“Persistent Homology” Summer School - Rabat

Persistent Homology Computation and Discrete Morse Theory

Ulderico Fugacci

Kaiserslautern University of Technology

Department of Computer Science

Technische Universität Kaiserslautern

July 4, 2017

Outline

Persistent Homology

Computation

Outline

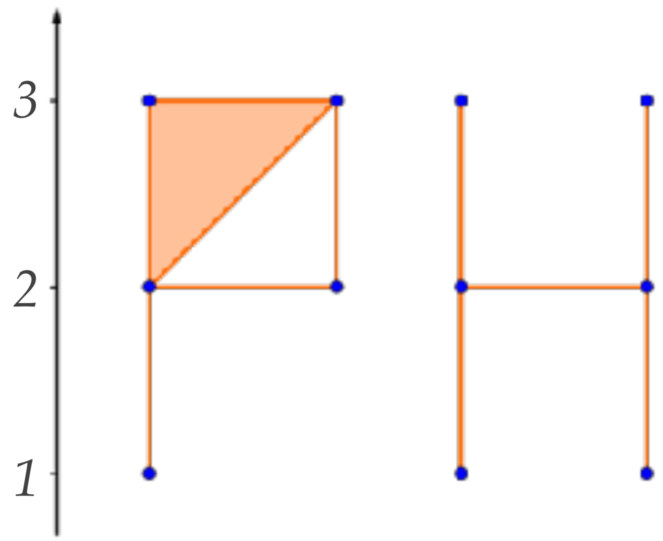
Persistent Homology
Computation

Discrete Morse Theory and
Persistent Homology

Computing Persistent Homology

Standard Algorithm: [Edelsbrunner et al. 2002; Zomorodian, Carlsson 2005]

From:



To:

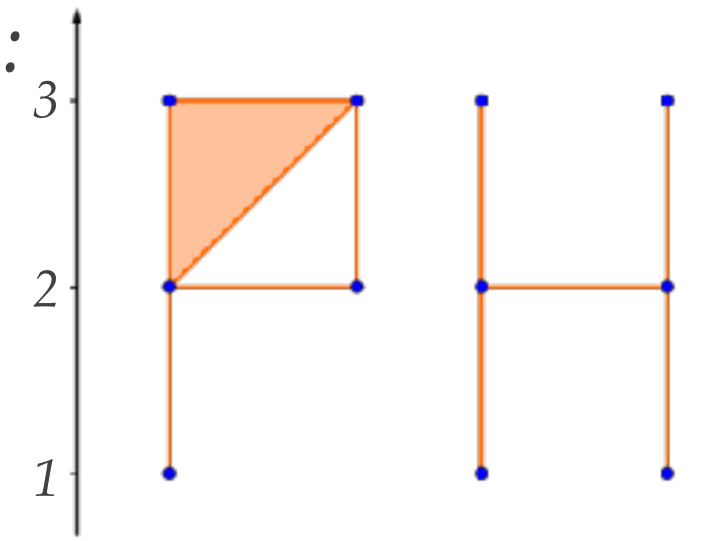
H_0 $[1, 2]$, $[1, \infty)$, $[1, \infty)$

H_1 $[3, \infty)$

Computing Persistent Homology

Standard Algorithm: [Edelsbrunner et al. 2002; Zomorodian, Carlsson 2005]

From:



To:

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

H_1 $[3, \infty)$

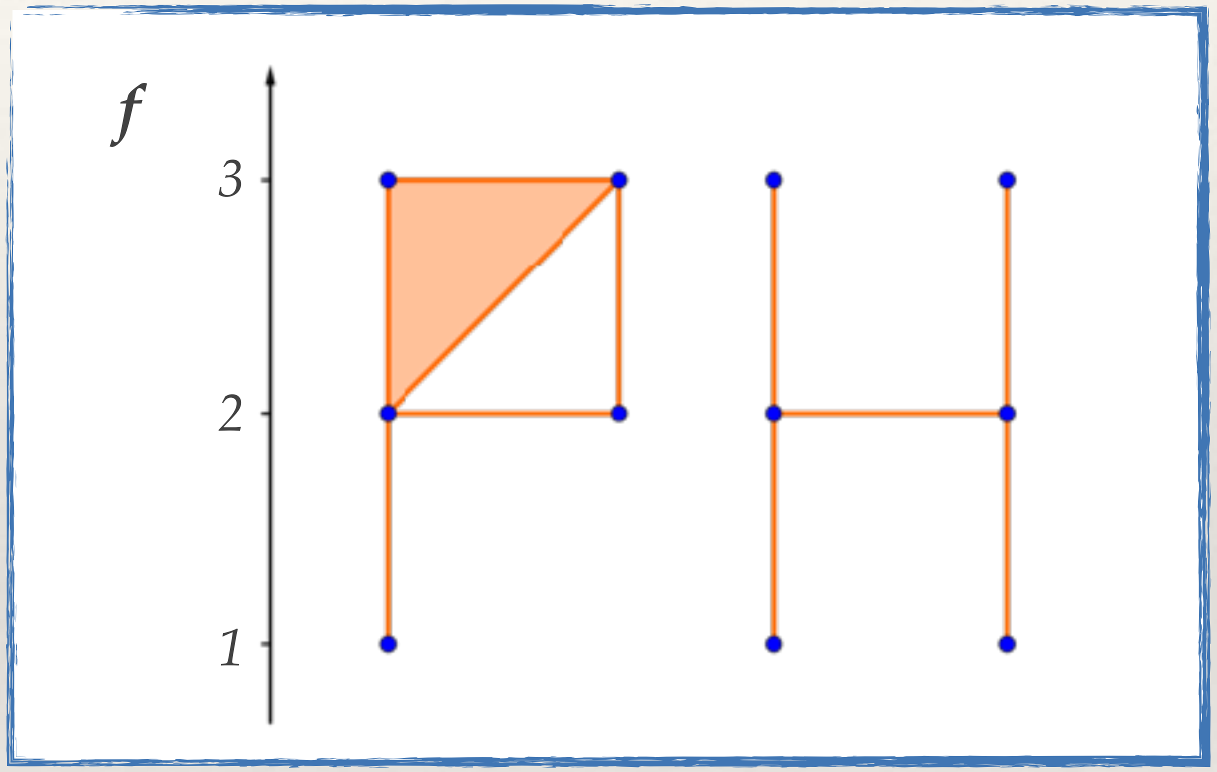
Compute a ***reduced boundary matrix*** for Σ^f
from which easily read the persistence pairs

$i\backslash j$	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

Computing Persistent Homology

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

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



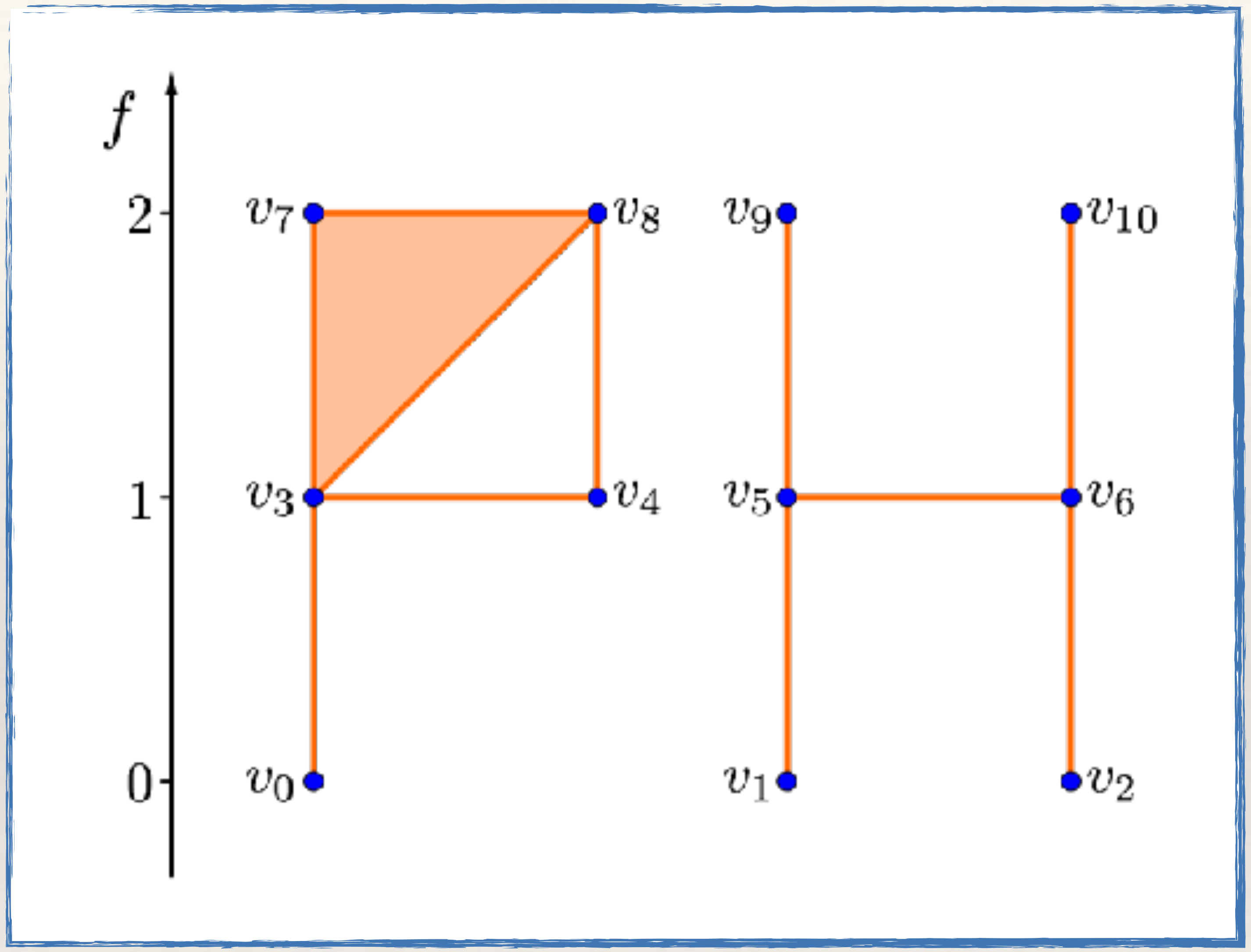
Conversely, $\Sigma^p := \{ \sigma \in \Sigma \mid f(\sigma) \leq p \}$

Total Ordering on Σ^f:

A sequence $\sigma_1, \sigma_2, \dots, \sigma_n$ of the simplices of Σ such that:

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

Computing Persistent Homology



A possible choice:

Set $\sigma < \sigma'$ if:

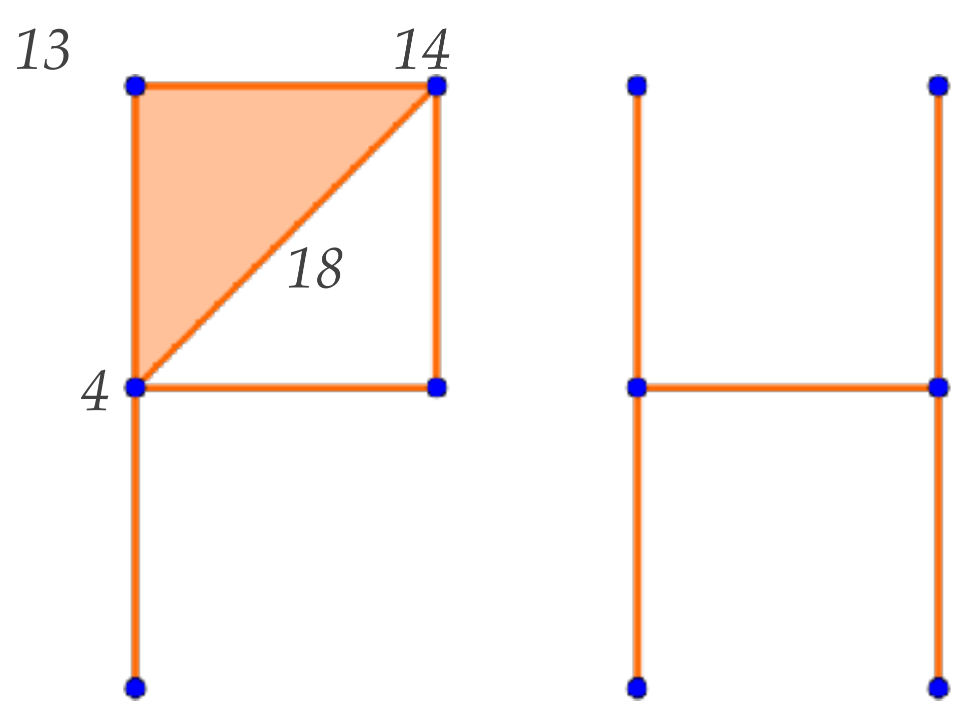
- if $f(\sigma) < f(\sigma')$
- if $f(\sigma) = f(\sigma')$ and $dim(\sigma) < dim(\sigma')$
- if $f(\sigma) = f(\sigma')$ and $dim(\sigma) = dim(\sigma')$ and σ precedes σ' with respect to the *lexicographic order* of their vertices

Computing Persistent Homology

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.,

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

Computing Persistent Homology

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 nun-null columns j_1, j_2,

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

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

Computing Persistent Homology

$i\backslash j$	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

$\boldsymbol{low(10) = 7 = low(12)}$

$\Rightarrow$

M is **not** reduced

Computing Persistent Homology

Reduction Algorithm:

Matrix $R = M$

`for` $j = 1, \ldots, n$ `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

Initialization:

$i\backslash j$	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 Σ^f

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

Step 1:

$j < 12$

$i \backslash j$	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**

Step 2:

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**

Step 2:

										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*

Step 2:

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

$\text{column } 12 := \text{column } 12 + \text{column } 10 \longrightarrow low(12) = 6$

Step 2:

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 $j = 12$, $low(12) = 6$

Step 2:

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*

Step 2:

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

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

Step 2:

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**

Step 3:

$12 < j < 19$

$i \backslash j$	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***

Step 4:

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 $j = 19$, $low(19) = 14$

Step 4:

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

Step 4:

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***

Step 4:

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**

Step 4:

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*

Step 4:

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*** $\longrightarrow$ ***low*(19) *undefined***

Step 4:

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**

Step 5:

$19 < j < 22$

$i \backslash j$	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***

Step 6:

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 $j = 22$, $low(22) = 14$

Step 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			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*

Step 6:

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 $j = 22$, $low(22) = 14$

$\qquad$ column $j' = 18$ is such that $low(j') = low(j) = 14$

So, set

$\text{column } 22 := \text{column } 22 + \text{column } 18 \longrightarrow low(22) = 13$

Step 6:

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 $j = 22$, $low(22) = 13$

Step 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			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 $j = 22$, $low(22) = 13$

$\quad$ column $j' = 17$ is such that $low(j') = low(j) = 13$

So, set

column 22 := column 22 + column 17

Step 6:

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 $j = 22$, $low(22) = 13$

$\quad\quad$ column $j' = 17$ is such that $low(j') = low(j) = 13$

So, set

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

Step 6:

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**

Step 7:

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**

Output:

$i \backslash j$	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***

Computing Persistent Homology

Retrieving Persistence Pairs:

- For each $i = 0, \ldots, n$,

  if there exists j such that $\boldsymbol{low(j) = i}$ $\Rightarrow$ ***[i, j]* is a pair for *R***

- Once every i has been parsed,

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

From pairs of R to the **"actual" persistence pairs of Σ^f:**

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

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

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

Computing Persistent Homology

H_0

[1, ∞)

[2, ∞)

[3, 12]

[4, 8]

[5, 11]

[6, 9]

[7, 10]

[13, 17]

[14, 18]

[15, 20]

[16, 21]

H_1

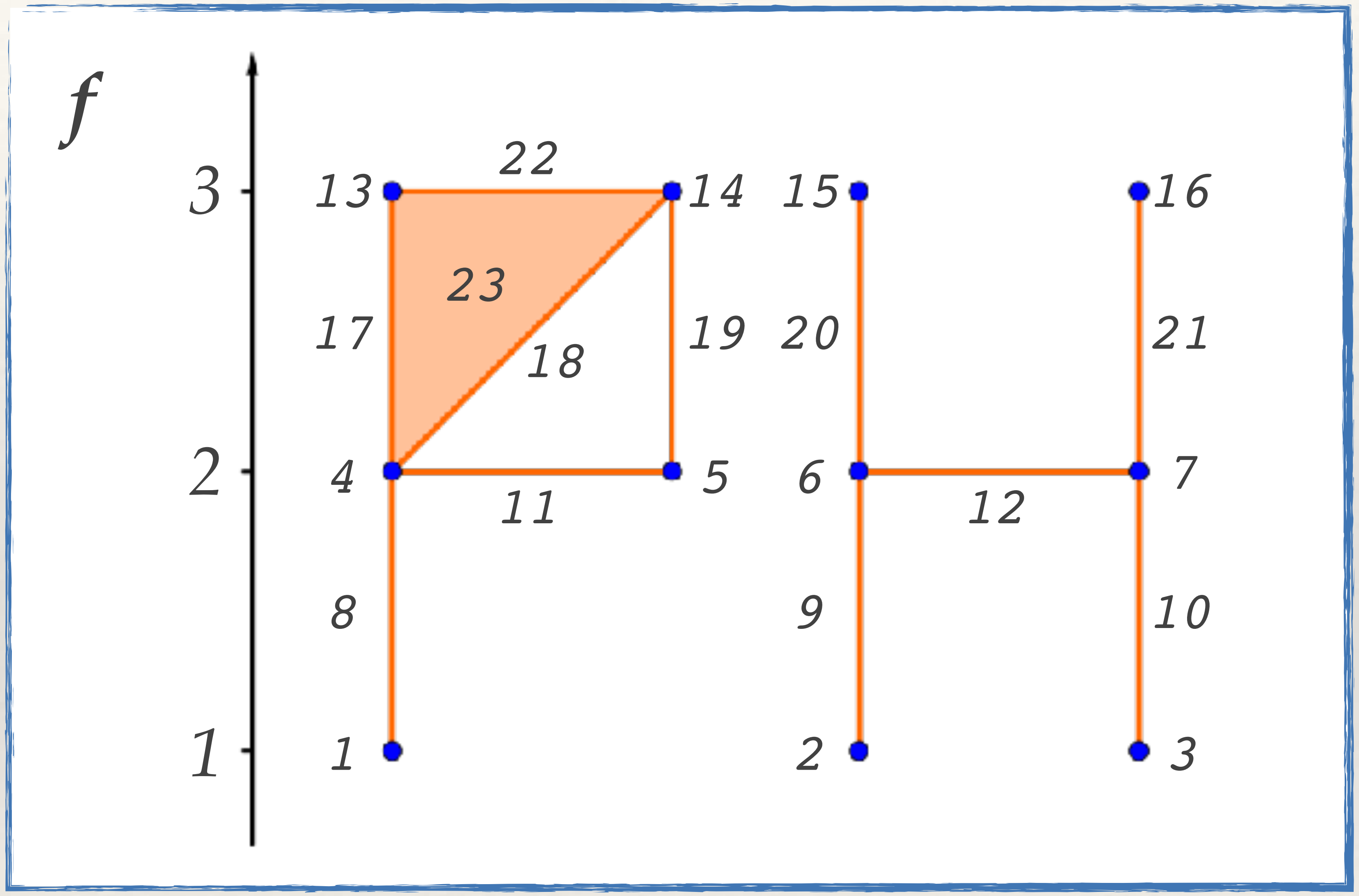
[19, ∞)

[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

Computing Persistent Homology

H_0		
[1, ∞)		[1, ∞)
[2, ∞)		[1, ∞)
[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, ∞)	⇒	[3, ∞)
[22, 23]		[3, 3]

Computing Persistent Homology

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

- Based on a **matrix reduction**
- **Linear complexity** in practical cases
- **Quadratic 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; Dlotko, Wagner '14]
- ***Morse-based approaches*** [Robins et al. '11; Harker et al. '14; Fugacci et al. '14]

