ***Operator-driven*** *representations*

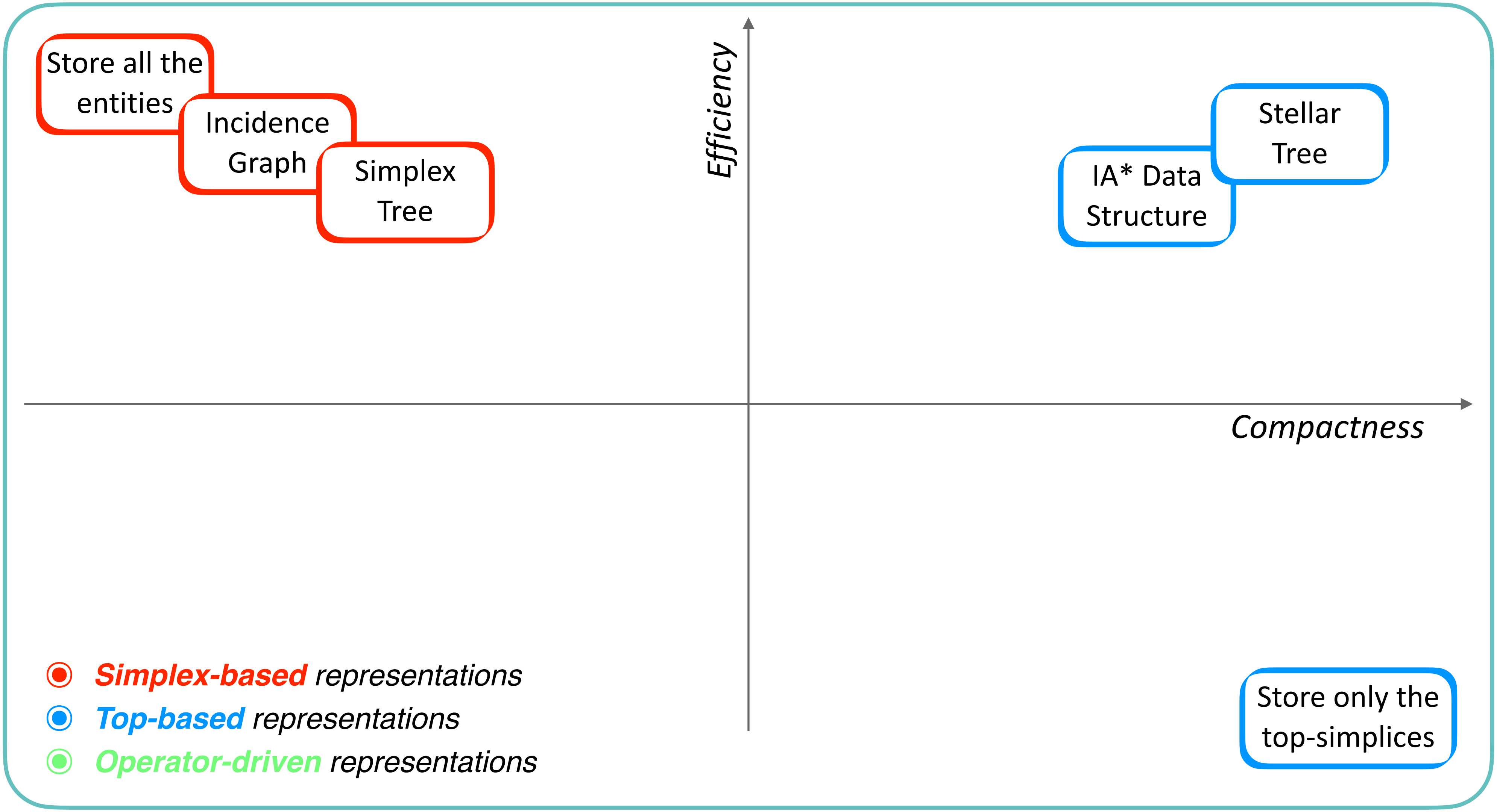
Encoding Simplicial Complexes



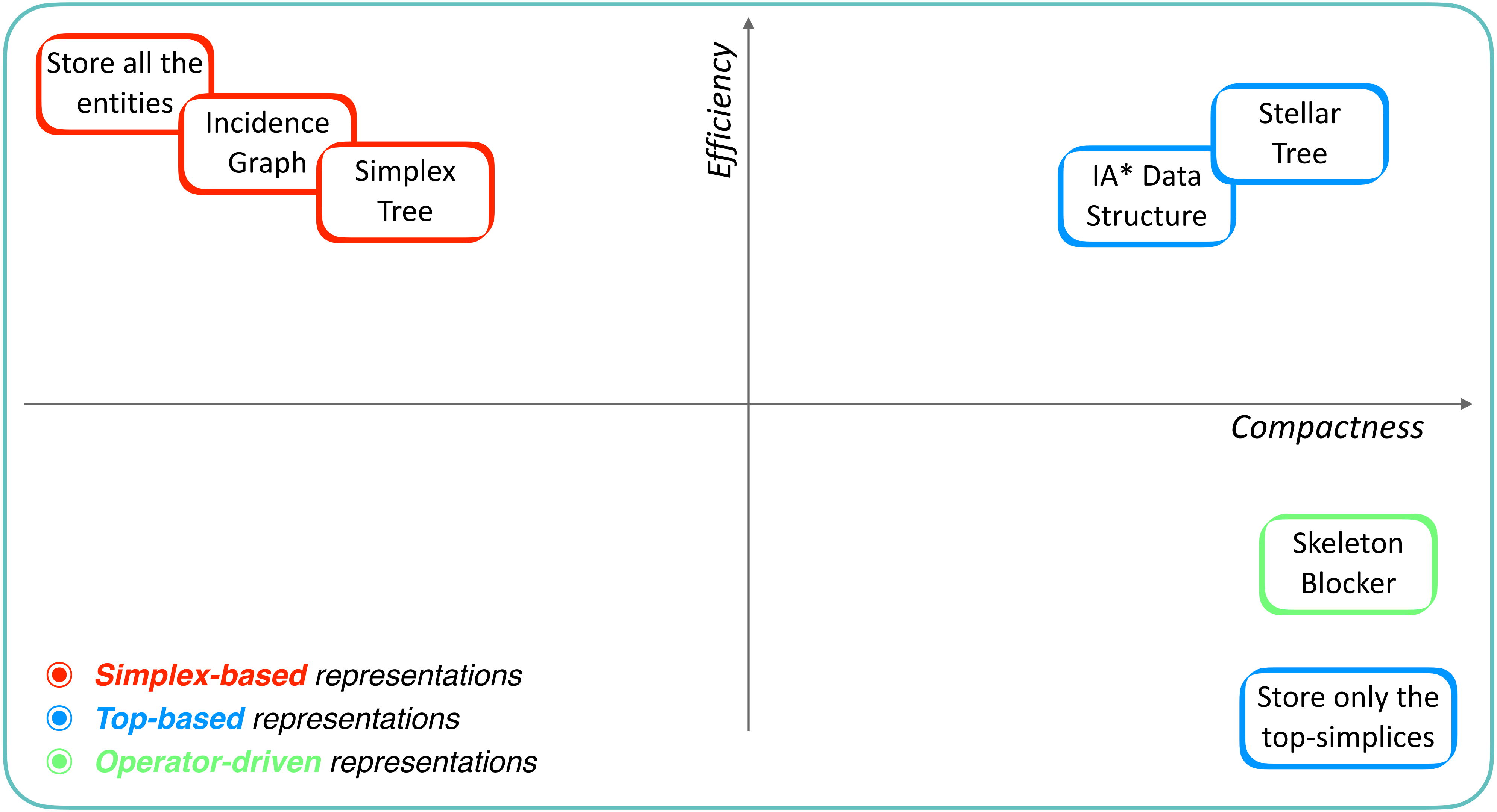
Encoding Simplicial Complexes



Encoding Simplicial Complexes

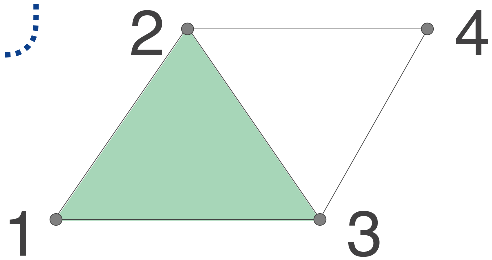


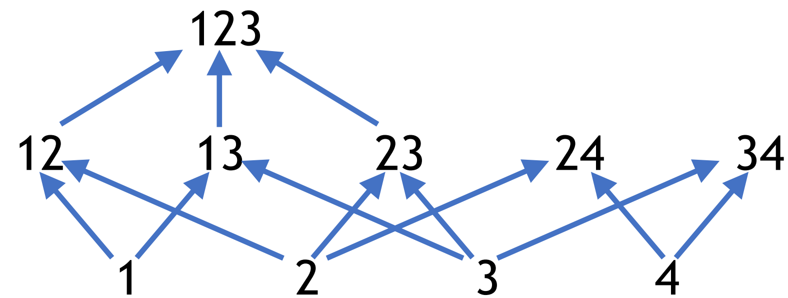
Encoding Simplicial Complexes



Simplex-based Representations

