Persistent Homology: Formalization

Tao Hou, University of Oregon

Outline for studying persistent homology

1. Intro to persistent homology
   - Build intuitions of persistent homology: what it does, what it produces
2. Formalizing persistent homology
   - Introduce its input (filtration) and study an algorithm for computation
3. Different ways for building filtrations
   - Vietoris-Rips filtration, sub-levelset filtration
   - Cubical complexes (for images)
4. Interpretation and stability of persistence diagram

“The growing space”

- Recall the growing space:
  - We have a value α ranging within an interval, say, from 0 to ∞
  - Let each value α corresponds to a topological space so that
  - The topological space grows as α increases from 0 to ∞

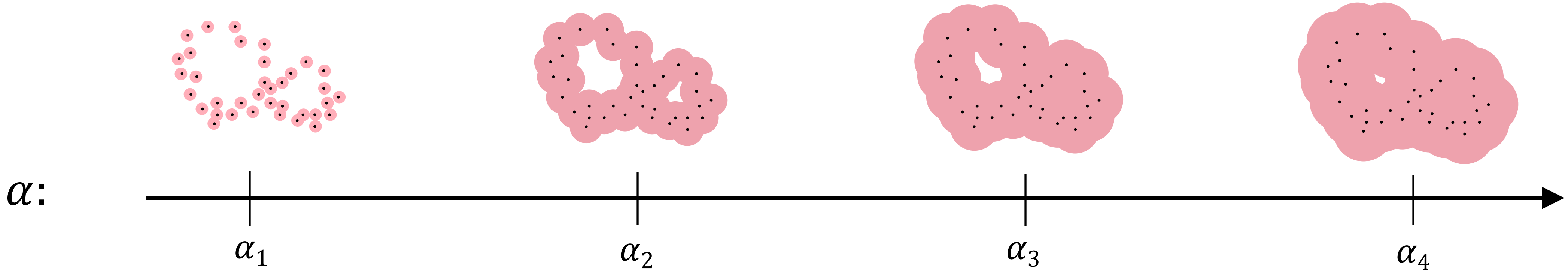
"The growing space"

- Recall the growing space:
  - We have a value α ranging within an interval, say, from 0 to ∞
  - Let each value α corresponds to a topological space so that
  - The topological space grows as α increases from 0 to ∞
- Suppose I ask you to represent such a growing space in the computer, can you think of any problems?

"The growing space"

- Problem 1:
  - When α ranges within an interval $[s, f]$, no matter how small the interval is, there are always infinitely many values within the interval

“The growing space”



- Problem 1:
  - When α ranges within an interval $[s, f]$, no matter how small the interval is, there are always infinitely many values within the interval
  - Each α value may correspond to a possibly different space

“The growing space”

- Problem 1:
  - When α ranges within an interval $[s, f]$, no matter how small the interval is, there are always infinitely many values within the interval
  - Each α value may correspond to a possibly different space
  - This means there could be infinitely many spaces that we need to store in the computer, which is impossible

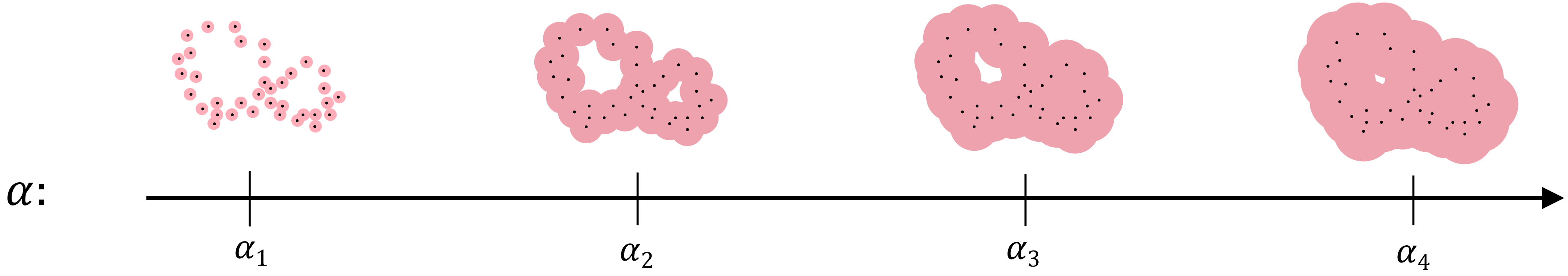
“The growing space”