Computing Persistent Homology

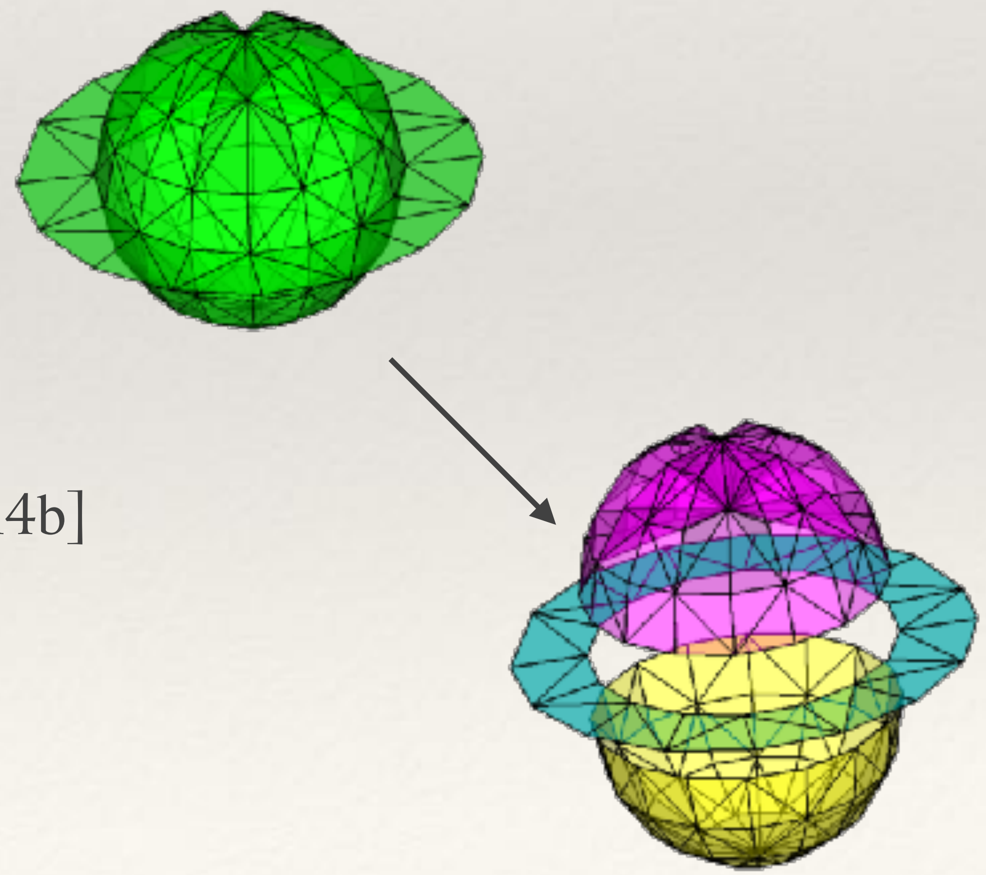
Direct approaches:

- ***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]

Computing Persistent Homology

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]



Computing Persistent Homology

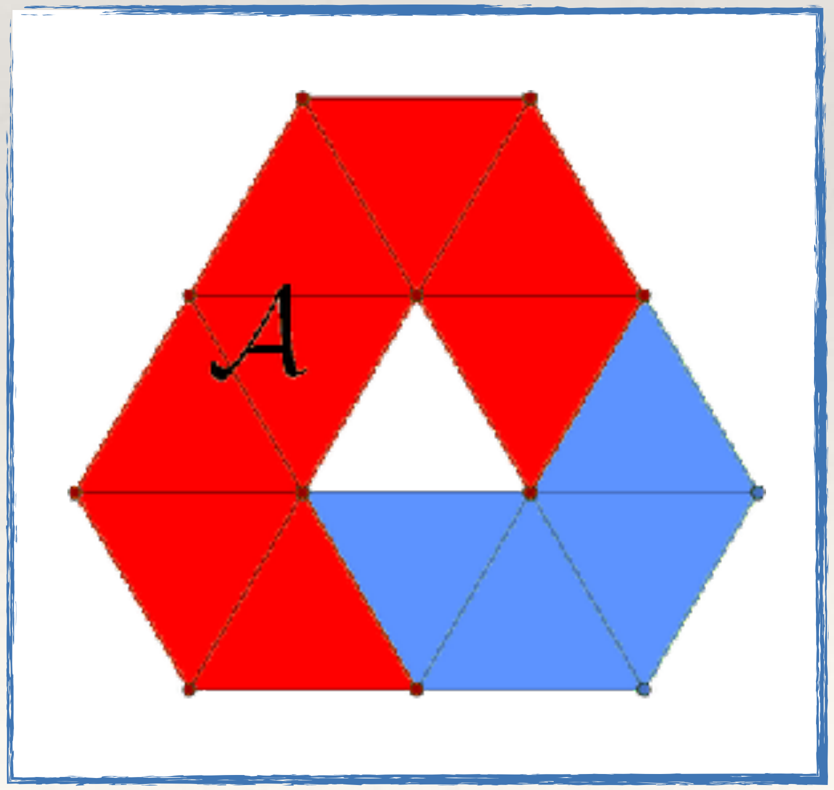
Coarsening approaches:

- *Topological operators and simplifications* [Dlotko, Wagner '14]

Computing Persistent Homology

Coarsening approaches:

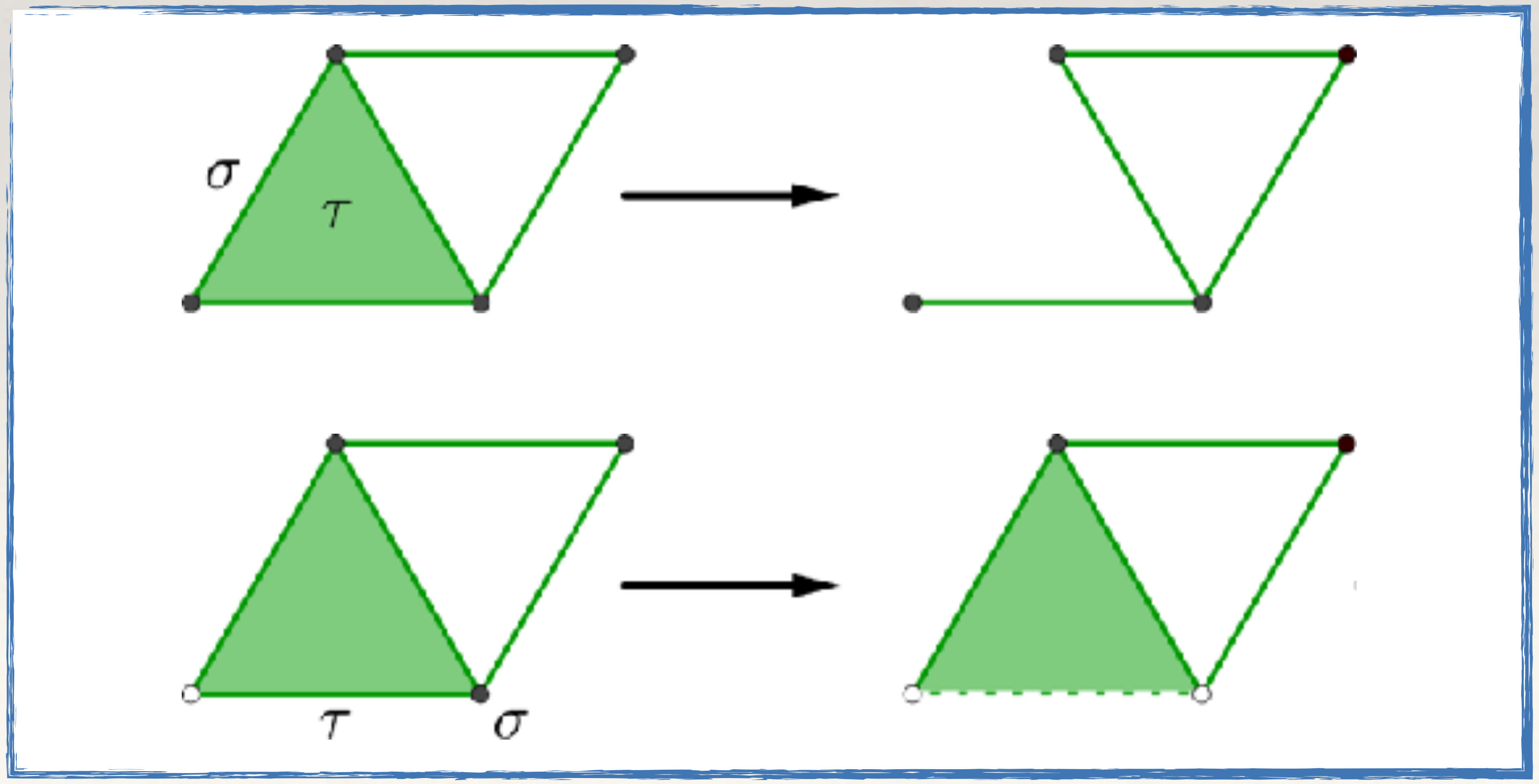
- *Topological operators and simplifications* [Dlotko, Wagner '14]
  - **Acyclic Subcomplexes** [Mrozek et al. '08]



Computing Persistent Homology

Coarsening approaches:

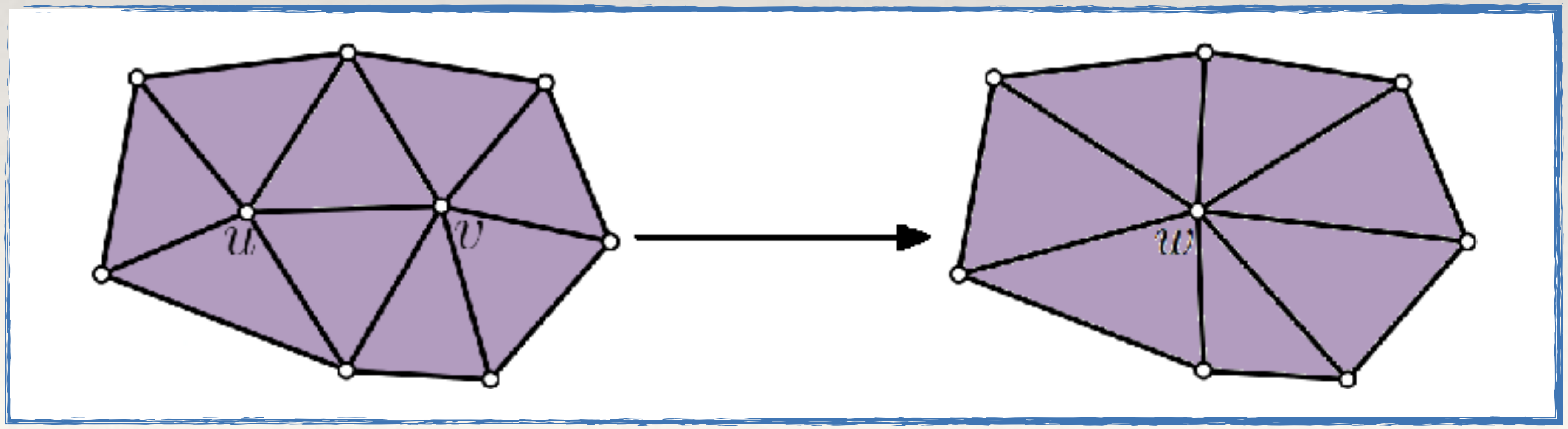
- *Topological operators and simplifications* [Dlotko, Wagner '14]
  - Acyclic Subcomplexes [Mrozek et al. '08]
  - **Reductions and Coreductions** [Mrozek et al. '10]



Computing Persistent Homology

Coarsening approaches:

- *Topological operators and simplifications* [Dlotko, Wagner '14]
  - Acyclic Subcomplexes [Mrozek et al. '08]
  - Reductions and Coreductions [Mrozek et al. '10]
  - **Edge Contractions** [Attali et al. '11]



Computing Persistent Homology

Coarsening approaches:

- ***Topological operators and simplifications*** [Dlotko, 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]

Outline

Persistent Homology Computation

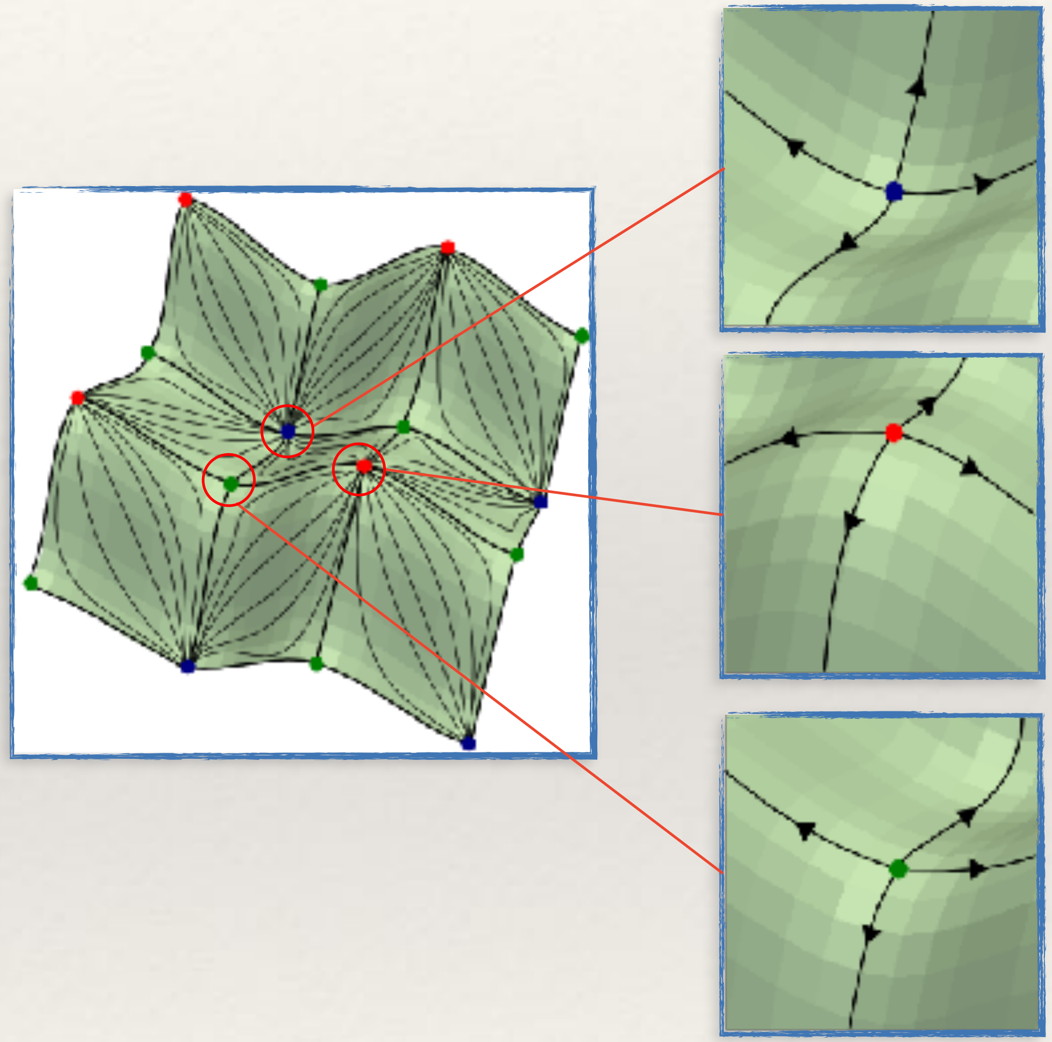
Discrete Morse Theory and Persistent Homology

Morse Theory [Milnor 1963, Matsumoto 2002]

- Topological tool for efficiently analyzing a **shape** by studying the behavior of a smooth **scalar function** *f* defined on it

- Relates the critical points of a smooth scalar function on a shape with their **regions of influence**

- Analysis of scalar fields requires extracting **morphological features** (e.g., critical points, integral lines and surfaces)

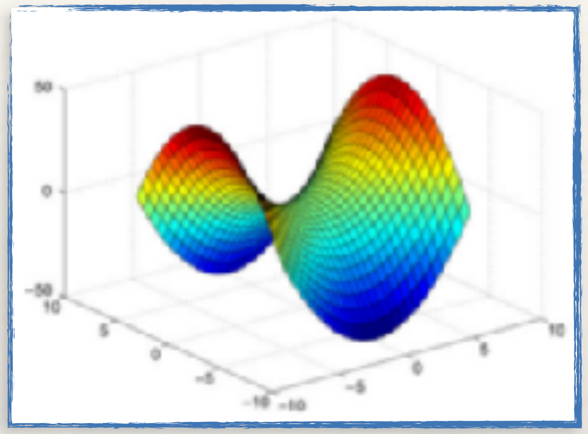


Morse Theory

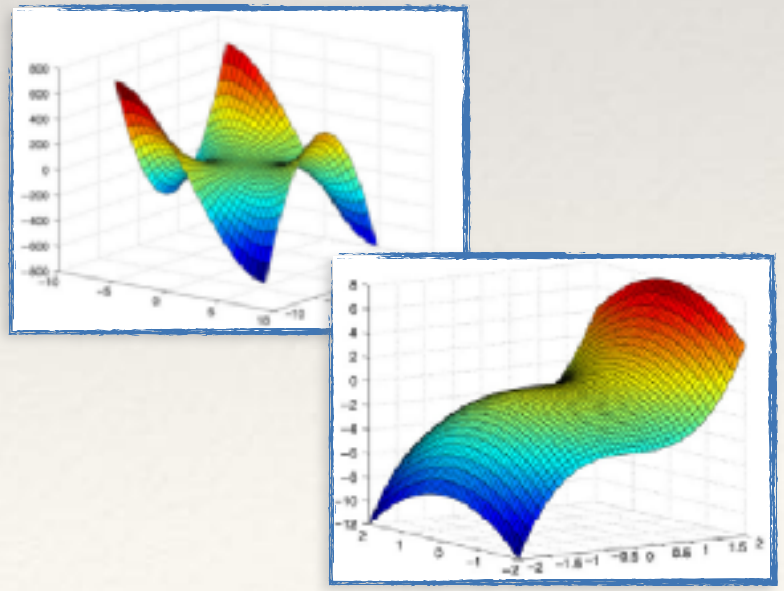
Let f be a real-valued C^2-function defined on a d-dimensional manifold M

- **Critical point** of f: any point on M in which the gradient of f vanishes
- Critical points can be **degenerate** or **non-degenerate**
  - A critical point p is degenerate iff the determinant the Hessian matrix H of the second order derivatives of function f is null

Non-degenerate critical point



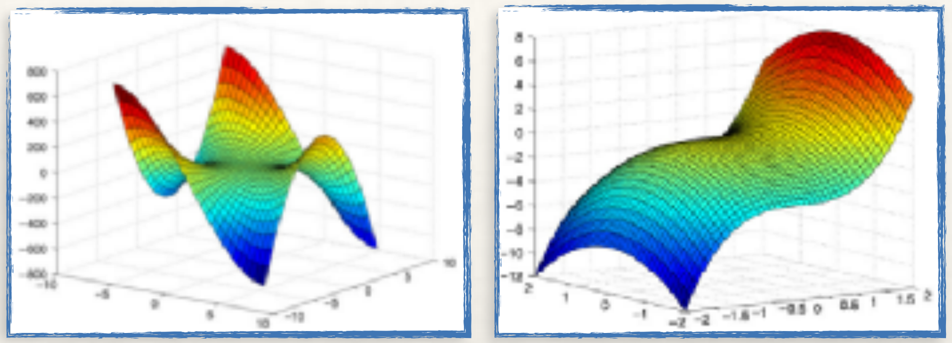
Degenerate critical points (monkey saddle and flat saddle)



Function f is a **Morse function** if and only if ***all its critical points are non-degenerate***

Morse Theory

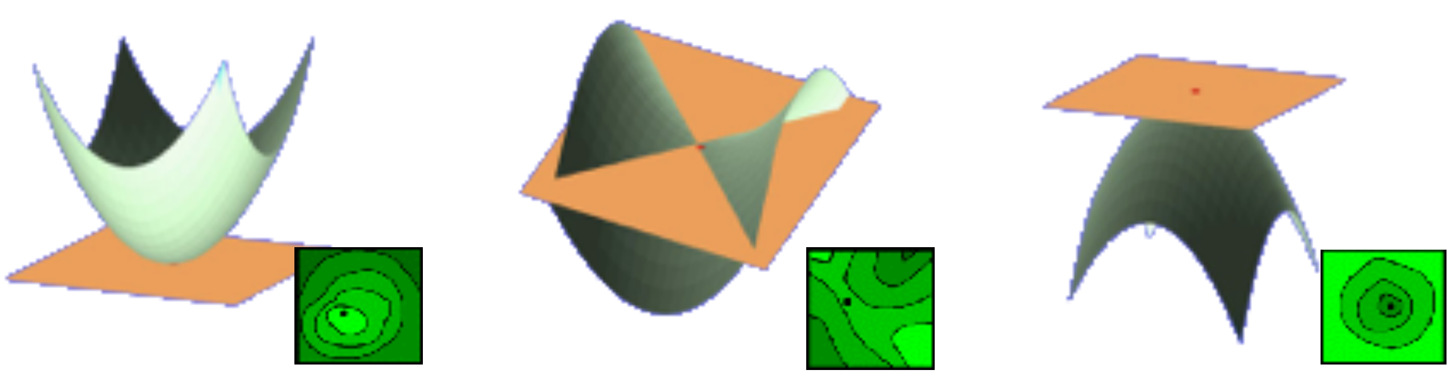
Critical points of a Morse function are ***isolated***