Incidence Graph:





The simplicial complex K is encoded via a ***directed graph*** ***G = (N, A)***:

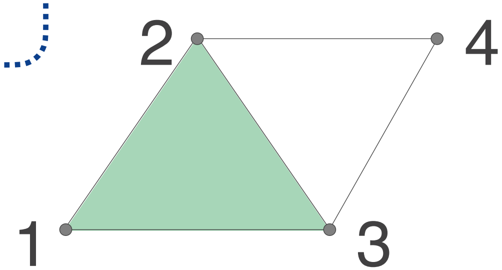
$N \longleftrightarrow K$

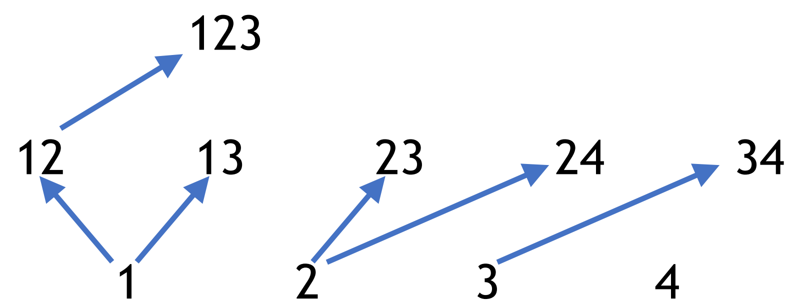
$(\sigma, \tau) \in A \longleftrightarrow (\sigma, \tau) \in R_{i,i+1}$

- *All the relations between simplices can be immediately retrieved*
- *The representation **size exponentially increases** with the complex dimension*

Simplex-based Representations

Simplex Tree:





The simplicial complex K is encoded via a ***directed graph*** ***G = (N, A)***:

$$N \longleftrightarrow K$$

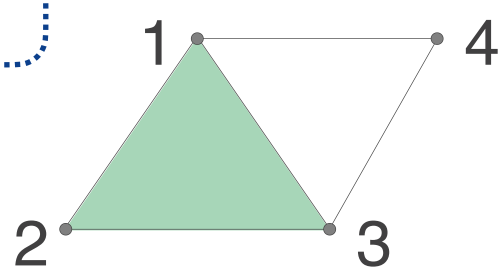
$$(\sigma, \tau) \in A \longleftrightarrow (\sigma, \tau) \in R_{i,i+1} \text{ and } I(\sigma) < I(\tau)$$