- Solution:
  - While there are infinitely many values for α, our data is still “finite” (e.g., the above point cloud contains finitely many points)

“The growing space”

- Solution:
  - While there are infinitely many values for α, our data is still “finite” (e.g., the above point cloud contains finitely many points)
  - This means that there are only finitely many values of α where the topological space “essentially changes”

“The growing space”

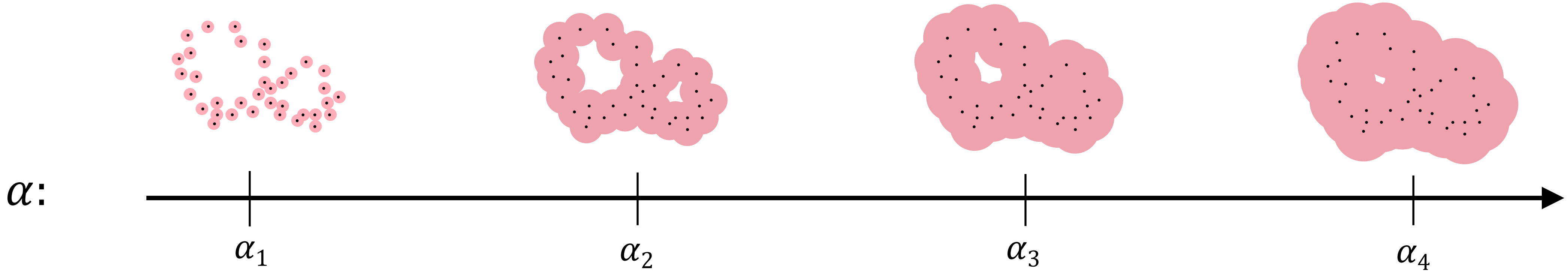


- Solution:
  - While there are infinitely many values for α, our data is still “finite” (e.g., the above point cloud contains finitely many points)
  - This means that there are only finitely many values of α where the topological space “essentially changes”
  - So we only need to record finitely many spaces in computer

“The growing space”

- Remark
  - We will not be very accurate on what the “essential changes” mean here (should be clearer later)

"The growing space"

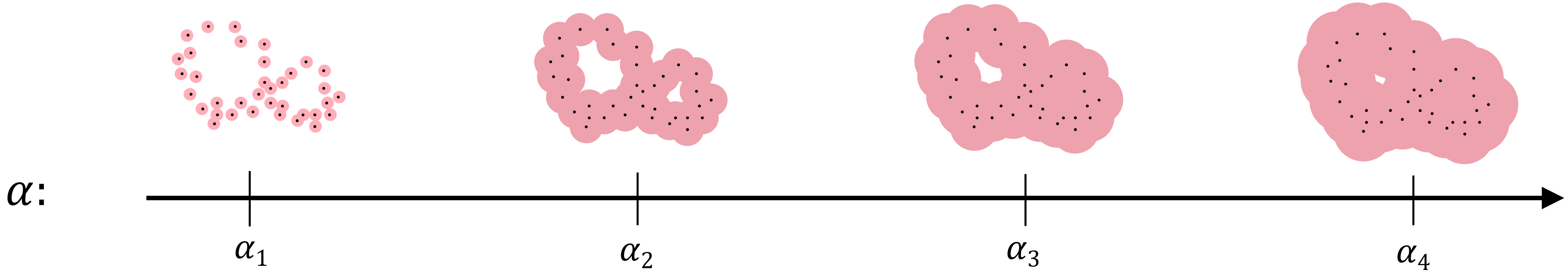


- Remark
  - We will not be very accurate on what the "essential changes" mean here (should be clearer later)
  - BTW, these values where topological space "essentially changes" are called critical values
  - Critical values are important concepts in "Morse theory", but we will not go very deep on it in this course

“The growing space”

- Problem 2:

“The growing space”



- Problem 2:
  - Even there are finitely many spaces to record, we still need a way to represent each topological space in computer

"The growing space"

- Problem 2:
  - Even there are finitely many spaces to record, we still need a way to represent each topological space in computer
- Solution:

“The growing space”

- Problem 2:
  - Even there are finitely many spaces to record, we still need a way to represent each topological space in computer
- Solution:
  - Using simplicial complexes!