Examples of **non-Morse** functions

- A d-dimensional Morse function f has $d+1$ types of critical points, called ***i*-saddles** (i is the **index** of the critical point)
  - For $d = 2$: ***minima***, ***saddles*** and ***maxima***
  - For $d = 3$: ***minima***, ***1-saddles***, ***2- saddles*** and ***maxima***

Critical points of a 2D function

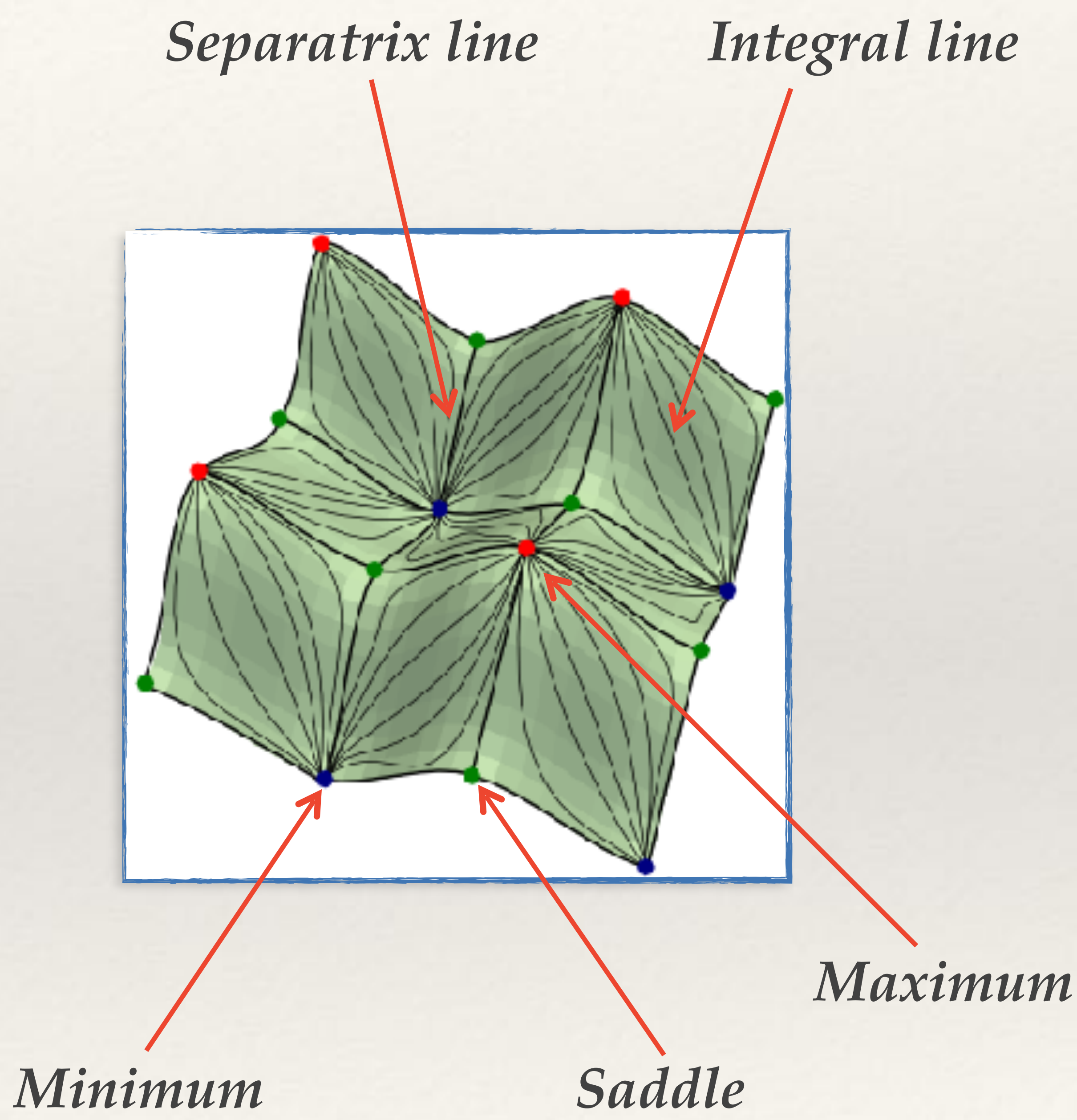


Morse Theory

- An **integral line** of a smooth function f is a maximal path which is everywhere tangent to the gradient vector field of f

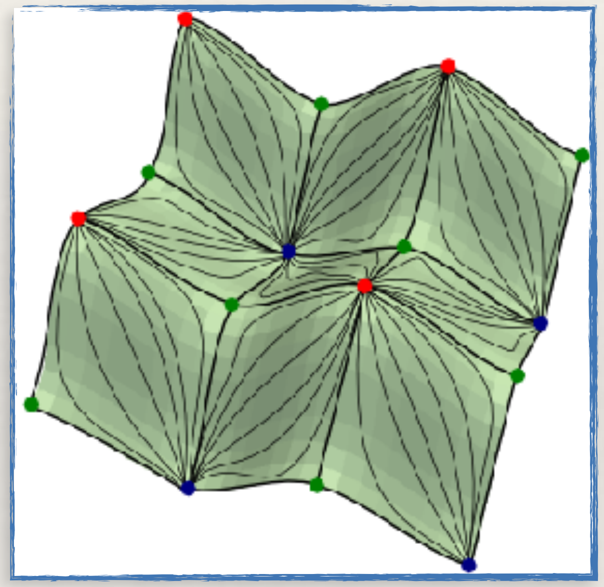
- Integral lines **start** and **end** at the critical points of f

- Integral lines that connect critical points of consecutive index are called **separatrix lines**



Morse Theory

- Integral lines that converge to a critical point p of index i form an i-cell called the **descending cell** of p
  - Descending cell of a maximum: 2-cell
  - Descending cell of a saddle: 1-cell
  - Descending cell of a minimum: 0-cell

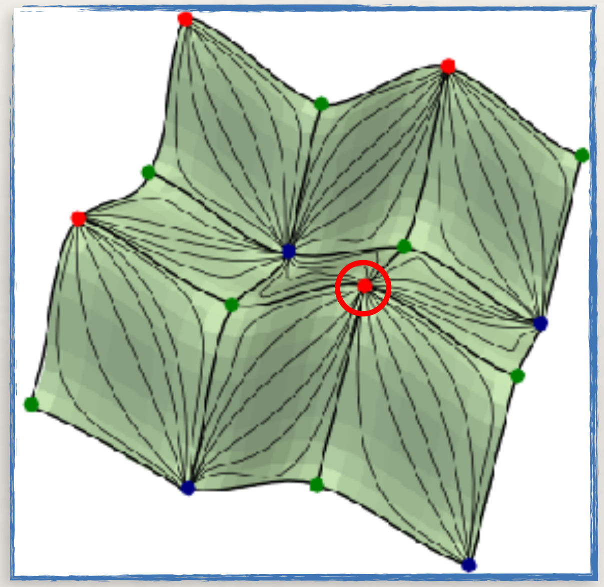


Descending Morse Complex:

Collection of the descending cells of all critical points of function f

Morse Theory

- Integral lines that converge to a critical point p of index i form an i-cell called the **descending cell** of p
  - Descending cell of a maximum: 2-cell
  - Descending cell of a saddle: 1-cell
  - Descending cell of a minimum: 0-cell

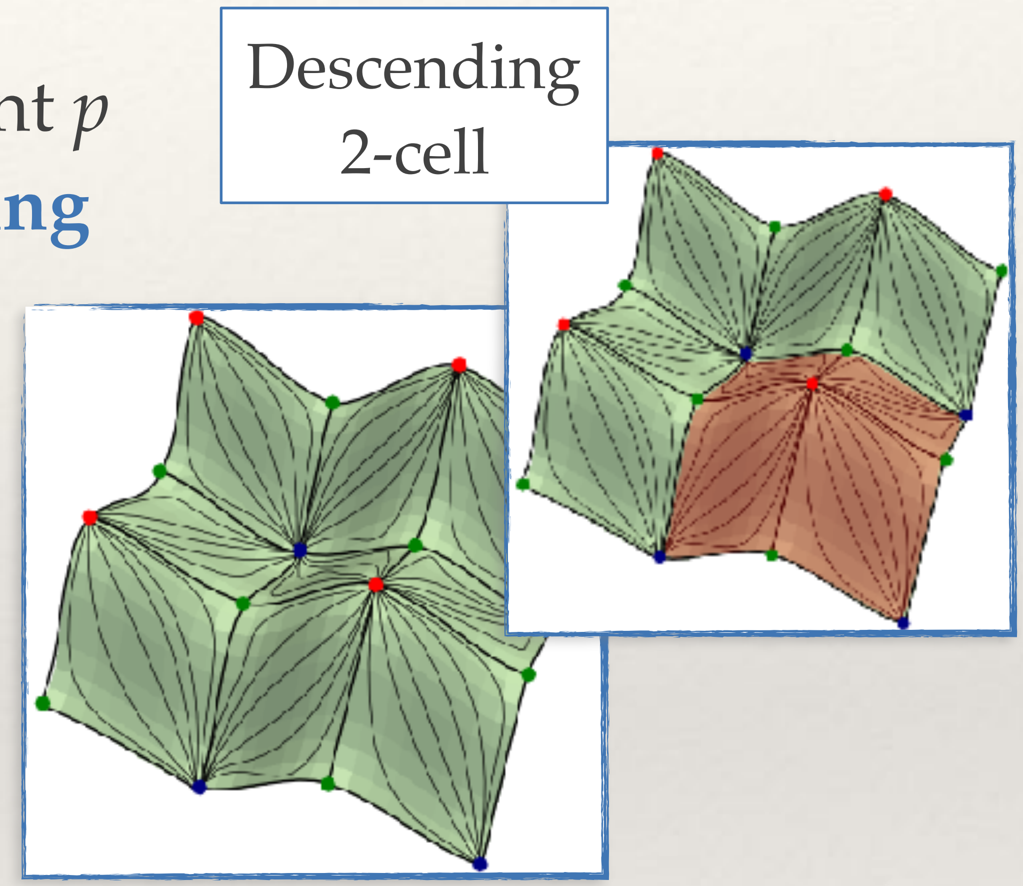


Descending Morse Complex:

Collection of the descending cells of all critical points of function f

Morse Theory

- Integral lines that converge to a critical point p of index i form an i-cell called the **descending cell** of p
  - Descending cell of a maximum: 2-cell
  - Descending cell of a saddle: 1-cell
  - Descending cell of a minimum: 0-cell



Descending Morse Complex:

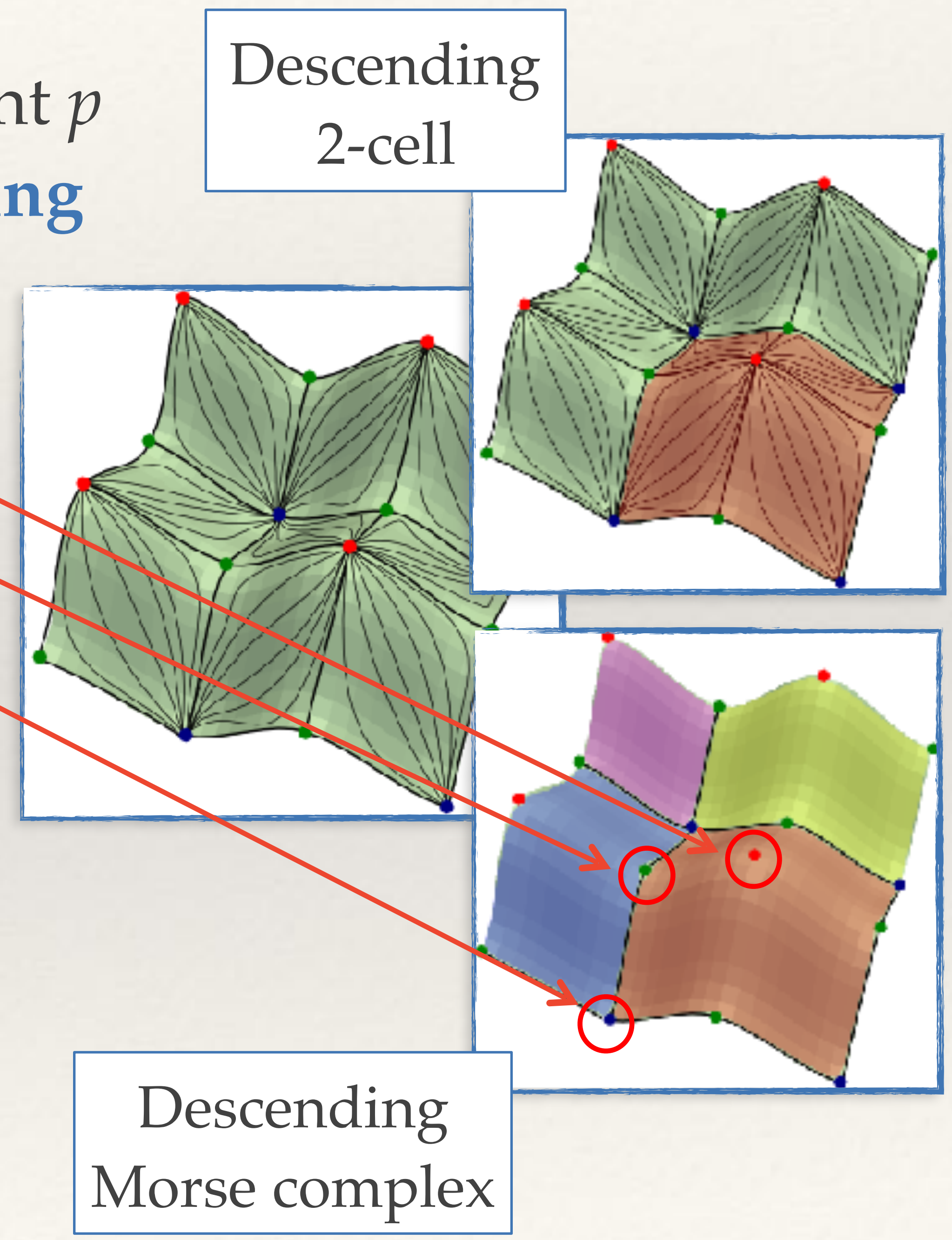
Collection of the descending cells of all critical points of function f

Morse Theory

- Integral lines that converge to a critical point p of index i form an i-cell called the **descending cell** of p
  - Descending cell of a maximum: 2-cell
  - Descending cell of a saddle: 1-cell
  - Descending cell of a minimum: 0-cell

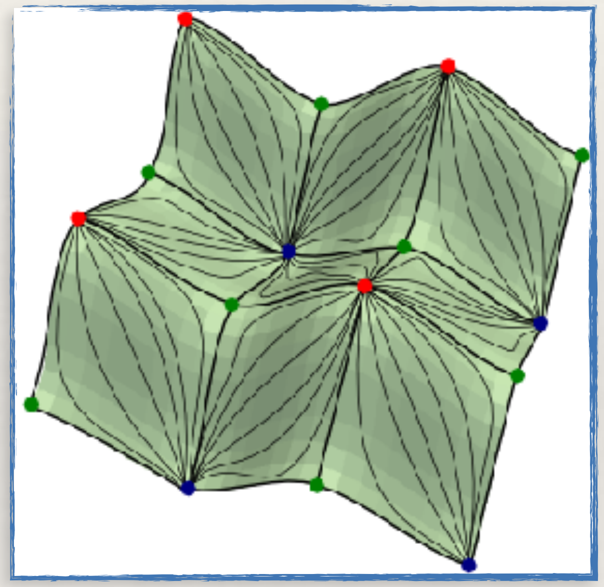
Descending Morse Complex:

Collection of the descending cells of all critical points of function f



Morse Theory

- Integral lines that originate at a critical point p of index i form a $(d\text{-}i)$-cell called the **ascending cell** of p
  - Ascending cell of a minimum: 2-cell
  - Ascending cell of a saddle: 1-cell
  - Ascending cell of a maximum: 0-cell

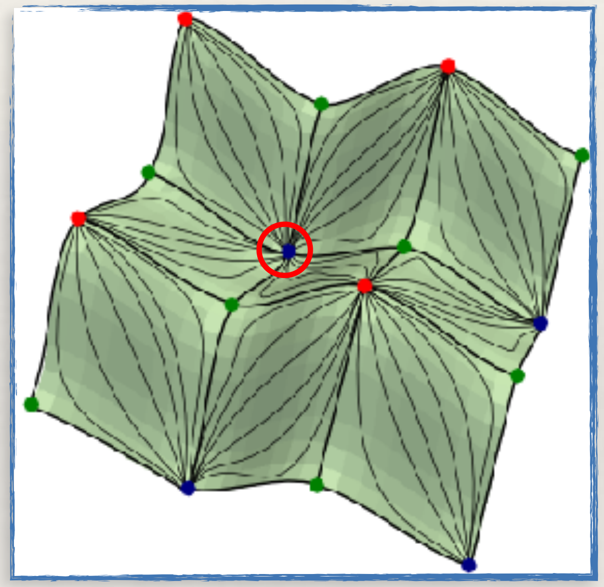


Ascending Morse Complex:

Collection of the ascending cells of all critical points of function f

Morse Theory

- Integral lines that originate at a critical point p of index i form a $(d\text{-}i)$-cell called the **ascending cell** of p
  - Ascending cell of a minimum: 2-cell
  - Ascending cell of a saddle: 1-cell
  - Ascending cell of a maximum: 0-cell



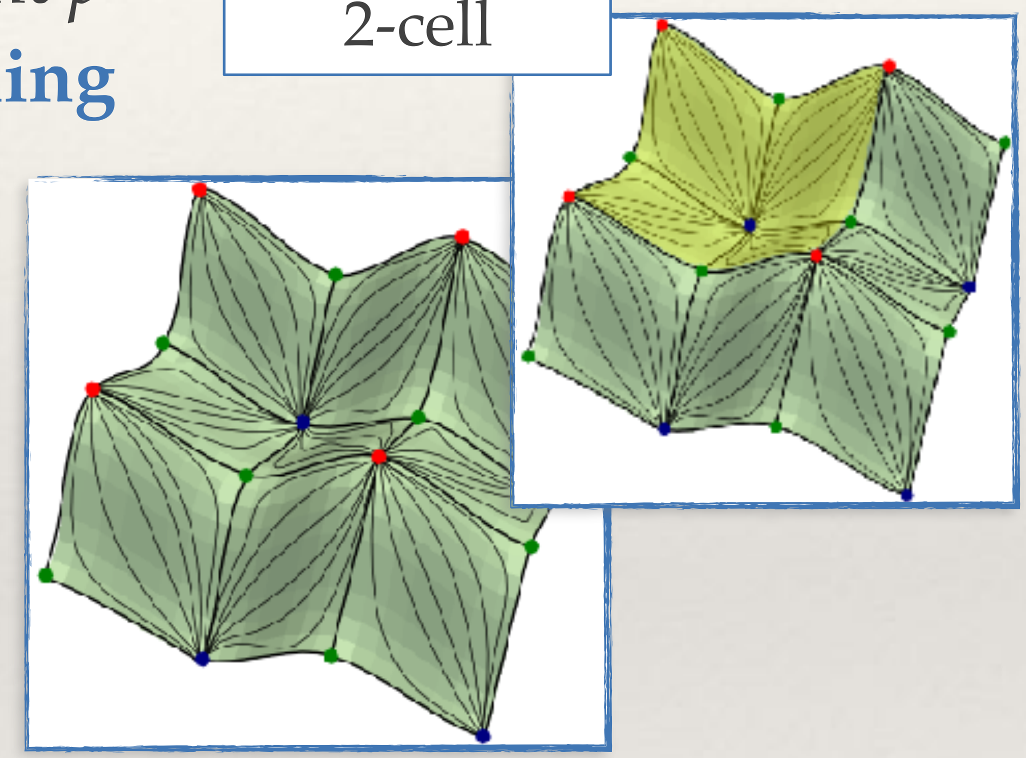
Ascending Morse Complex:

Collection of the ascending cells of all critical points of function f

Morse Theory

- Integral lines that originate at a critical point p of index i form a $(d\text{-}i)$-cell called the **ascending cell** of p
    - Ascending cell of a minimum: 2-cell
    - Ascending cell of a saddle: 1-cell
    - Ascending cell of a maximum: 0-cell

Ascending 2-cell



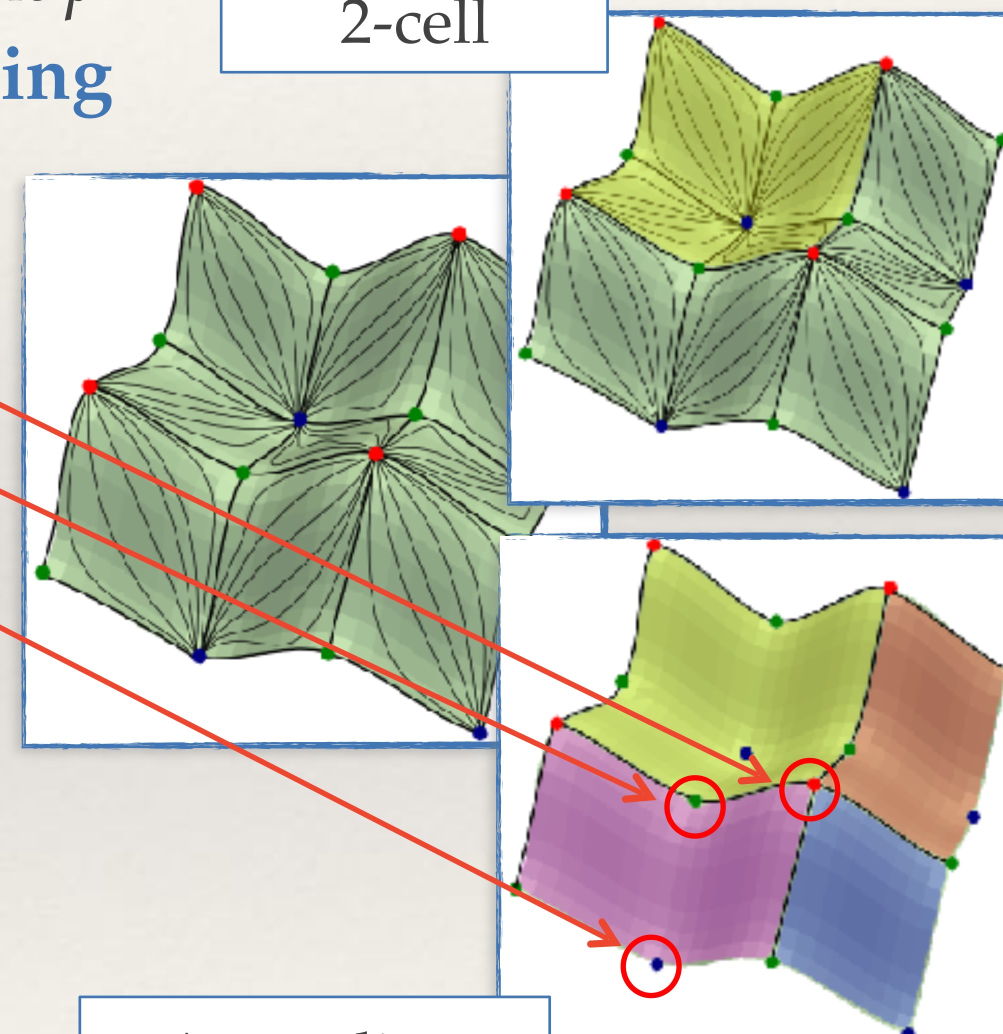
Ascending Morse Complex:

Collection of the ascending cells of all critical points of function f

Morse Theory

- Integral lines that originate at a critical point p of index i form a $(d\text{-}i)$-cell called the **ascending cell** of p
  - Ascending cell of a minimum: 2-cell
  - Ascending cell of a saddle: 1-cell
  - Ascending cell of a maximum: 0-cell

Ascending 2-cell



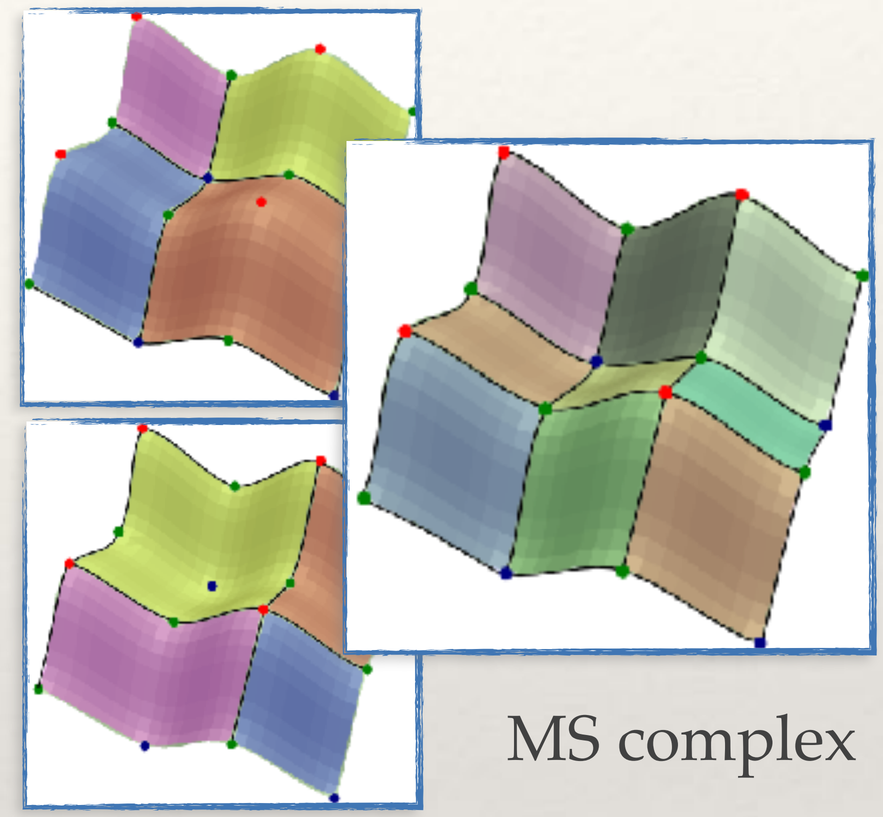
Ascending Morse complex

Ascending Morse Complex:

Collection of the ascending cells of all critical points of function f

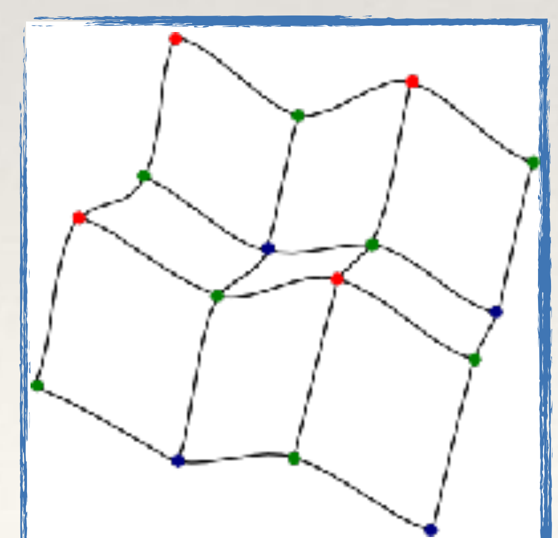
Morse Theory

- Function f is a **Morse-Smale** function if its ascending and descending Morse cells intersect transversally



MS complex

- **Morse-Smale (MS) complex** is the complex obtained from the mutual intersection of all the ascending and descending cells

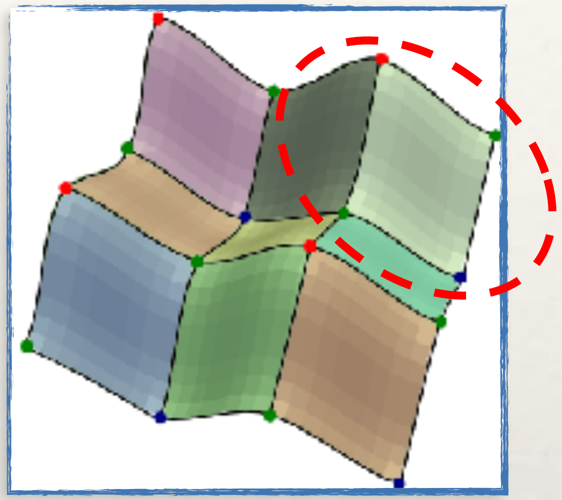


MS
1-skeleton

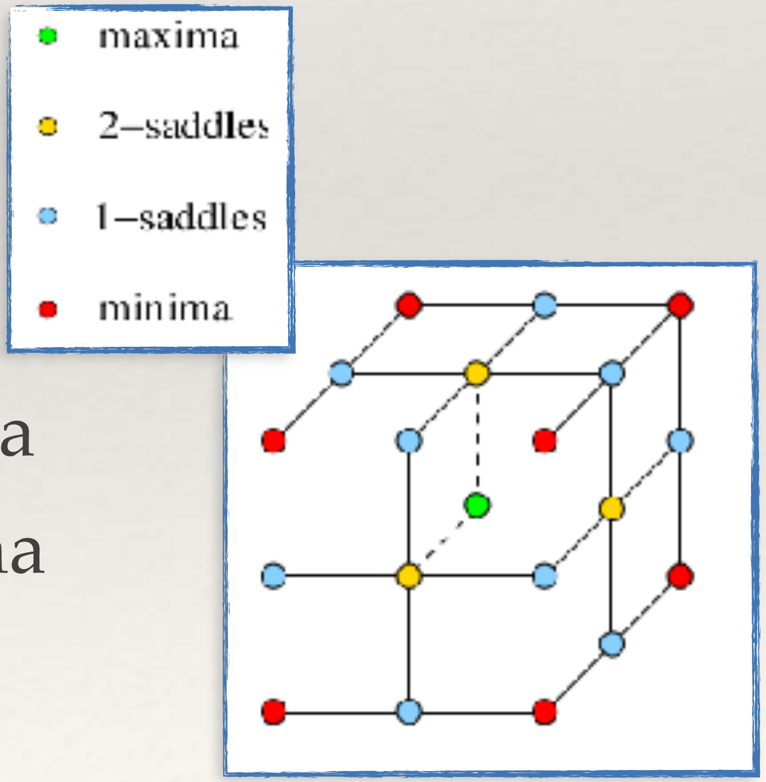
Morse Theory

- In a 2D Morse-Smale complex:
  - a 2-cell is a **quadrilateral** bounded by the sequence

    maximum – saddle – minimum – saddle



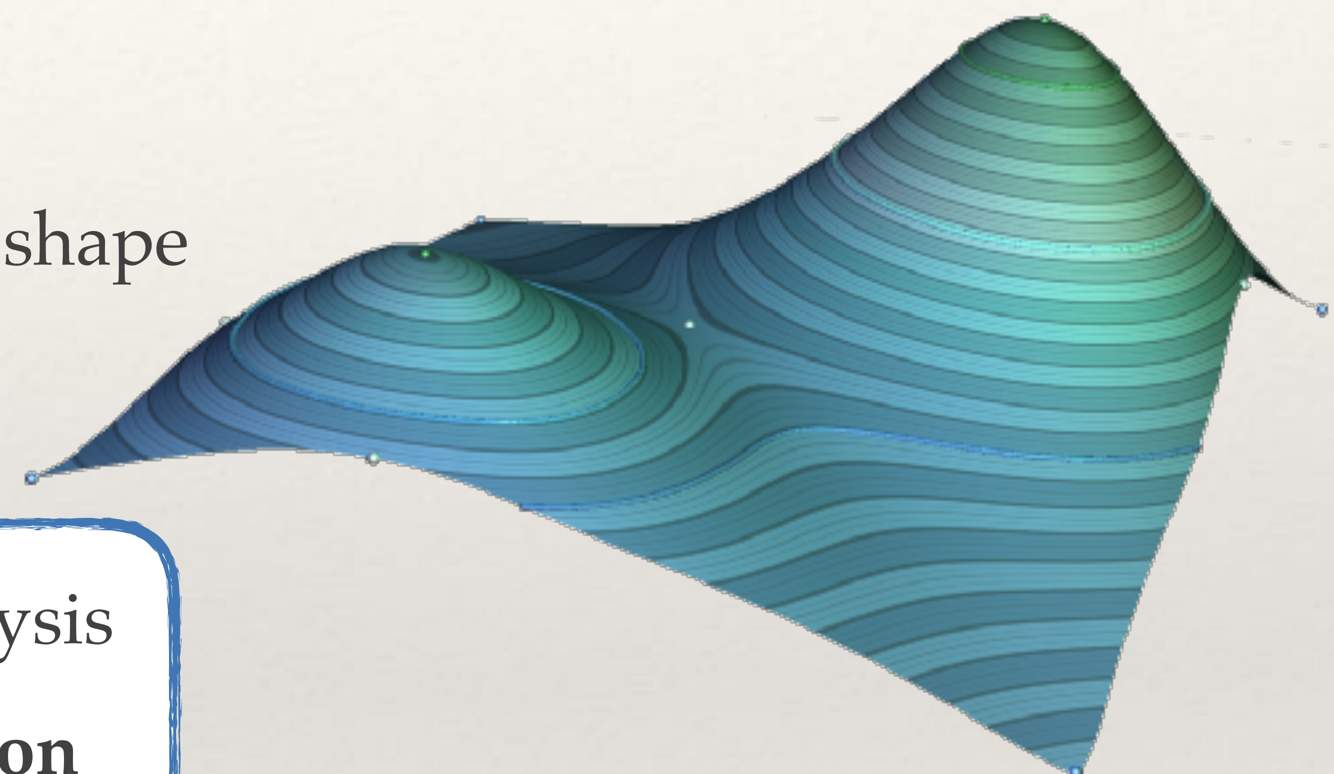
- In a 3D Morse-Smale complex:
  - each 1-saddle is connected to exactly two minima
  - each 2-saddle is connected to exactly two maxima



Discrete Morse Theory

Morse Theory:

Enables to **analyze** the **topology** of a shape by **studying functions** defined on it



Useful for
- **homological** analysis
- shape **segmentation**

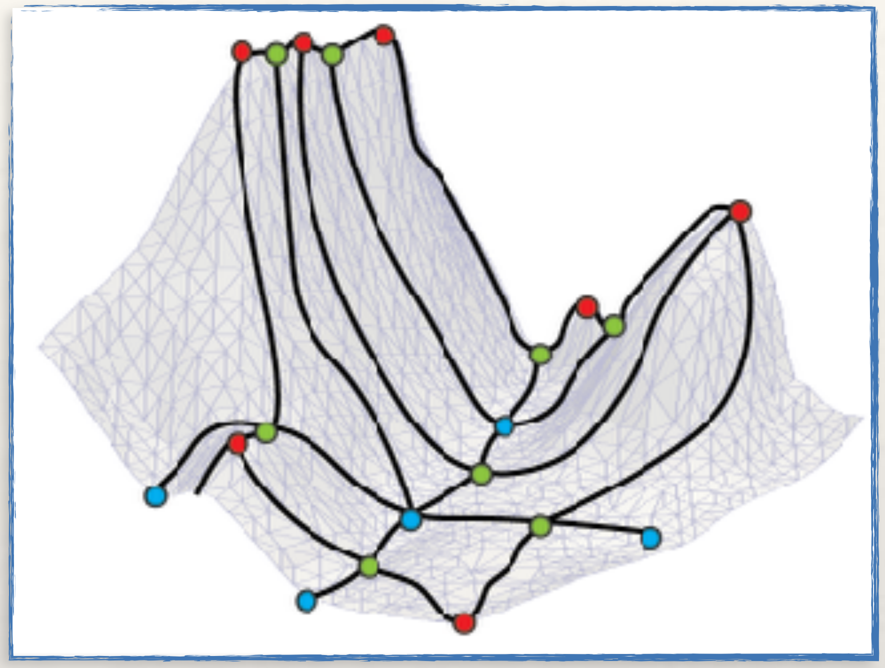
Various **discretizations** of Morse theory:

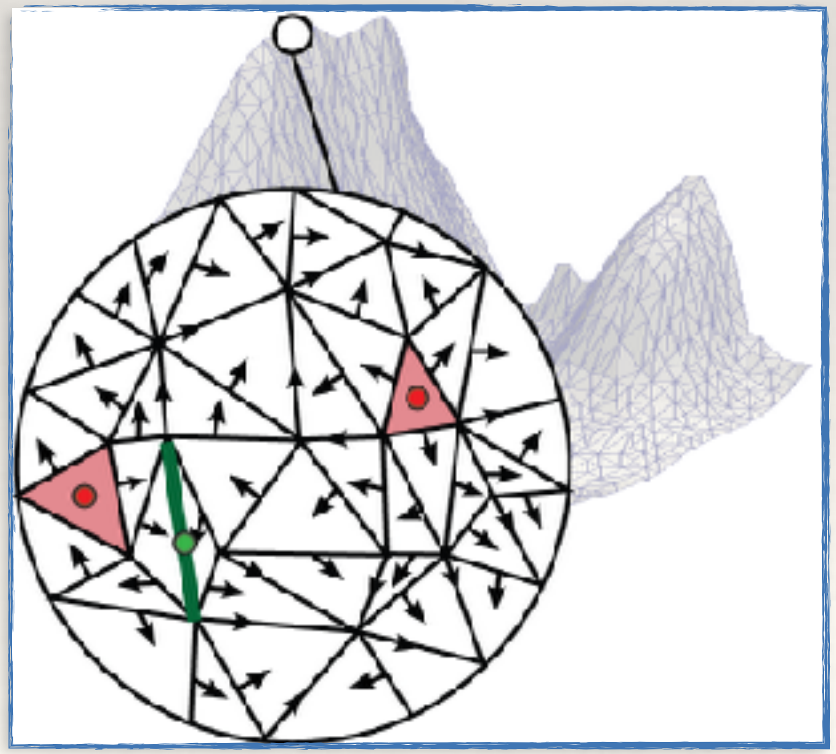
- *Piecewise linear Morse theory* [Banchoff '67]
- *Watershed transform* [Meyer '94]
- ***Discrete Morse theory*** [Forman '98]

Discrete Morse Theory

Combinatorial counterpart of Morse theory:

- Introduced for **cell complexes**
- Gives a compact **homology-equivalent** model for a shape
- **Derivative free** tool for computing segmentations of shapes





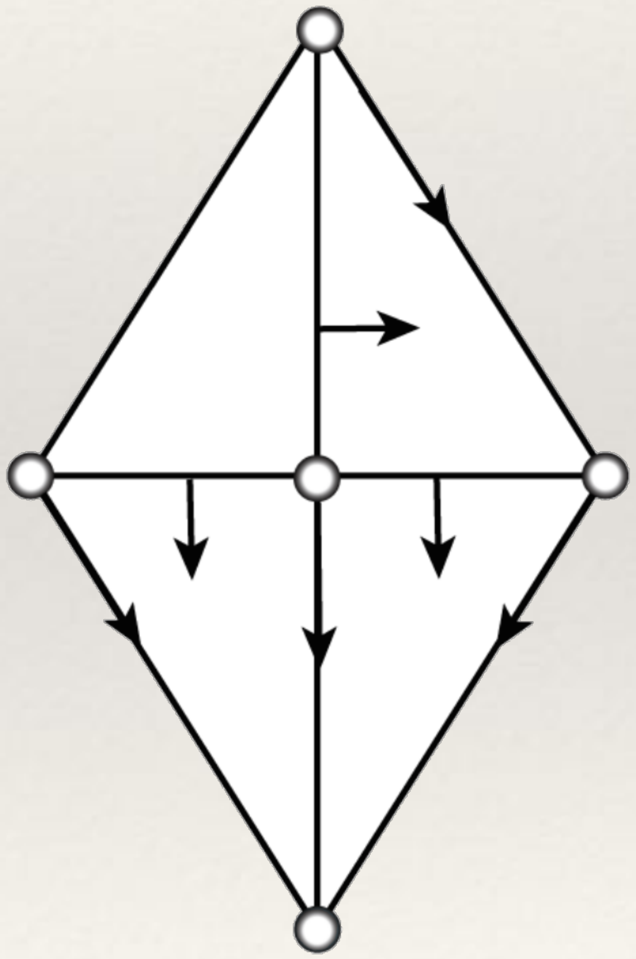
Discrete Morse Theory

Discrete Morse Theory:

Gradient of a function is **simulated by a matching** V of the simplices in Σ

A **matching** V is a **collection of pairs** (σ,τ) such that:

- σ, τ are **incident** simplices of dimension k and $k+1$
- each simplex of Σ is **in at most one pair** of V



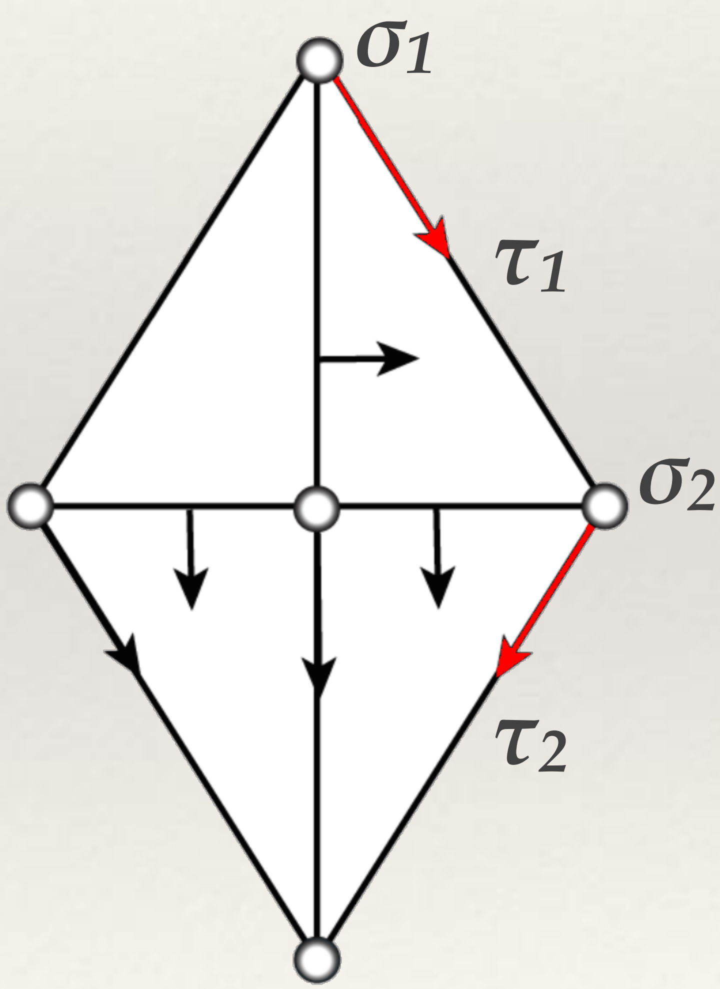
Discrete Morse Theory

Discrete Morse Theory:

Gradient of a function is **simulated by a matching** V of the simplices in Σ

A **matching** V is a **collection of pairs** (σ,τ) such that:

- σ, τ are **incident** simplices of dimension k and $k+1$
- each simplex of Σ is **in at most one pair** of V



V-path: Sequence of pairs of V

$$(\sigma_1,\tau_1), (\sigma_2,\tau_2), \ldots, (\sigma_{r-1},\tau_{r-1}), (\sigma_r,\tau_r)$$

such that

- σ_{i+1} is a k-simplex face of the $(k+1)$-simplex τ_i
- σ_{i+1} is different from σ_i

A matching V is called **Forman gradient** if it is **free of closed V-paths**

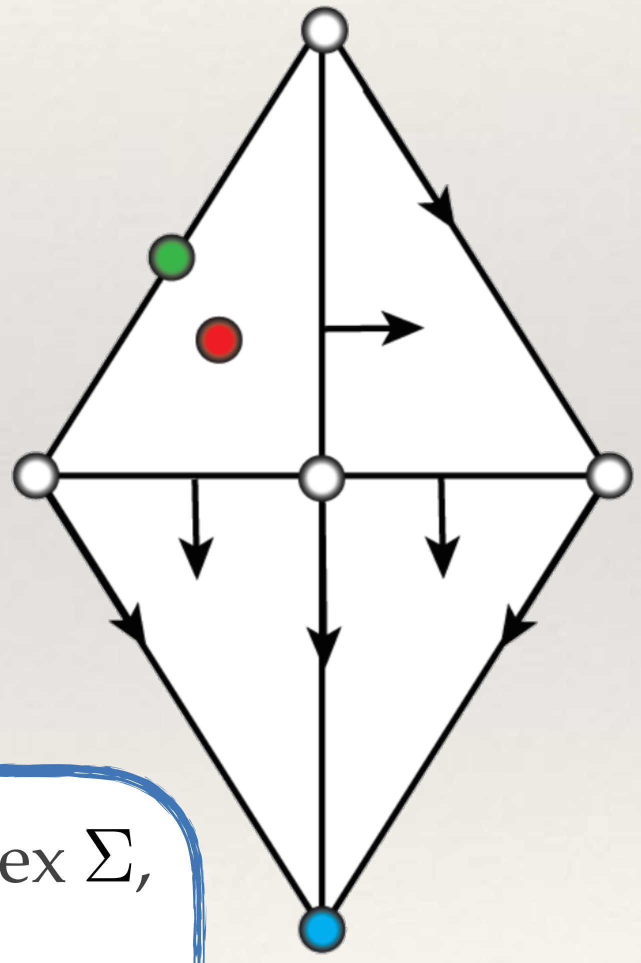
Discrete Morse Theory

Unpaired simplices of dimension k are denoted as
critical simplices of index k

Discrete Morse Complex:

A chain complex whose:

- k-cells $\longleftrightarrow$ critical simplices of index k
- boundary relations are induced by V-paths



Theorem.

Given a Forman gradient V defined on a simplicial complex Σ,
the associated **discrete Morse complex is**
homologically equivalent to Σ

Discrete Morse Theory

Filtered Forman Gradient:

Given a filtration F of a simplicial complex Σ,
a Forman gradient V is a ***filtered Forman gradient of F*** if, for each pair $(\sigma,\tau) \in V$,
there exists p such that $\sigma,\tau \in \Sigma^p$ and $\sigma,\tau \notin \Sigma^{p-1}$

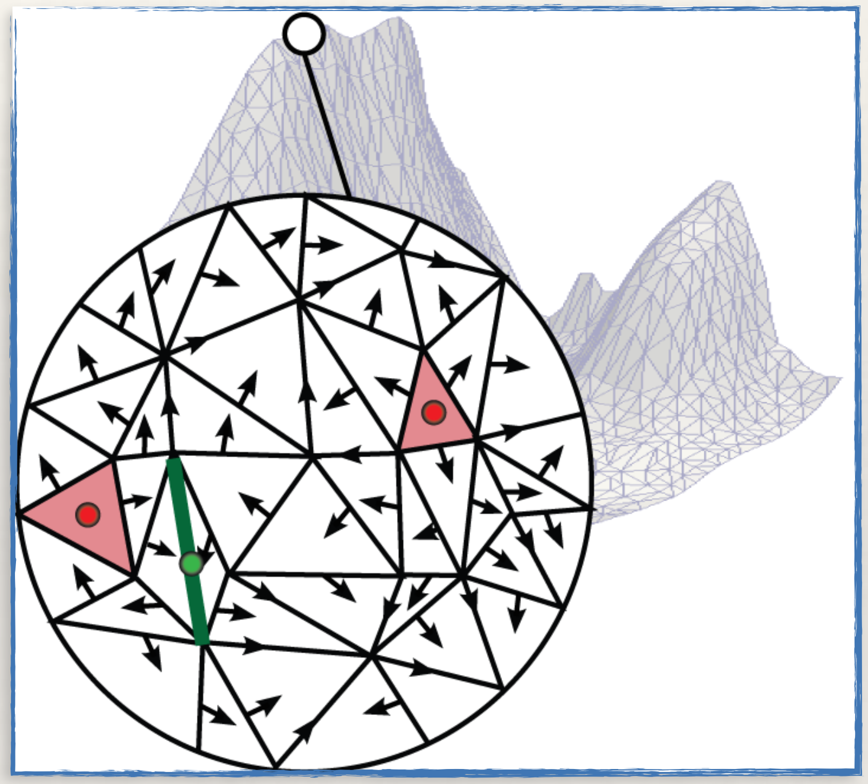
Filtration F ***naturally induces*** a filtration on the discrete Morse Complex

Theorem.

If V is a filtered Forman gradient of F, then Σ and the associated discrete Morse complex have **isomorphic persistent homology**

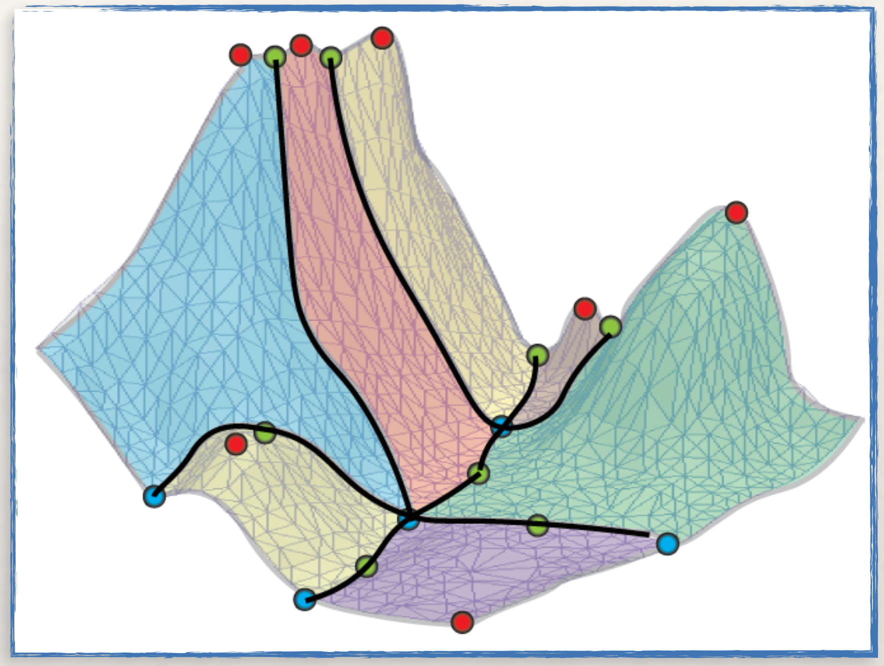
Discrete Morse Theory

Let Σ be a simplicial complex of dimension d



Discrete Morse Theory

Let Σ be a simplicial complex of dimension d

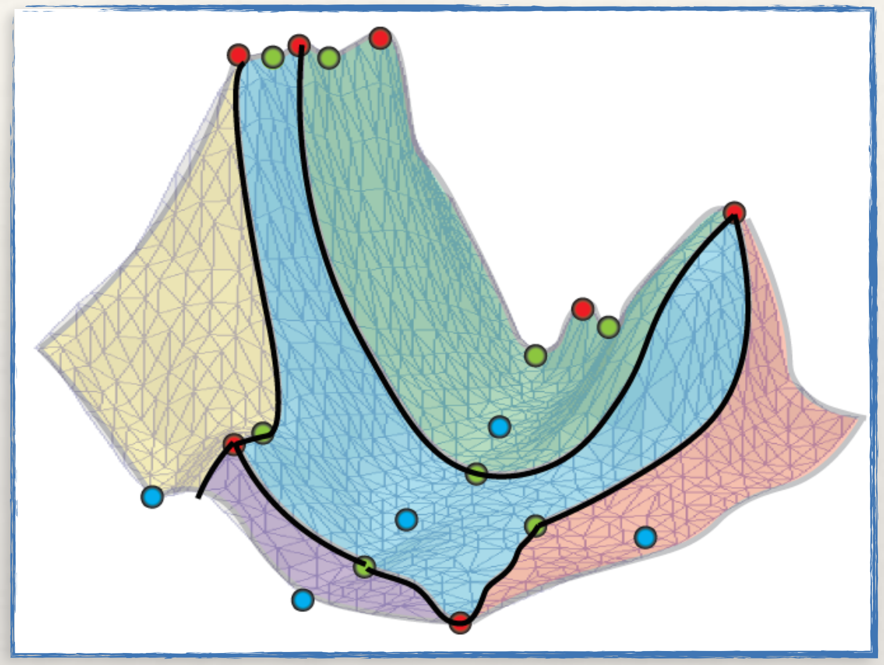


Navigating the V-paths, one can retrieve:

- **Descending Morse complex Γ_D**
  - generated by collection of the d-cells representing the regions of influence of the ***maxima*** of f: **k-cells of $\Gamma_D \longleftrightarrow$ critical simplices of index k**

Discrete Morse Theory

Let Σ be a simplicial complex of dimension d

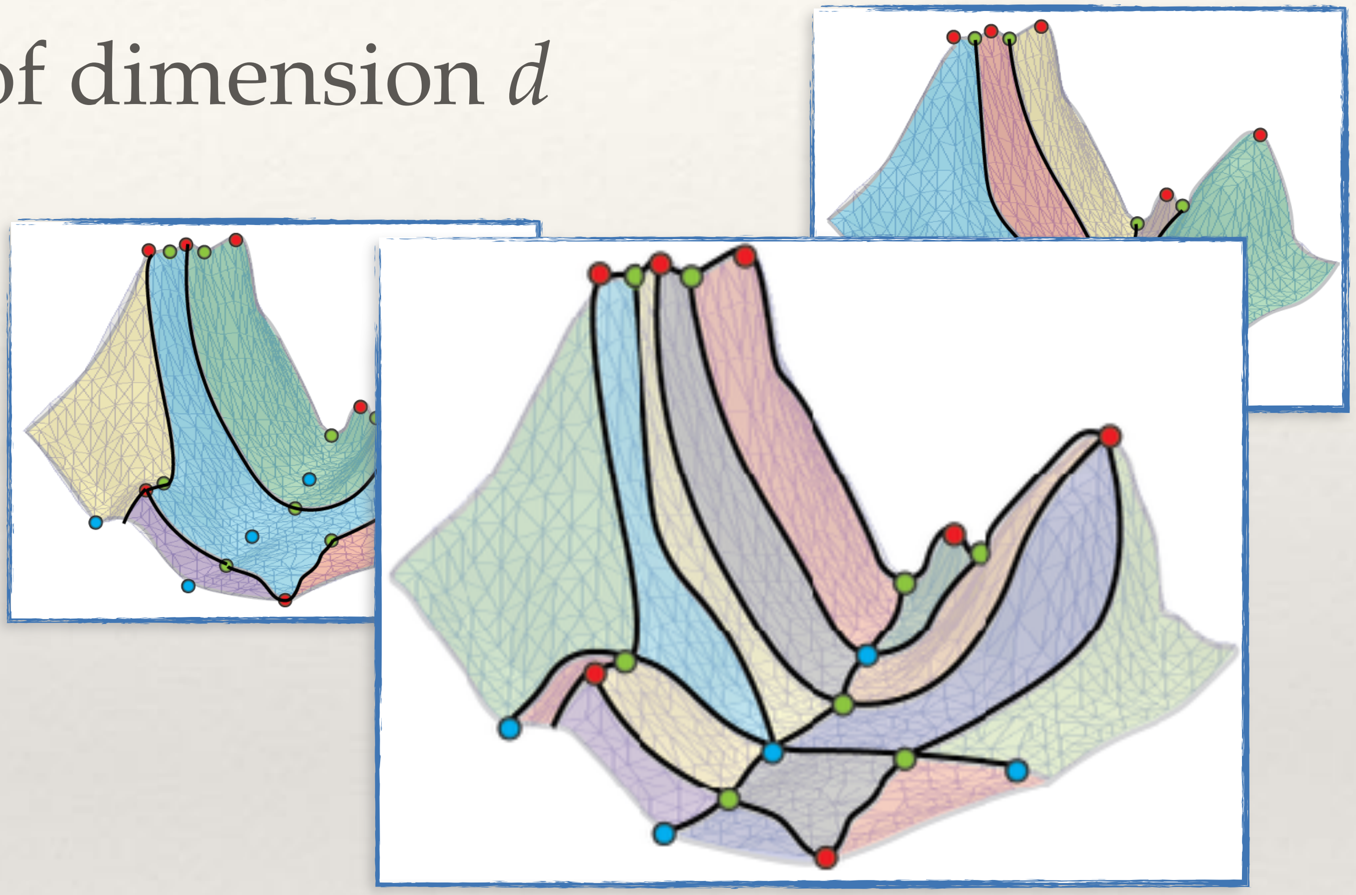


Navigating the V-paths, one can retrieve:

- **Ascending Morse complex Γ_A**
  - generated by collection of the d-cells representing the regions of influence of the ***minima*** of f : **(d-k)-cells of $\Gamma_A \longleftrightarrow$ critical simplices of index k**

Discrete Morse Theory

Let Σ be a simplicial complex of dimension d



Navigating the V-paths, one can retrieve:

- **Morse-Smale complex Γ_{MS}**
    - generated by the connected components of the ***intersection*** of the cells of the descending and ascending Morse complexes

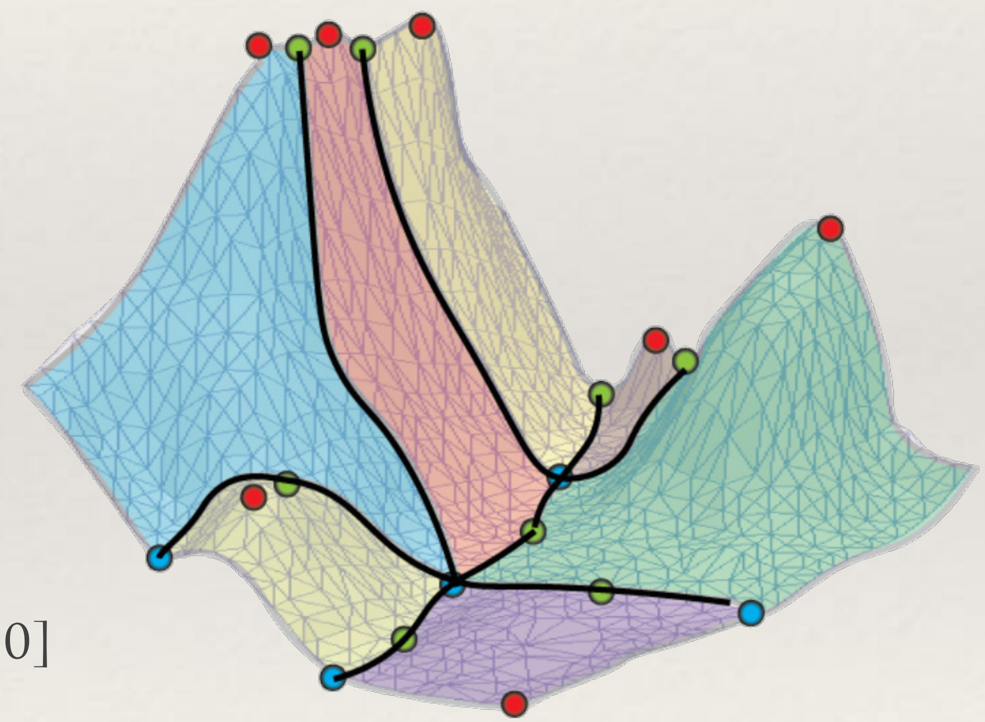
Morse Theory

Algorithms for computing Morse complexes:

Boundary-based
- ***Triangle meshes*** [Takahashi et al. '95; Edelsbrunner et al. '01; Bremer et al. '04]
- ***Tetrahedral meshes*** [Edelsbrunner et al. '03]
- ***Regular grids*** [Bajaj et al. '98; Schneider '04; Schneider '05]

Region-growing
- ***Adding triangles*** [Magillo et al. '99; Danovaro et al. '03]
- ***Adding vertices*** [Gyulassy et al. '07]

Watershed
- ***Topographic distance*** [Meyer et al. '90; Meyer '94]
- ***Simulated immersion*** [Vincent et al. '91; Soille '04]
- ***Rain falling simulation*** [Mangan et al. '99; Stove et al. '00]



Forman-based
- ***Constrained approaches*** [Cazals et al. '03; King et al. '05; Gyulassy et al. '08; Robins et al. '11; Gyulassy et al. '12]
- ***Unconstrained approaches*** [Lewiner et al. '03; Benedetti et al. '14; Harker et al. '14]
- ***Gradient traversal*** [Gunther et al. '12; Shivashankarar et al. '12; Weiss et al. '13]