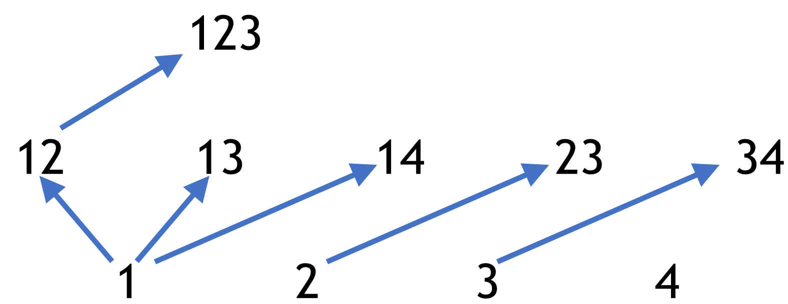
where $I(\sigma)$ denotes the ***maximum value*** taken by the vertices of σ w.r.t. a ***total order*** on K_0

*Graph is **not uniquely determined** but it depends on the chosen vertex order*

Simplex-based Representations

Simplex Tree:





The simplicial complex K is encoded via a ***directed graph*** ***G = (N, A)***:

$$N \longleftrightarrow K$$

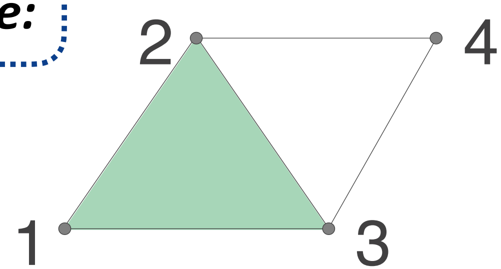
$$(\sigma, \tau) \in A \longleftrightarrow (\sigma, \tau) \in R_{i,i+1} \text{ and } I(\sigma) < I(\tau)$$

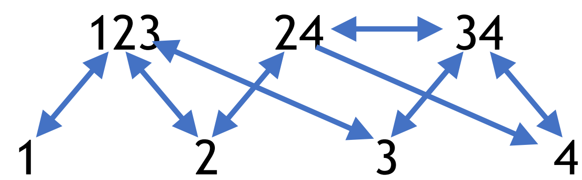
where $I(\sigma)$ denotes the ***maximum value*** taken by the vertices of σ w.r.t. a ***total order*** on K_0

*Graph is **not uniquely determined** but it depends on the chosen vertex order*

Top-based Representations

IA* Data Structure:





The simplicial complex K is encoded via a ***directed graph*** ***G = (N, A)***:

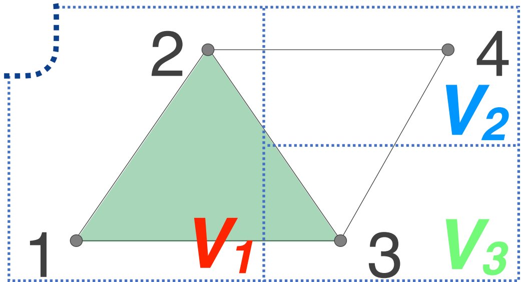
$$N \longleftrightarrow K_0 \cup K_{top}$$

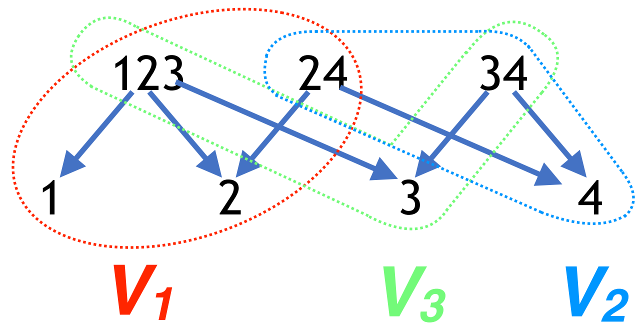
$$(\sigma, \tau) \in A \longleftrightarrow \begin{cases} \sigma \in K_{top} \text{ and } (\sigma, \tau) \in R_{i,0} \\ \sigma, \tau \in K_{top} \text{ and } (\sigma, \tau) \in R_{i,i} \\ \tau \in K_{top} \text{ and } (\sigma, \tau) \in (R_{0,i})^* \end{cases}$$

- ***Compact****: it explicitly stores just a fraction of the entities of a simplicial complex*
- ***Not*** *all the relations between simplices are* ***immediately available***

Top-based Representations

Stellar Tree:





Given a decomposition of K_0, the simplicial complex K is encoded via a ***directed graph*** ***G = (N, A)***:

$N \longleftrightarrow K_0 = V_1 \cup V_2 \cup \dots \cup V_n$

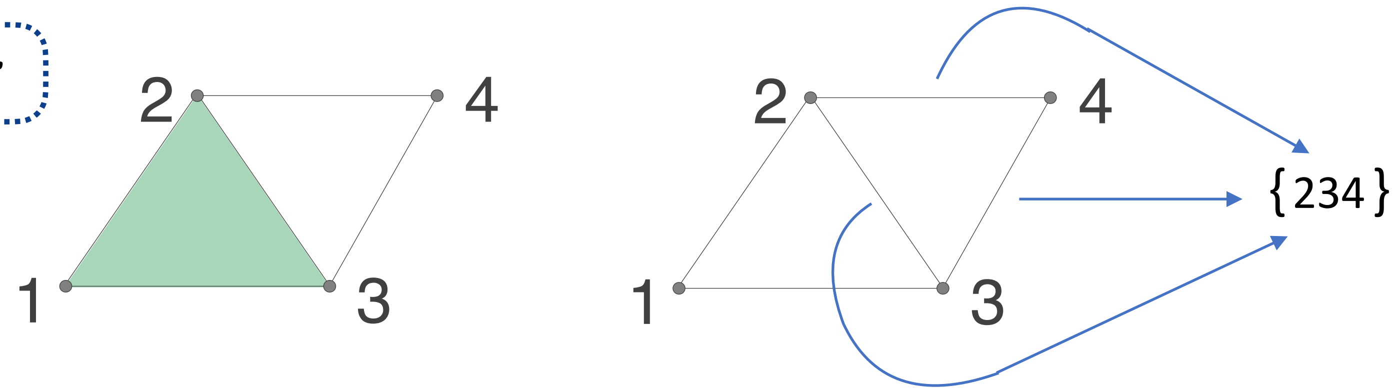
$(\sigma, \tau) \in A \longleftrightarrow \sigma \in K_{top}$ *and* $(\sigma, \tau) \in R_{i,0}$

plus a ***map*** returning, for each j, the vertices of K in V_j and the top simplices with at least one vertex in V_j

- ***Compact*** *and* ***highly adjustable*** *(e.g. choice of the decomposition, of the maximum number of vertices in each region)*
- ***Not*** *all the relations between simplices are* ***immediately available***

Operator-driven Representations

Skeleton Blocker:



The simplicial complex K is encoded by storing its ***1-skeleton*** (i.e. the graph consisting of the 0- and the 1-simplices) and a ***map*** returning, for each 1-simplex σ, the blockers of K containing σ, where:

*A simplex τ is a **blocker** if τ does not belong to K but all its faces do*

- ***Designed for*** flag complexes (e.g. ***VR complexes***) and edge contraction
- Too specific: ***inefficient in any other task***