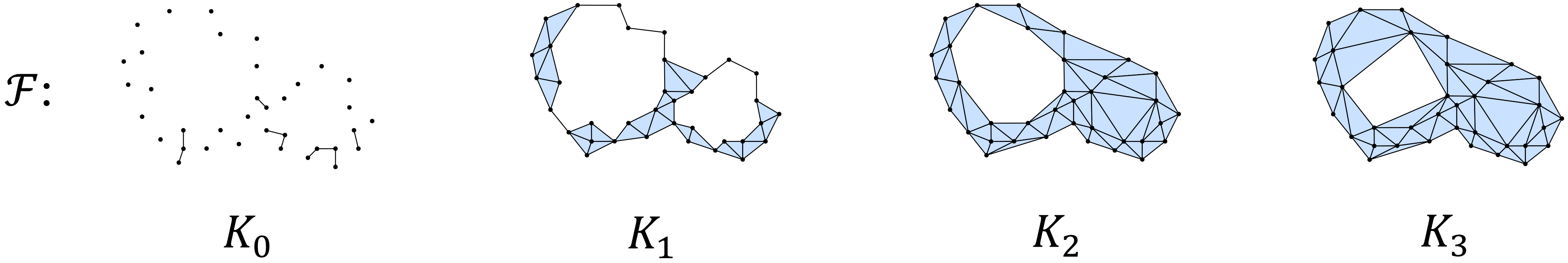
Filtration

- Hence, the "growing space" in computer is represented by a finite sequence of simplicial complexes, called a **filtration**, which is typically denoted by a calligraphic letter $\mathcal{F}$,

$$\mathcal{F}:\ K_0, K_1, \ldots, K_m$$

Filtration

- **Below** is an example of a filtration:



Filtration

- Another example:

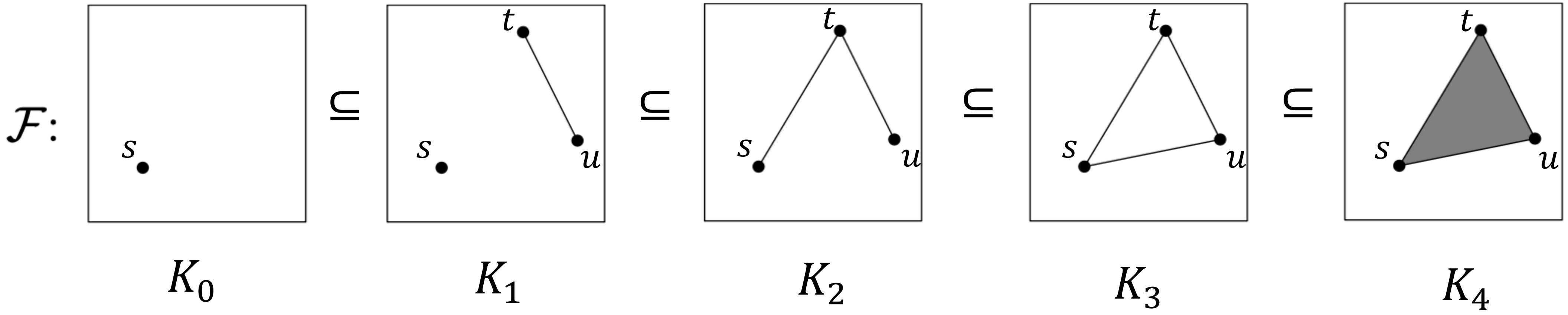


Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

Filtration

- Question: In previous definition, a filtration is only a sequence of complexes.
- How do we account for the fact that the spaces (complexes) grow?

Filtration

- Question: In previous definition, a filtration is only a sequence of complexes.
- How do we account for the fact that the spaces (complexes) grow?
- Answer: We make sure the complexes "grow" by making sure the previous complex is a "subset" (subcomplex) of the next complex.