Discrete Morse Theory

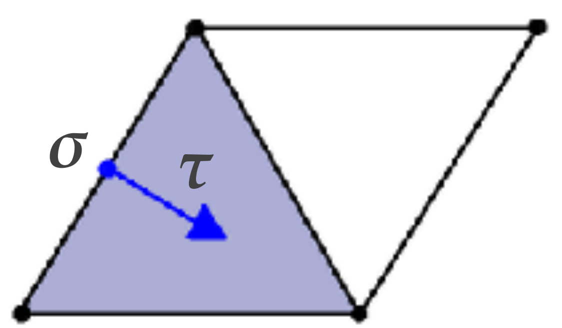
A (filtered) Forman gradient can be build by using
the **homology-preserving** operators of **reduction** and **coreduction**

Reduction and Coreduction Operators:

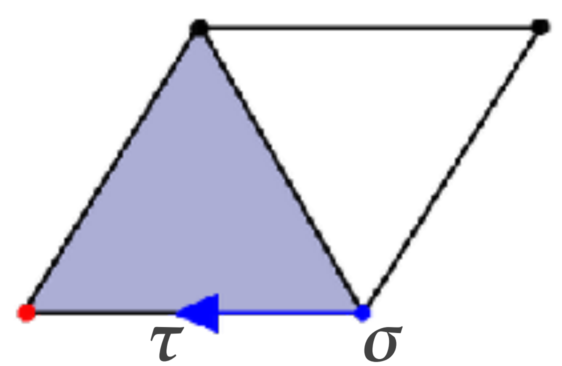
Let σ,τ be two incident simplices of dimension k and $k+1$, respectively

Pair **(σ,τ)** is called:

Reduction if
immediate coboundary of $\sigma = \{\tau\}$



Coreduction if
immediate boundary of $\tau = \{\sigma\}$



Discrete Morse Theory

Gradient through Reductions: [Benedetti et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

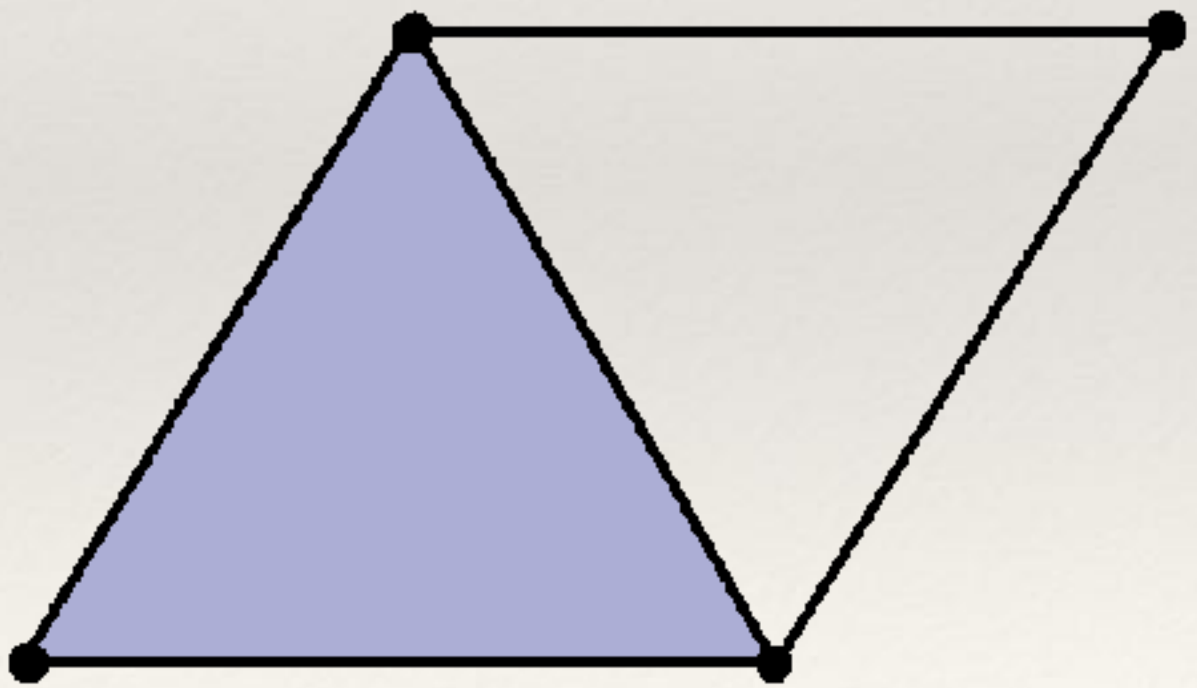
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *reduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *top simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Reductions: [Benedetti et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

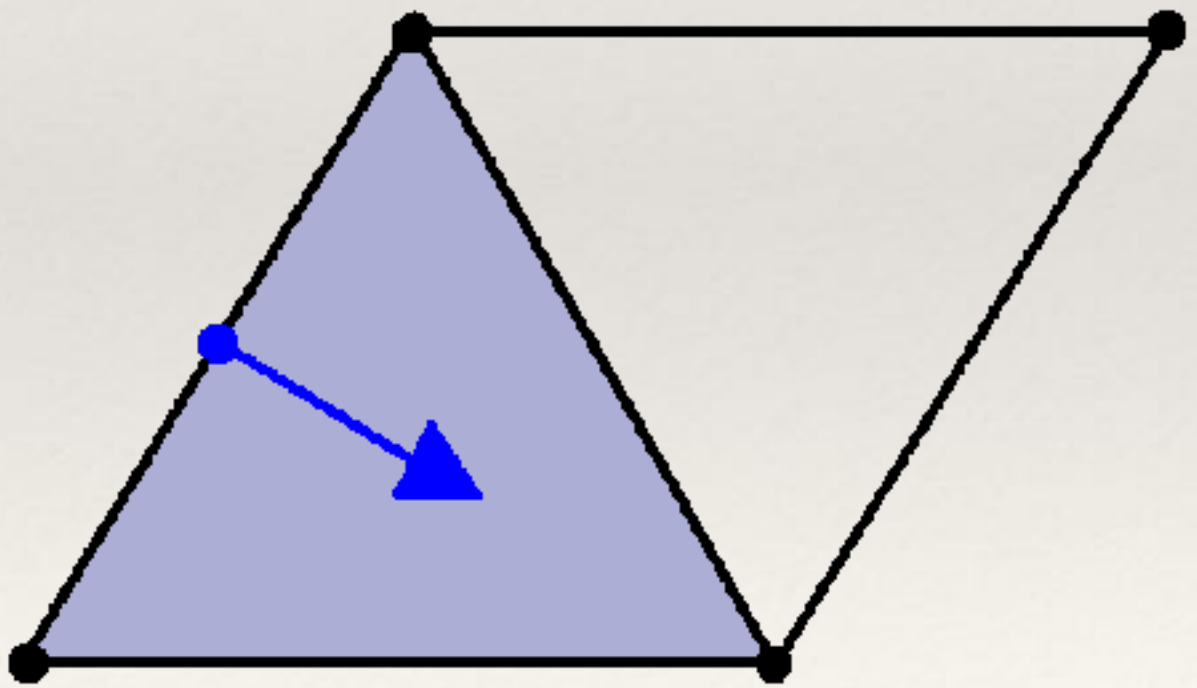
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *reduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *top simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Reductions: [Benedetti et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

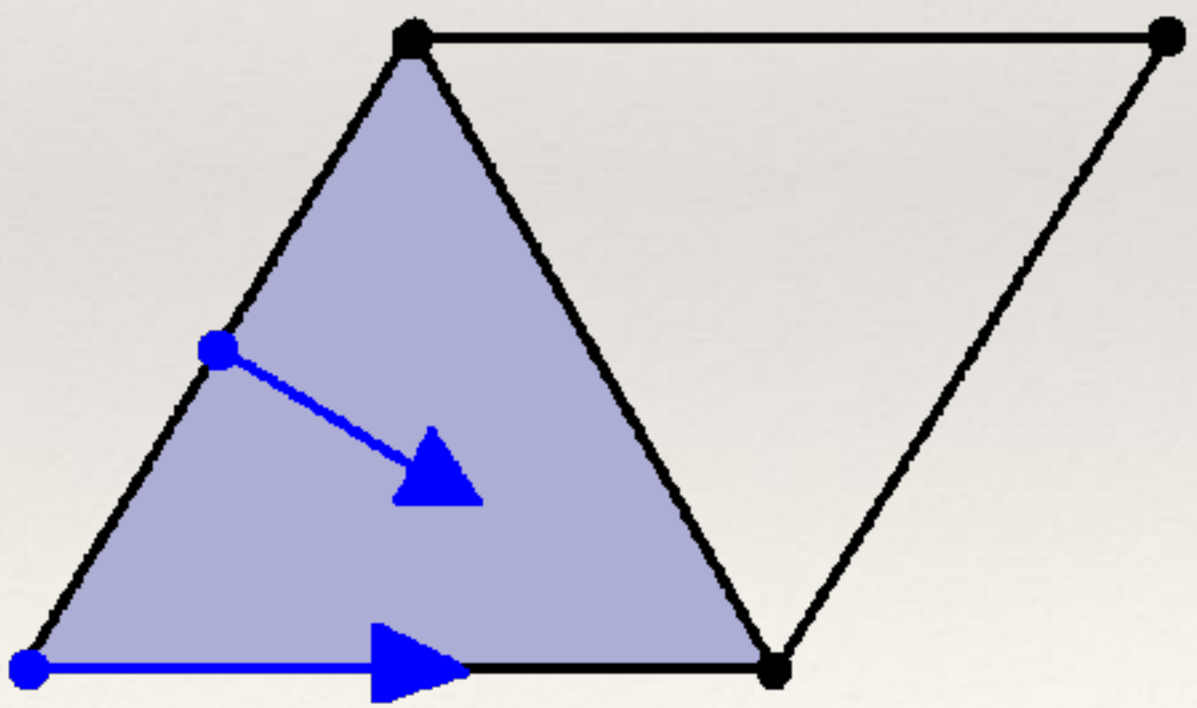
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *reduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *top simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Reductions: [Benedetti et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

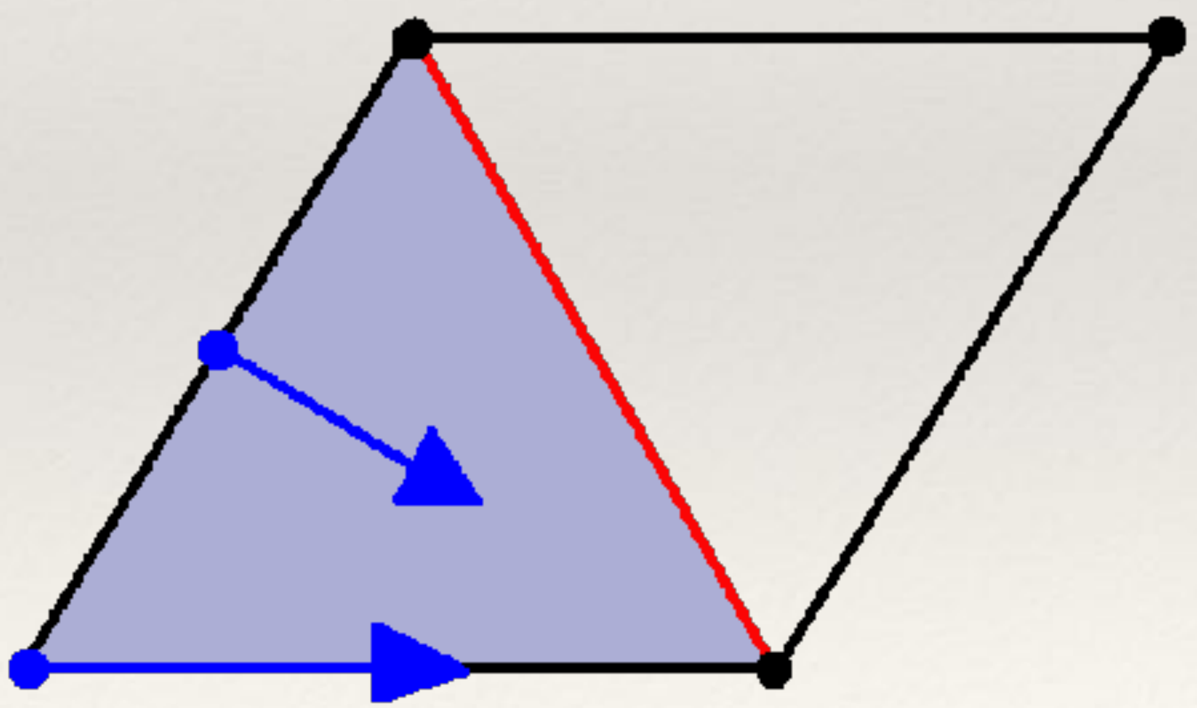
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

while Σ' admits a *reduction pair* (σ, τ) do
$V \leftarrow V \cup \{ (\sigma, \tau) \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
end while

Let η be a ***top simplex*** in Σ'
$A \leftarrow A \cup \{ \eta \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Reductions: [Benedetti et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

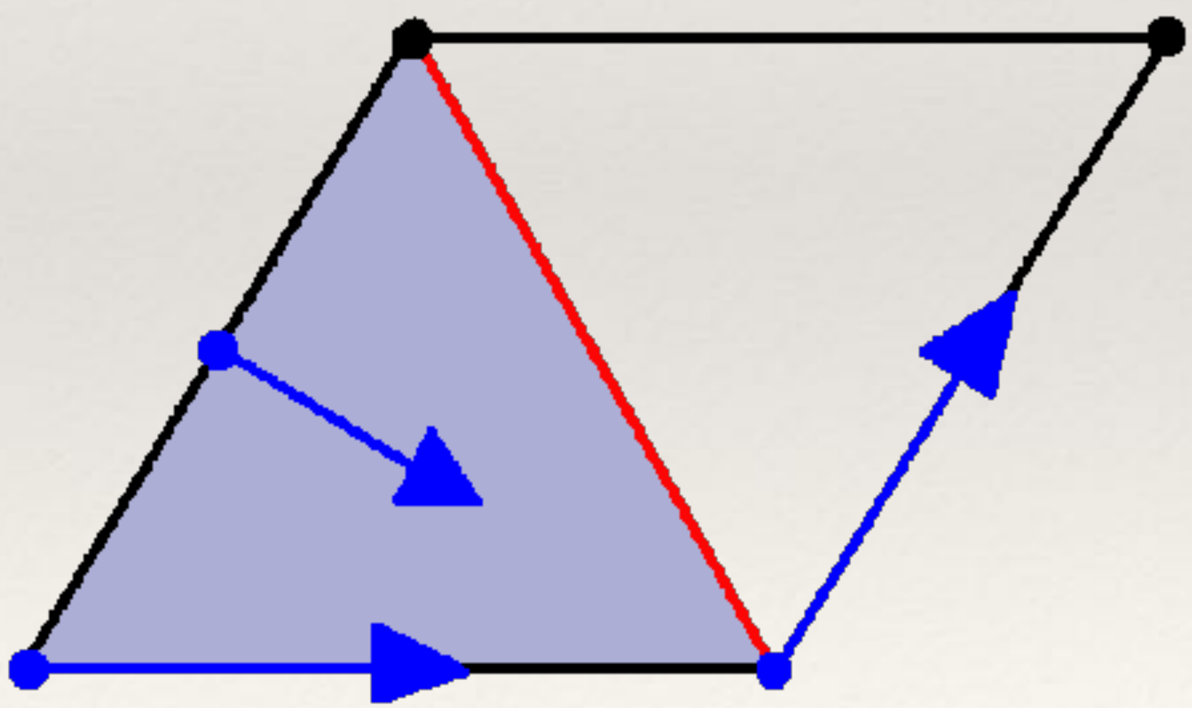
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *reduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *top simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Reductions: [Benedetti et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

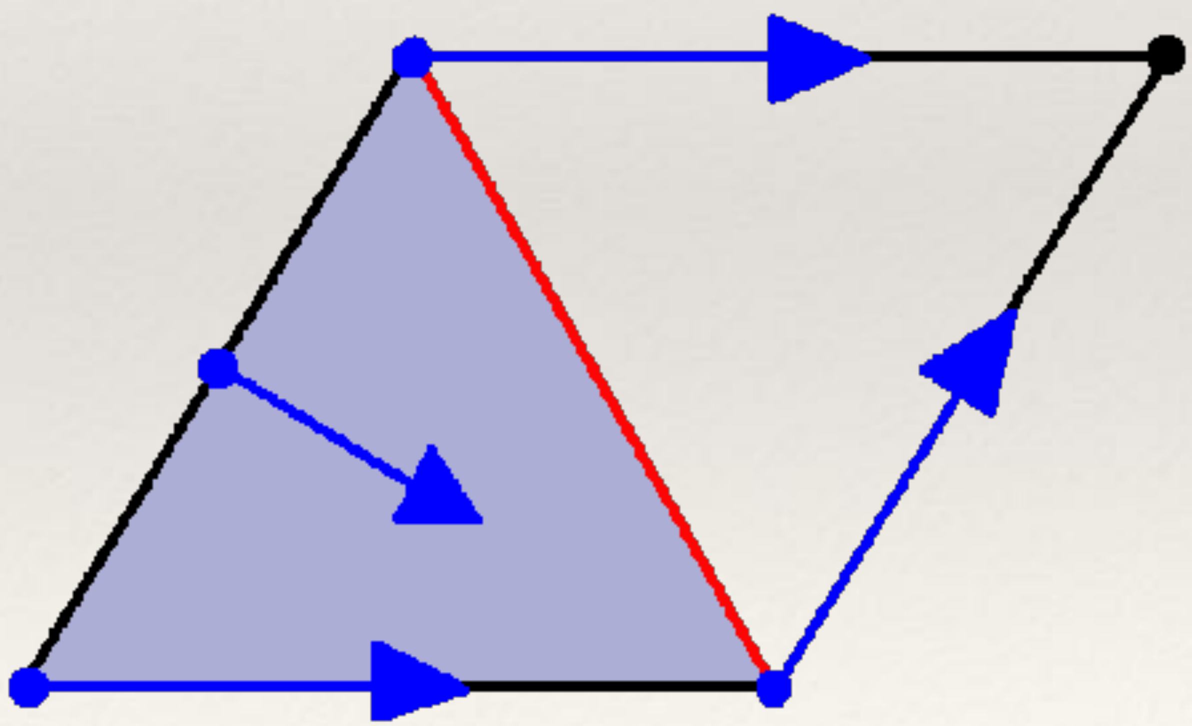
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

while Σ' admits a *reduction pair* (σ, τ) do
$V \leftarrow V \cup \{ (\sigma, \tau) \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
end while

Let η be a *top simplex* in Σ'
$A \leftarrow A \cup \{ \eta \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Reductions: [Benedetti et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

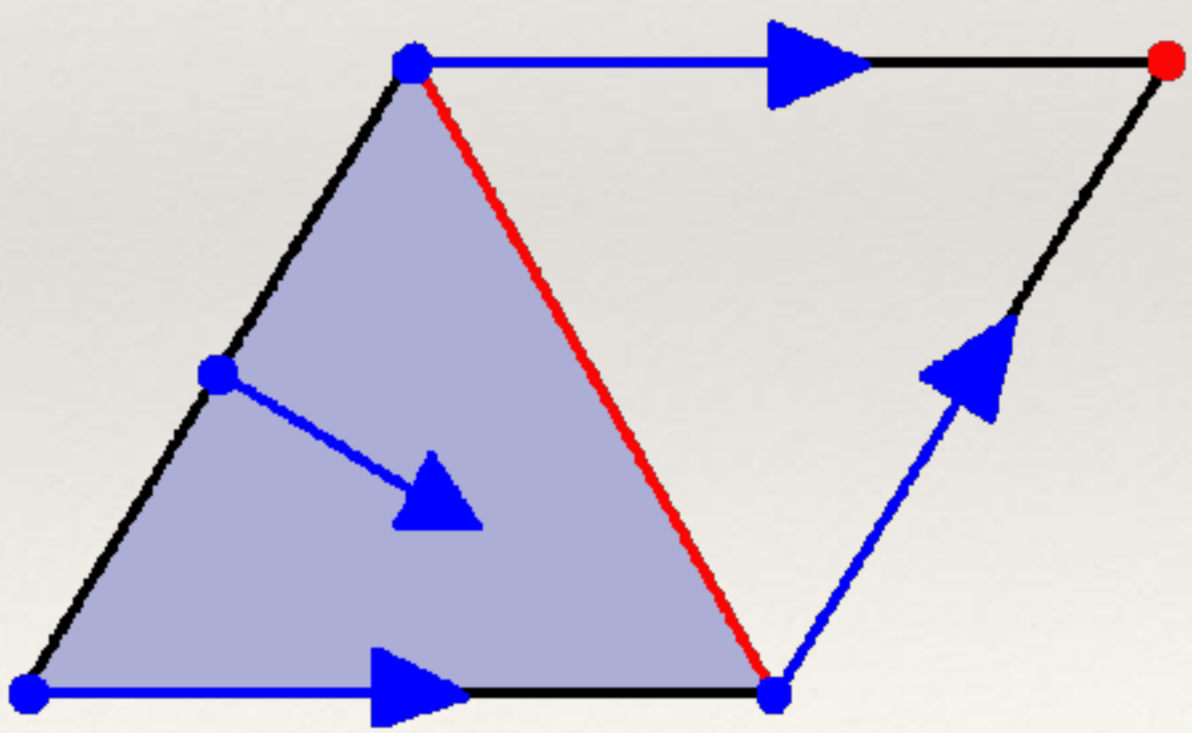
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *reduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *top simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Coreductions: [Harker et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