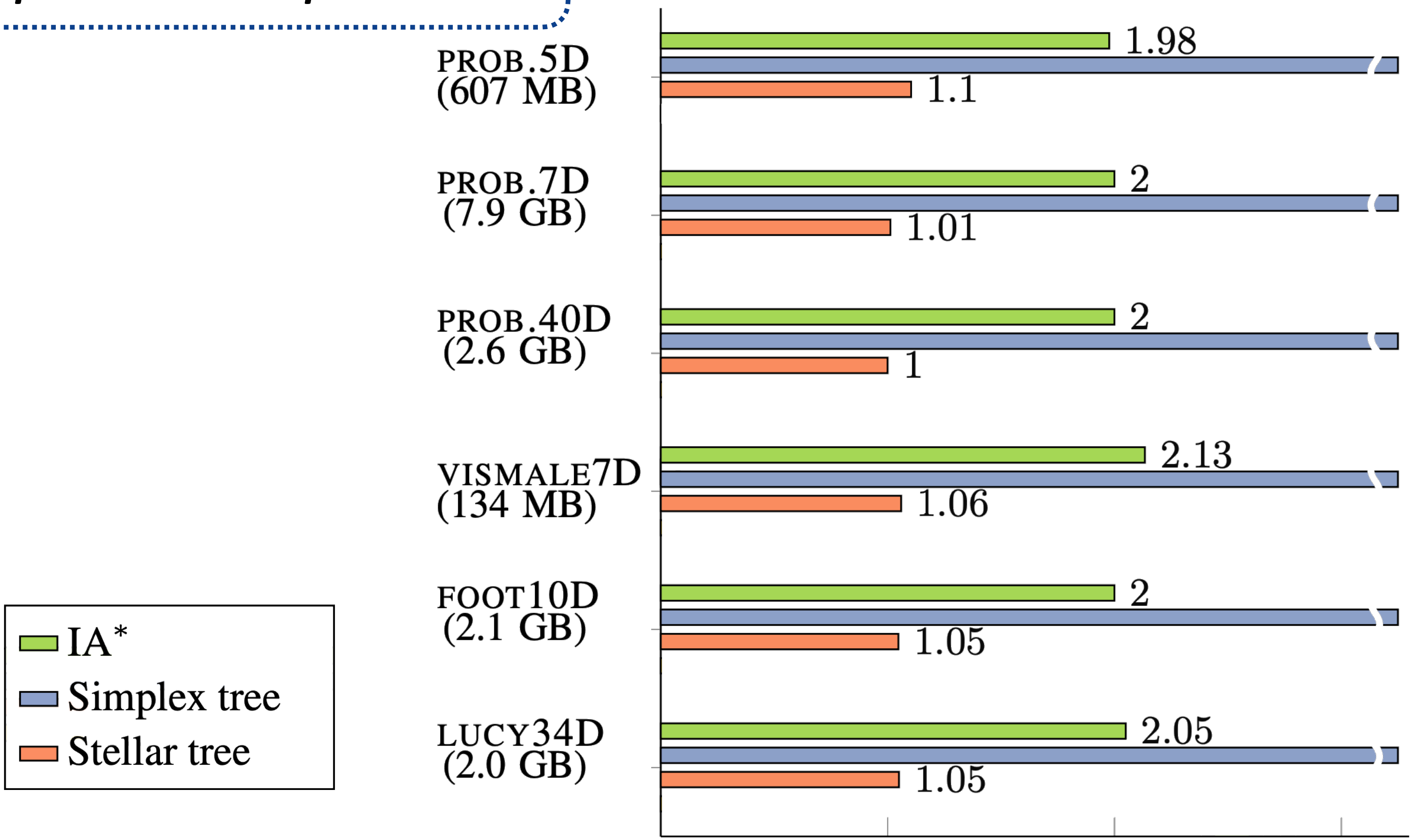
Encoding Simplicial Complexes

Top-based vs Simplex-based:

Dataset	d	$\|\Sigma_0\|$	$\|\Sigma_{top}\|$	$\|\Sigma\|$	Storage Cost		
					IA^*	IG	ST
DTI-SCAN	3	0.9M	5.5M	24M	0.97	11.9	2.4
VISMALE	3	4.6M	26M	118M	4.7	-	9.7
ACKLEY4	4	1.5M	32M	204M	6.8	-	12.8
AMAZON01	6	0.2M	0.4M	2.2M	0.12	1.6	0.3
AMAZON02	7	0.4M	1.0M	18.4M	0.28	9.8	1.5
ROADNET	3	1.9M	2.5M	4.8M	0.8	3.3	1.0
SPHERE-1.0	16	100	224	0.6M	0.003	0.9	0.04
SPHERE-1.2	21	100	285	26M	0.0032	-	1.5
SPHERE-1.3	23	100	382	197M	0.0034	-	11.01

Encoding Simplicial Complexes

Top-based vs Simplex-based:



Encoding Simplicial Complexes

Top-based vs Operator-driven:

data	ω		contr. edges	timings: check	timings: contr.	timings: tot	memory peak: gen.	memory peak: simpl.
CHICAGO	28	weak	6.38K	9.15h	2.27m	9.19h	5.6	57.2K
		top		0.01s	0.02s	0.09s		7.6
		Skel.		0.00s	0.15s	0.15s	7.8	7.8
	56	weak	7.99K	out-of-memory			6.2	–
		top		0.04s	0.06s	0.23s		10.8
		Skel.		0.00s	0.71s	0.71s	14.1	14.1
ATHENS	63	weak	27.9K	out-of-memory			11.6	–
		top		0.08s	0.11s	0.38s		14.9
		Skel.		0.00s	0.74s	0.75s	26.4	26.8
	126	weak	31.2K	out-of-memory			10.0	–
		top		0.40s	0.49s	1.36s		25.9
		Skel.		0.01s	7.73s	7.74s	66.1	66.7
VISMALE	3.5	weak	4.23M	34.3m	1.28m	40.4m	1.0K	2.0K
		top		4.34m	0.89m	7.20m		2.0K
		Skel.		0.76m	3.34h	3.35h	8.0K	8.0K
FOOT	4.5	weak	4.69M	killed after 25 hours			7.5K	–
		top		2.89h	26.0m	3.32h		10.7K
		Skel.		killed after 25 hours			19.4K	–
LUCY	1.5	weak	14.0M	killed after 25 hours			7.5K	–
		top		11.9m	14.8m	32.0m		15.4K
		Skel.		23.19s	14.6h	14.6h	50.9K	52.1K

Encoding Simplicial Complexes

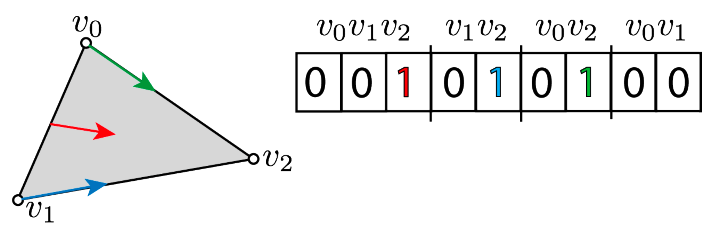
Possible Issues in Top-based Representations:

Top-based representations are promising data structures for encoding a simplicial complex K

but, how to …

- ***Store information associated to each simplex of K (e.g. labels, gradient, …)?***

Attach information to the top simplices only



- ***Efficiently perform operators having explicitly stored a fraction of the entities of K?***

Re-define the algorithms performing the operators trying to extract the lowest possible amount of non-explicitly stored entities

Bibliography

General References:

- ***Books on TDA:***
  - A. J. Zomorodian. ***Topology for computing***. Cambridge University Press, 2005.
  - H. Edelsbrunner, J. Harer. ***Computational topology: an introduction***. American Mathematical Society, 2010.
  - R. W. Ghrist. ***Elementary applied topology***. Seattle: Createspace, 2014.
- ***Papers on TDA:***
  - G. Carlsson. ***Topology and data***. Bulletin of the American Mathematical Society 46.2, pages 255-308, 2009.

Today's 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.

Bibliography

Today's References:

- ***Data Structures for Arbitrary Simplicial Complexes:***
  - D. Canino, L. De Floriani, K. Weiss. ***IA*: an adjacency-based representation for non-manifold simplicial shapes in arbitrary dimensions.*** Computers & Graphics, 35.3, pages 747-753, 2011.
  - D. Attali, A. Lieutier, D. Salinas. ***Efficient data structure for representing and simplifying simplicial complexes in high dimensions.*** International Journal of Computational Geometry & Applications, 22.4, pages 279-303, 2012.
  - J.D. Boissonnat, C. Maria. ***The simplex tree: An efficient data structure for general simplicial complexes.*** Algorithmica, 70.3, pages 406-427, 2014.
  - R. Fellegara, K. Weiss, L. De Floriani. **The Stellar tree: a compact representation for simplicial complexes and beyond.** arXiv preprint:1707.02211, 2017.
  - U. Fugacci, F. Iuricich, L. De Floriani. ***Computing discrete Morse complexes from simplicial complexes.*** Graphical models, 103, 101023, 2019.
  - R. Fellegara, F. Iuricich, L. De Floriani, U. Fugacci. ***Efficient Homology-Preserving Simplification of High-Dimensional Simplicial Shapes.*** Computer Graphics Forum, 39.1, pages 244-259, 2020.