Filtration

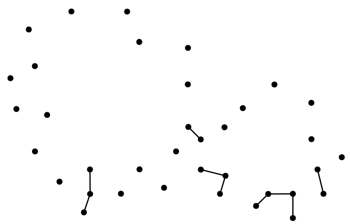
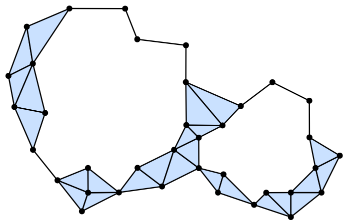
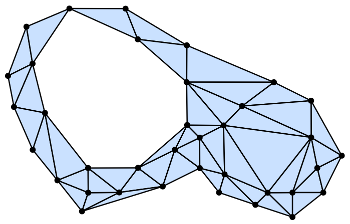
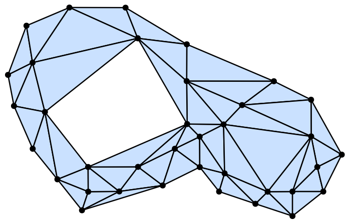
- Question: In previous definition, a filtration is only a sequence of complexes.
- How do we account for the fact that the spaces (complexes) grow?
- Answer: We make sure the complexes "grow" by making sure the previous complex is a "subset" (subcomplex) of the next complex.
- **Definition**: A **filtration** is a nested sequence of simplicial complexes

$$\mathcal{F}\colon\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$$

such that each K_i is a subcomplex of K_{i+1}.

Filtration

- Example:

$\mathcal{F}$:  $\subseteq$  $\subseteq$  $\subseteq$ 

Filtration

- Another example:

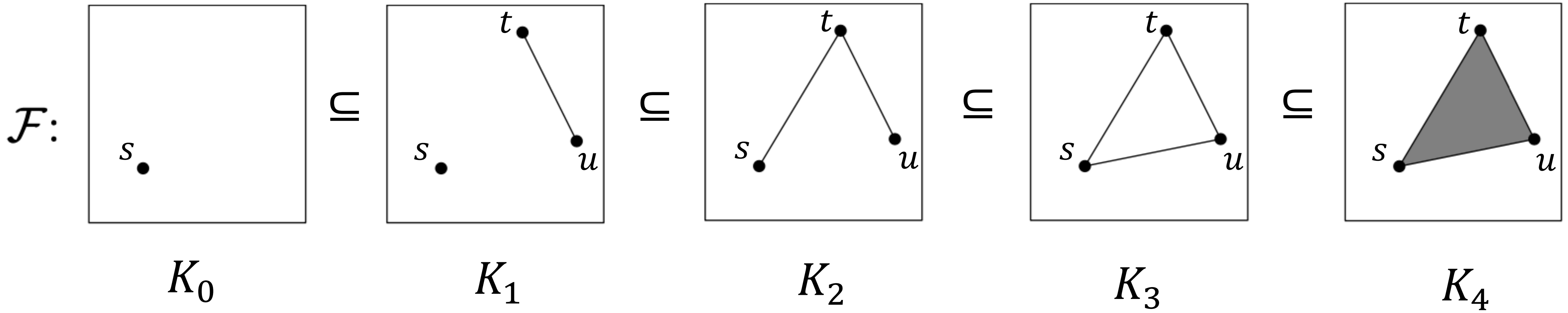
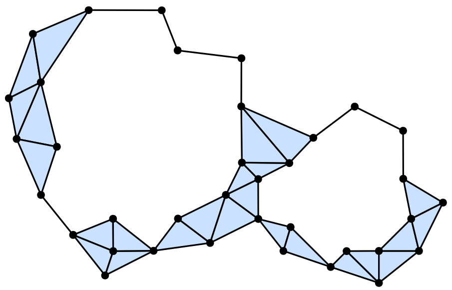
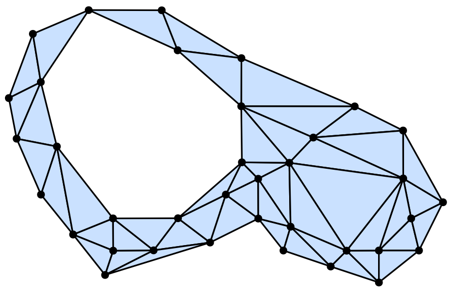


Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

Filtration

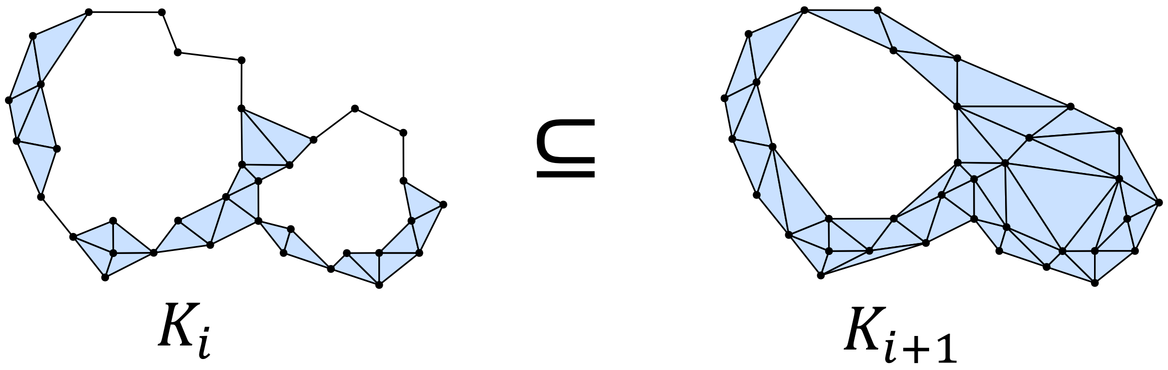




$$K_i \subseteq K_{i+1}$$

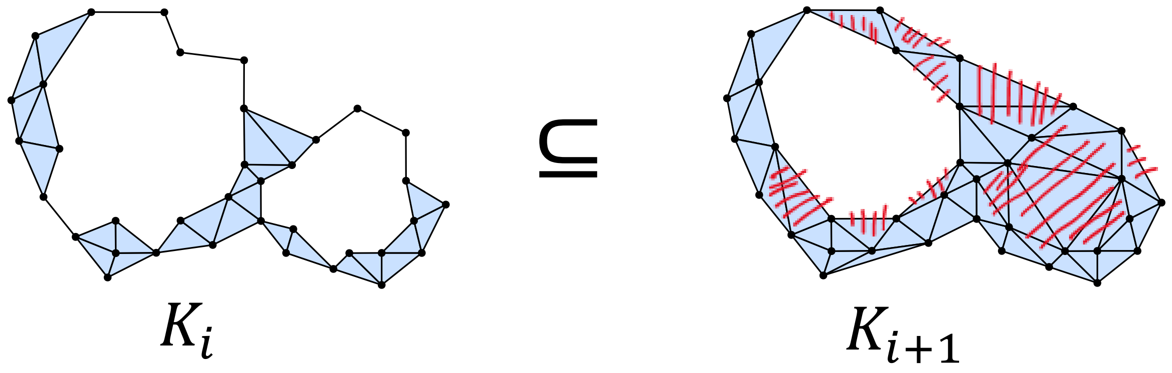
- Now we want to further interpret a filtration
- For this, we focus on a single inclusion in a filtration

Filtration



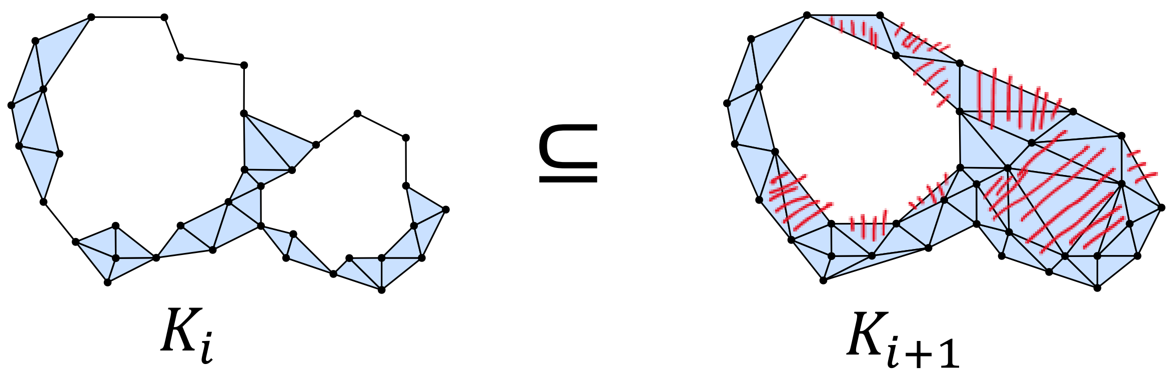
- Now we want to further interpret a filtration
- For this, we focus on a single inclusion in a filtration
- Since it's an inclusion, the difference of the two complexes is that K_{i+1} has some additional simplices than K_i

Filtration



- Now we want to further interpret a filtration
- For this, we focus on a single inclusion in a filtration
- Since it's an inclusion, the difference of the two complexes is that K_{i+1} has some additional simplices than K_i