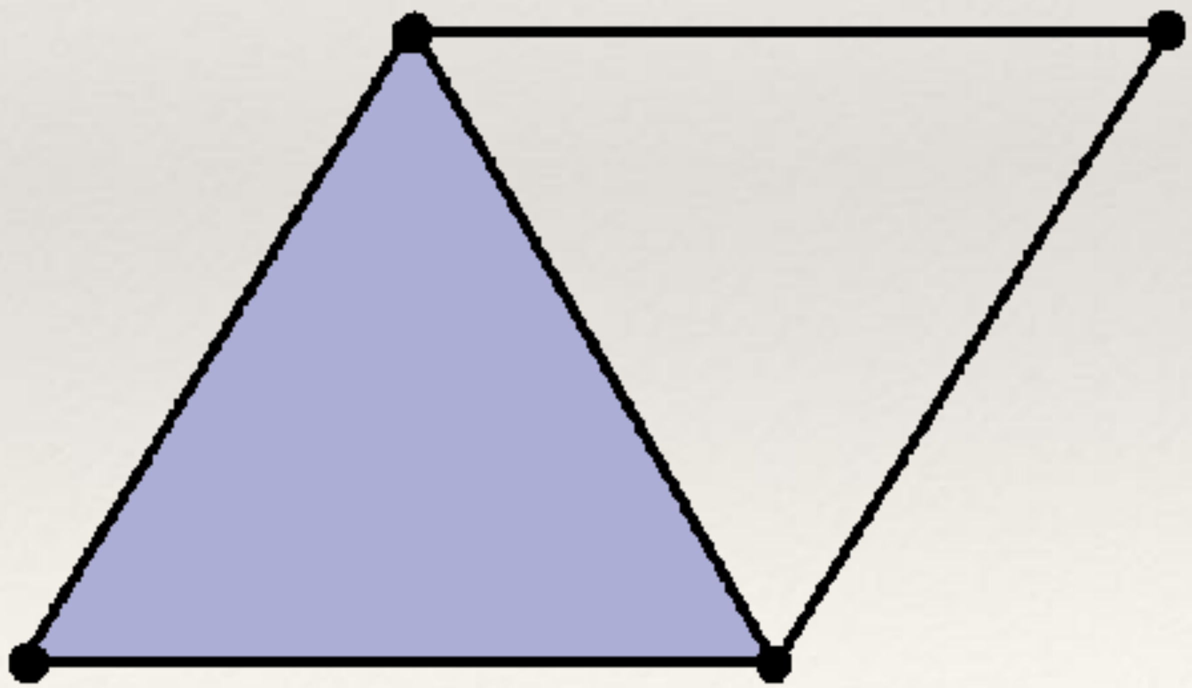
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \emptyset$, $A \leftarrow \emptyset$

while $\Sigma' \neq \emptyset$ do

  while Σ' admits a *coreduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *free simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Coreductions: [Harker et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

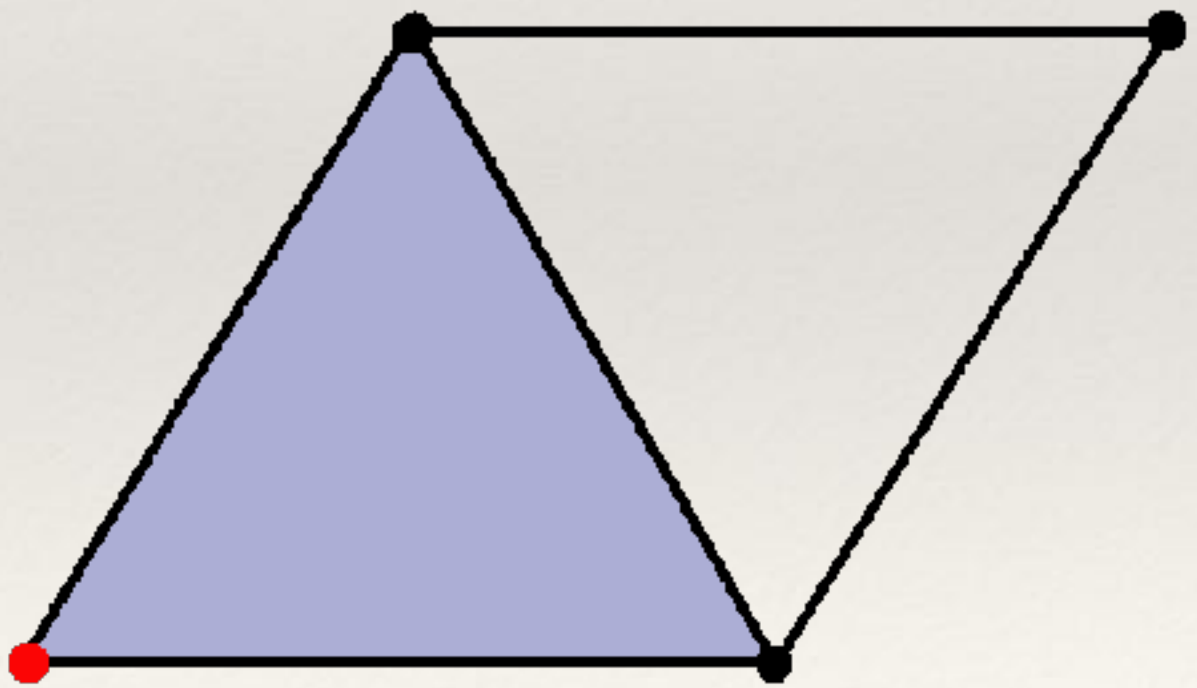
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *coreduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *free simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Coreductions: [Harker et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

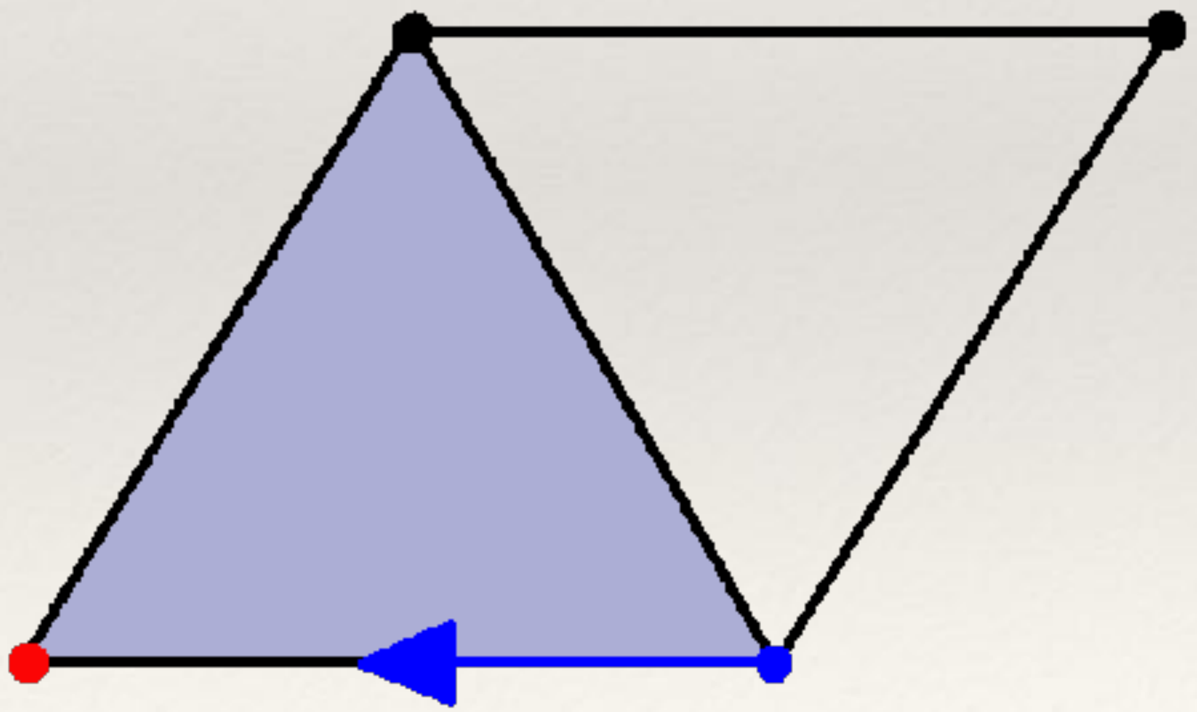
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

while Σ' admits a *coreduction pair* (σ, τ) do
$V \leftarrow V \cup \{ (\sigma, \tau) \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
end while

Let η be a *free simplex* in Σ'
$A \leftarrow A \cup \{ \eta \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Coreductions: [Harker et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

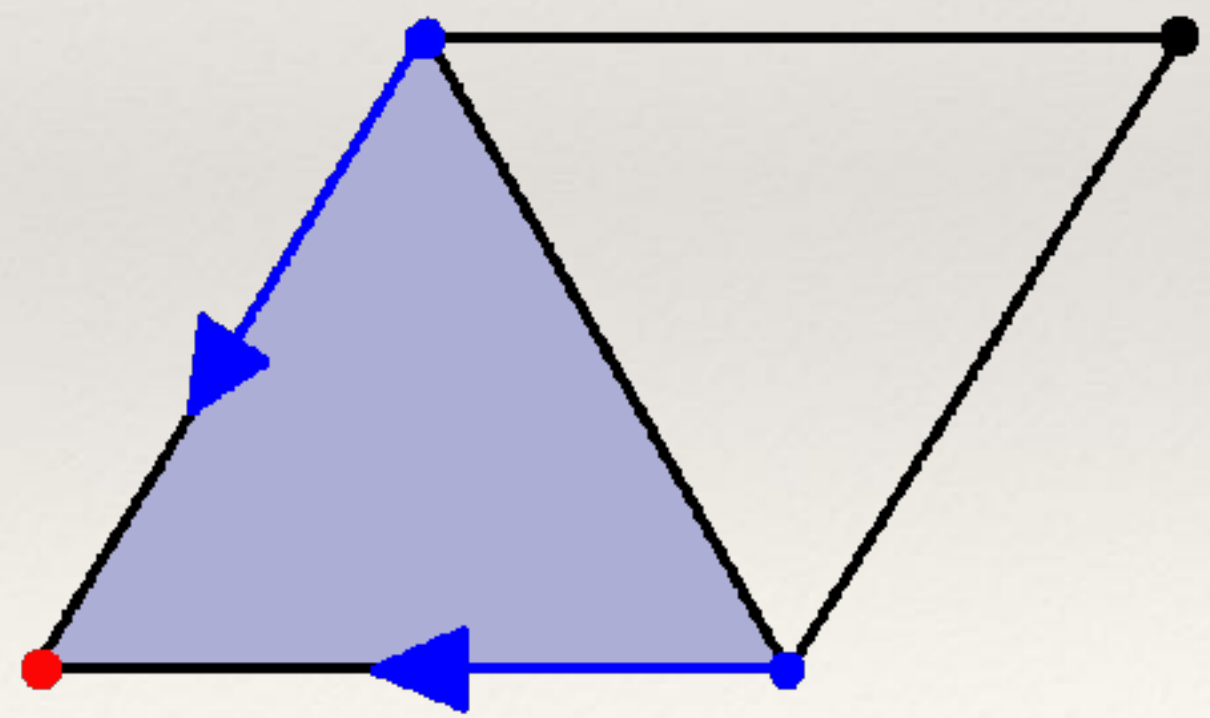
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *coreduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *free simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Coreductions: [Harker et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

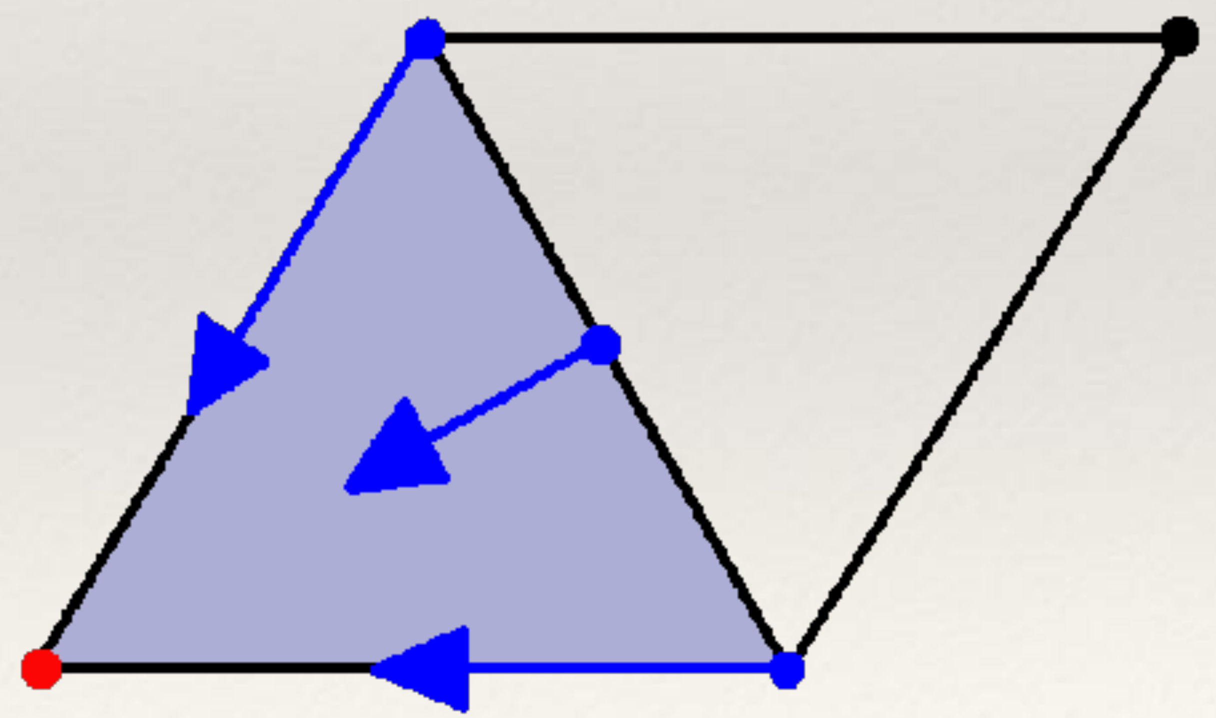
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

  while Σ' admits a *coreduction pair* (σ, τ) do
    $V \leftarrow V \cup \{ (\sigma, \tau) \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
  end while

  Let η be a *free simplex* in Σ'
    $A \leftarrow A \cup \{ \eta \}$
    $\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Coreductions: [Harker et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

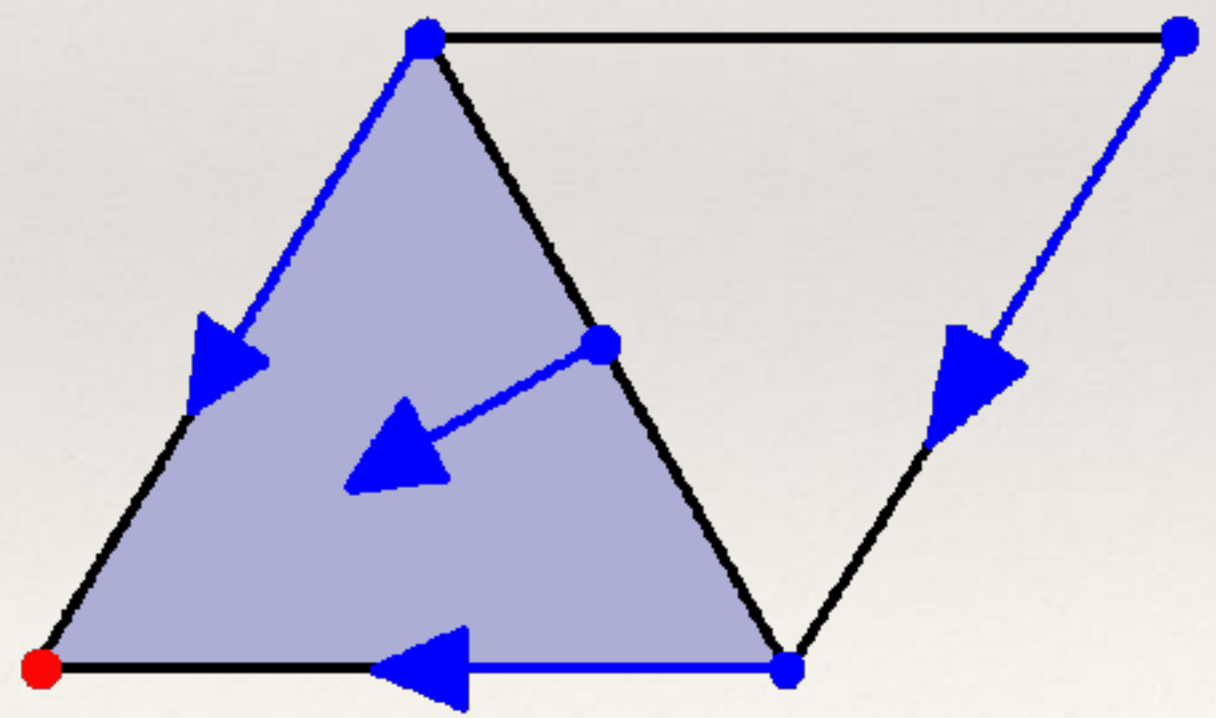
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

while Σ' admits a *coreduction pair* (σ, τ) do
$V \leftarrow V \cup \{ (\sigma, \tau) \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
end while

Let η be a *free simplex* in Σ'
$A \leftarrow A \cup \{ \eta \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Gradient through Coreductions: [Harker et al. 2014]

Input: Σ simplicial complex
Output: V gradient vector field, A set of critical simplices

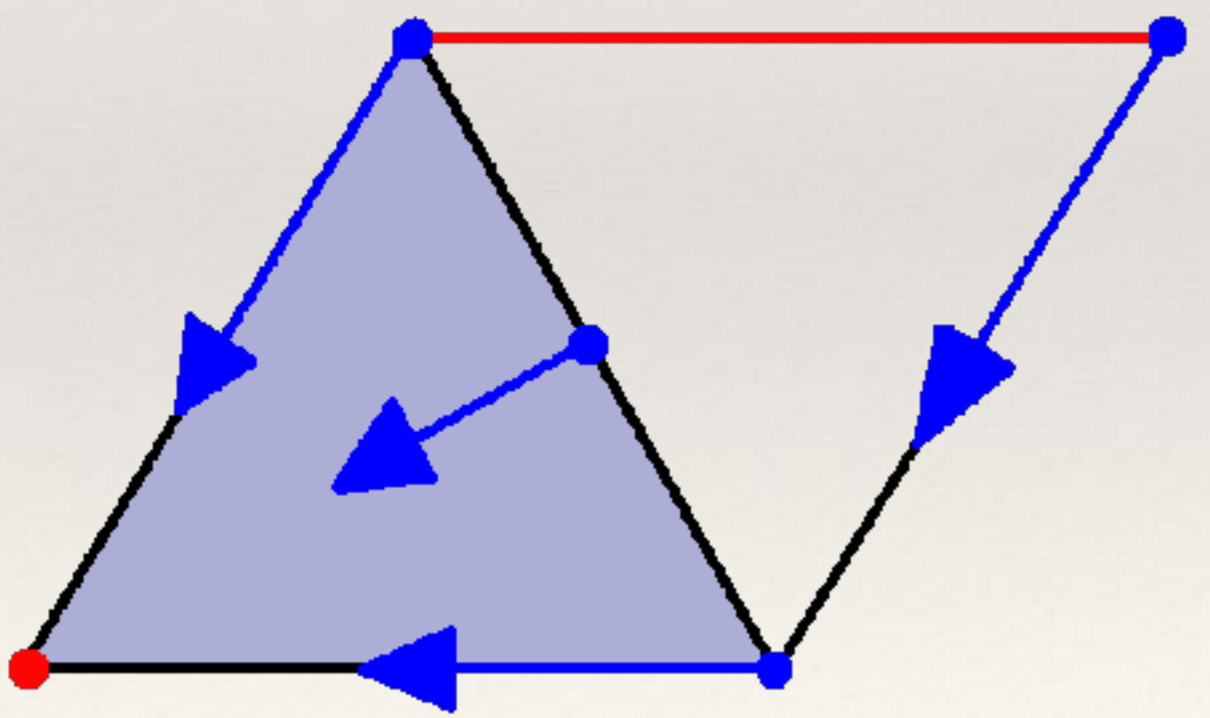
Set $\Sigma' \leftarrow \Sigma$, $V \leftarrow \varnothing$, $A \leftarrow \varnothing$

while $\Sigma' \neq \varnothing$ do

while Σ' admits a *coreduction pair* (σ, τ) do
$V \leftarrow V \cup \{ (\sigma, \tau) \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \sigma, \tau \}$
end while

Let η be a *free simplex* in Σ'
$A \leftarrow A \cup \{ \eta \}$
$\Sigma' \leftarrow \Sigma' \setminus \{ \eta \}$

end while



Discrete Morse Theory

Proposition.

Both the algorithms **produce** a **Forman gradient** on Σ

Which approach is able to compute a Forman gradient with **less critical simplices**?

Discrete Morse Theory

Proposition.

Both the algorithms **produce** a **Forman gradient** on Σ

Which approach is able to compute a Forman gradient with **less critical simplices**?

Reduction-based and coreduction-based approaches are **equivalent**

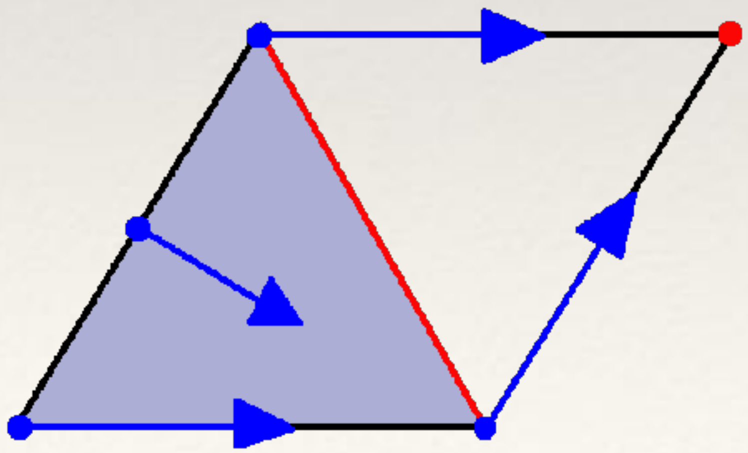
Theorem.

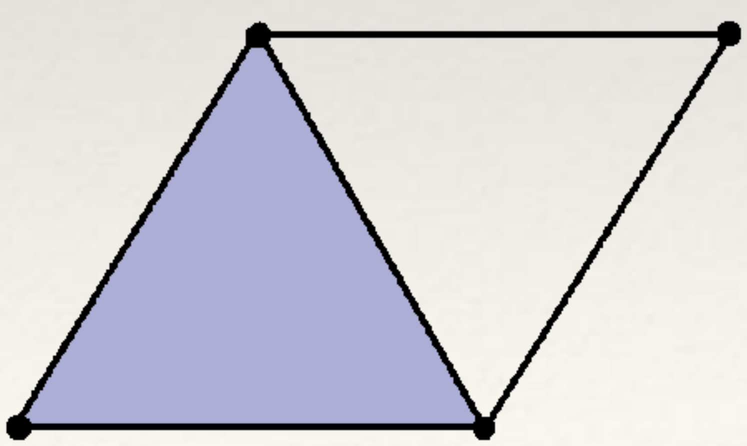
Any Forman gradient V on Σ produced by a reduction-based algorithm can be obtained through a coreduction-based algorithm; and the converse is also true

Discrete Morse Theory

Proof's guidelines: (from reduction-based to coreduction-based)

- Consider a simplicial complex Σ and run the reduction-based approach on it
- Take the sequence of reduction pairs and top simplex removals operated by the algorithm
- Reverse the order of the sequence: this new sequence represents for Σ a performable sequence of coreduction pairs and free simplex removals

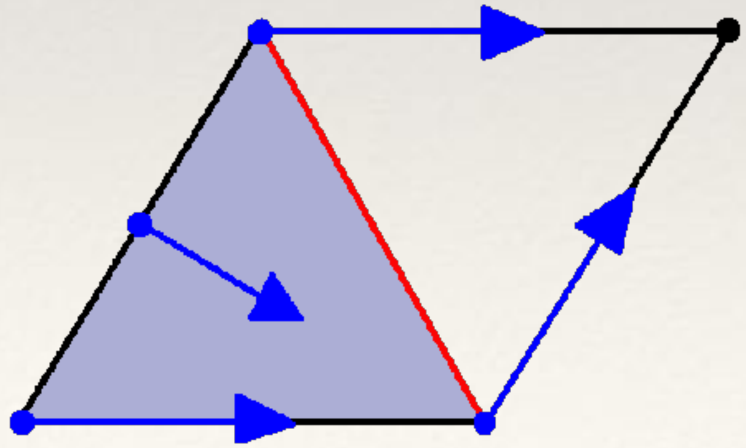


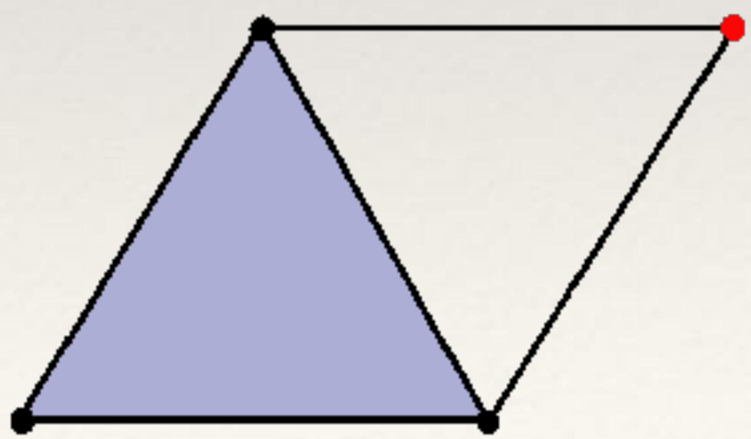


Discrete Morse Theory

Proof's guidelines: (from reduction-based to coreduction-based)

- Consider a simplicial complex Σ and run the reduction-based approach on it
- Take the sequence of reduction pairs and top simplex removals operated by the algorithm
- Reverse the order of the sequence: this new sequence represents for Σ a performable sequence of coreduction pairs and free simplex removals

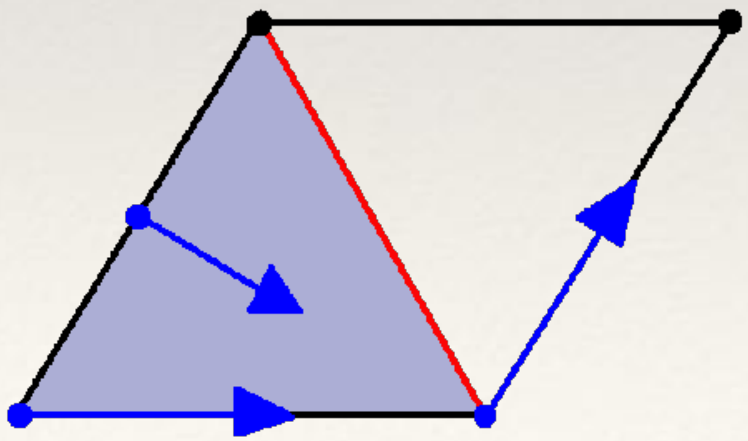


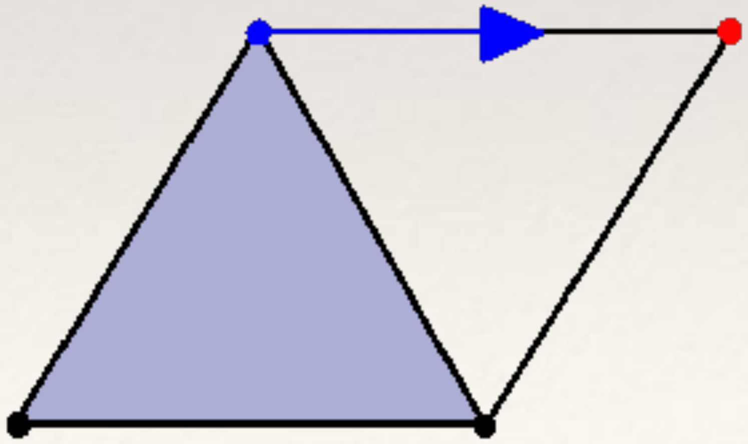


Discrete Morse Theory

Proof's guidelines: (from reduction-based to coreduction-based)

- Consider a simplicial complex Σ and run the reduction-based approach on it
- Take the sequence of reduction pairs and top simplex removals operated by the algorithm
- Reverse the order of the sequence: this new sequence represents for Σ a performable sequence of coreduction pairs and free simplex removals

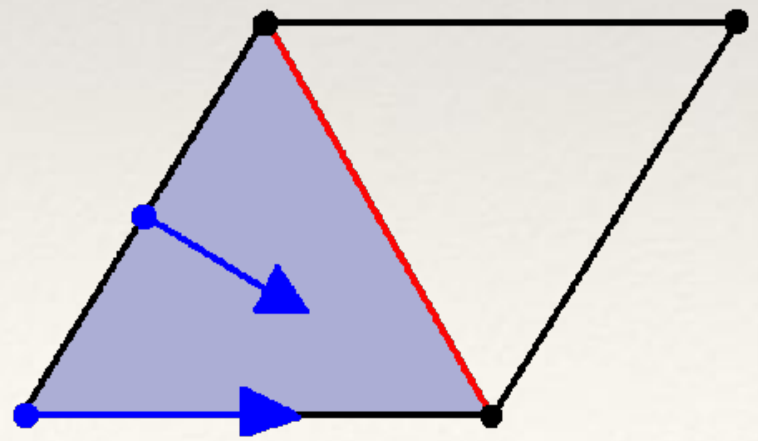


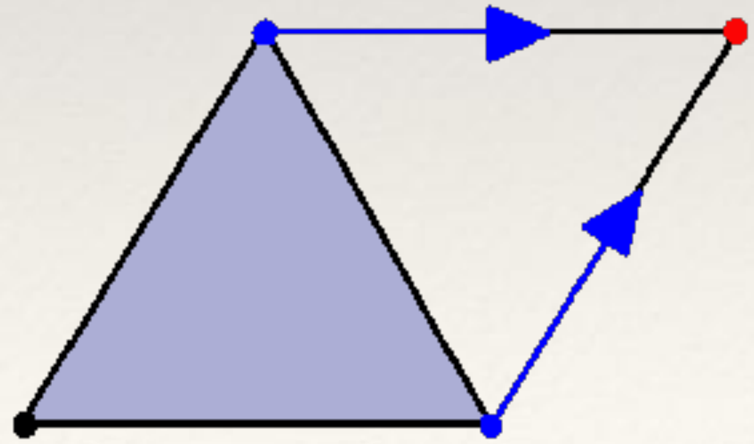


Discrete Morse Theory

Proof's guidelines: (from reduction-based to coreduction-based)

- Consider a simplicial complex Σ and run the reduction-based approach on it
- Take the sequence of reduction pairs and top simplex removals operated by the algorithm
- Reverse the order of the sequence: this new sequence represents for Σ a performable sequence of coreduction pairs and free simplex removals

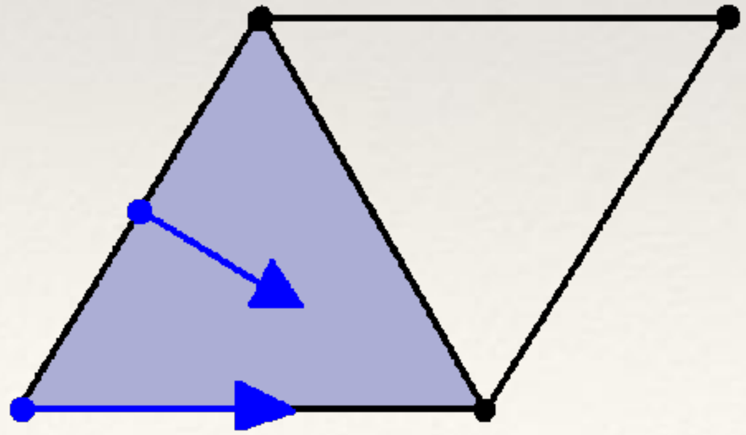


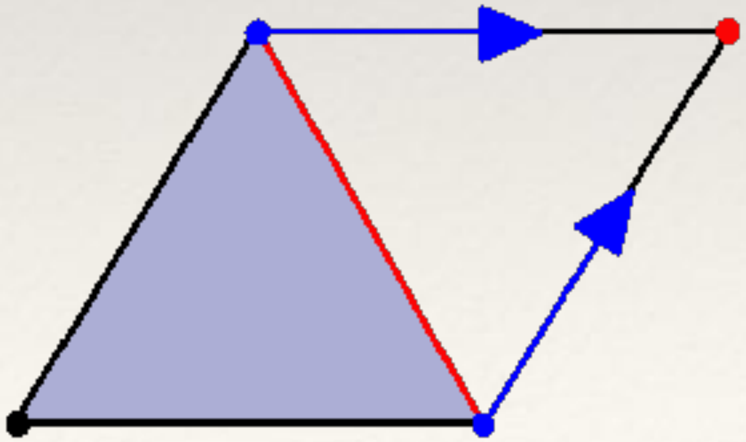


Discrete Morse Theory

Proof's guidelines: (from reduction-based to coreduction-based)

- Consider a simplicial complex Σ and run the reduction-based approach on it
- Take the sequence of reduction pairs and top simplex removals operated by the algorithm
- Reverse the order of the sequence: this new sequence represents for Σ a performable sequence of coreduction pairs and free simplex removals

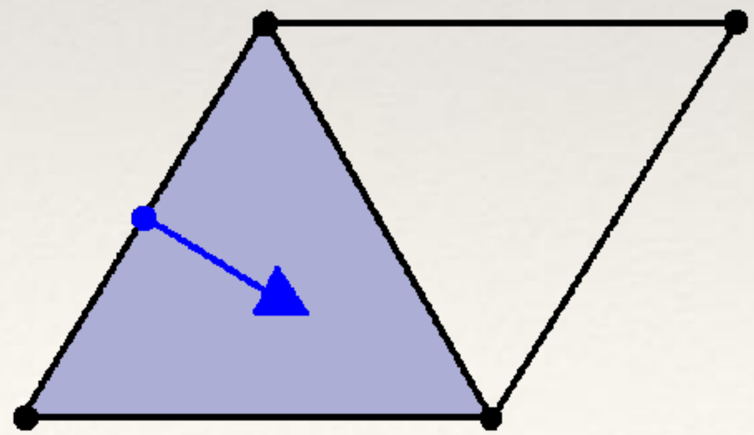


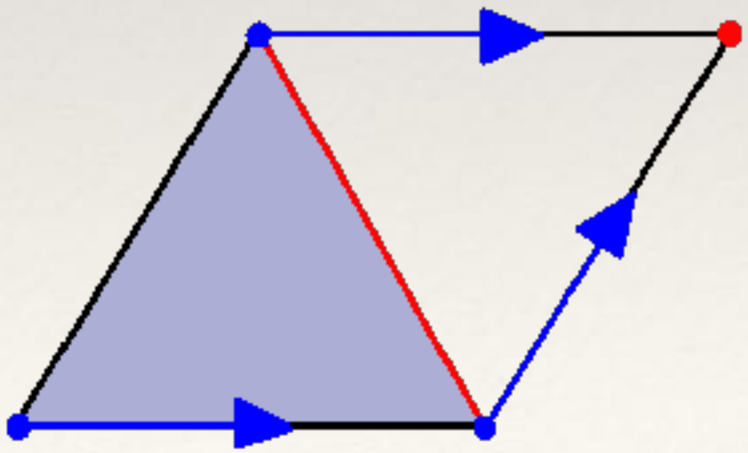


Discrete Morse Theory

Proof's guidelines: (from reduction-based to coreduction-based)

- Consider a simplicial complex Σ and run the reduction-based approach on it
- Take the sequence of reduction pairs and top simplex removals operated by the algorithm
- Reverse the order of the sequence: this new sequence represents for Σ a performable sequence of coreduction pairs and free simplex removals

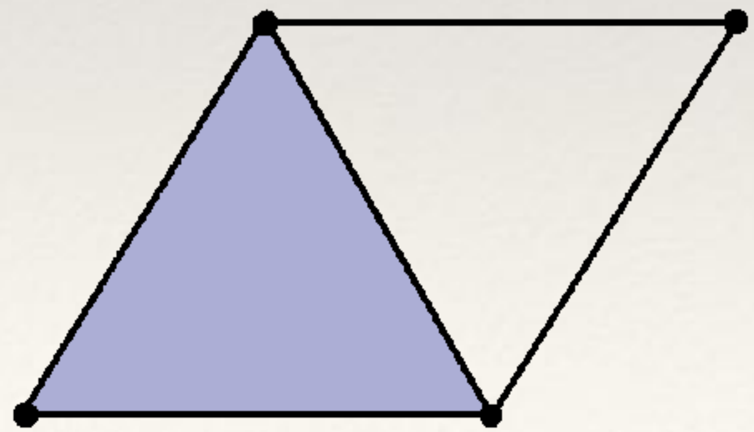


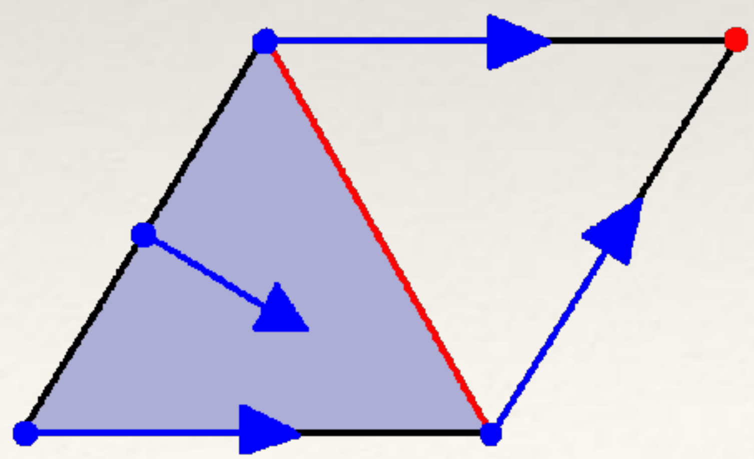


Discrete Morse Theory

Proof's guidelines: (from reduction-based to coreduction-based)

- Consider a simplicial complex Σ and run the reduction-based approach on it
- Take the sequence of reduction pairs and top simplex removals operated by the algorithm
- Reverse the order of the sequence: this new sequence represents for Σ a performable sequence of coreduction pairs and free simplex removals





Discrete Morse Theory

Interleaved Approach:

Another class of approaches **interleaving reductions and coreductions** has been considered

Proposition.

Any interleaved approach **produces** a **Forman gradient** on Σ

Discrete Morse Theory

Interleaved Approach:

Another class of approaches **interleaving reductions and coreductions** has been considered

Proposition.

Any interleaved approach **produces** a **Forman gradient** on Σ

Each interleaved approach has **equivalent** capabilities

Theorem.

Any Forman gradient V on Σ produced by an interleaved algorithm can be obtained through a reduction-based algorithm or, equivalently, through a coreduction-based algorithm

Outline

Persistent Homology
Computation

Discrete Morse Theory and
Persistent Homology

Thank you

Ulderico Fugacci

***TU Kaiserslautern**, Dept. of Computer Science*