Filtration



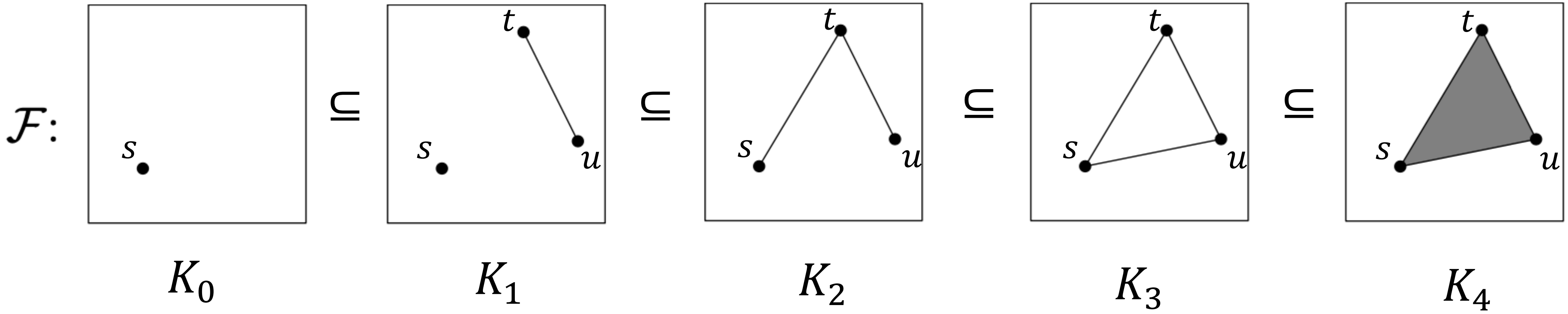
- Now we want to further interpret a filtration
- For this, we focus on a single inclusion in a filtration
- Since it's an inclusion, the difference of the two complexes is that K_{i+1} has some additional simplices than K_i
- So we can consider each inclusion $K_i \subseteq K_{i+1}$ in a filtration

$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$$

as an insertion of a bunch of simplices

Filtration

For the example:



- K_0 to K_1: insert vertices t and u and edge tu
- K_1 to K_2: insert edge st
- K_2 to K_3: insert edge su
- K_3 to K_4: insert triangle stu

Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

Filtration

- More **regulations:** For a filtration

$$\mathcal{F}\colon\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$$

we typically let the first complex K_0 be empty, and call the last complex K_m the "total complex" (because it contains all simplices) and denote it as K.

Filtration

- More **regulations:** For a filtration

$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$$

we typically let the first complex K_0 be empty, and call the last complex K_m the "total complex" (because it contains all simplices) and denote it as K.

- $\mathcal{F}$ is then called a filtration of K (because it eventually grows into K):

$$\mathcal{F}:\ \emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

Filtration

- More **regulations:** For a filtration

$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$$

we typically let the first complex K_0 be empty, and call the last complex K_m the "total complex" (because it contains all simplices) and denote it as K.

- $\mathcal{F}$ is then called a filtration of K (because it eventually grows into K):

$$\mathcal{F}:\ \emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

- **Observation**: (1). Any simplex of K is added exactly once in $\mathcal{F}$

  (2). For any two simplices σ and τ in K such that σ is a face of τ, we have σ cannot be added later than τ.

Filtration

- More **regulations:** For a filtration

$$\mathcal{F}:\ K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$$

we typically let the first complex K_0 be empty, and call the last complex K_m the "total complex" (because it contains all simplices) and denote it as K.

- $\mathcal{F}$ is then called a filtration of K (because it eventually grows into K):

$$\mathcal{F}:\ \emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

- **Observation**: (1). Any simplex of K is added exactly once in $\mathcal{F}$

  (2). For any two simplices σ and τ in K such that σ is a face of τ, we have σ cannot be added later than τ.

- (1) is easy to see. To see (2), suppose that σ is added later than τ. Then at a certain time, τ is already added to a complex K_i but σ is not in K_i yet. This contradicts the fact that any face of a simplex in the complex is also in the complex.

PD for Filtration

- Filtrations are inputs to the persistent homology pipeline that we want to formalize

PD for Filtration

- Filtrations are inputs to the persistent homology pipeline that we want to formalize
- But still we need to formally define a PD on a filtration of simplicial complexes

PD for Filtration

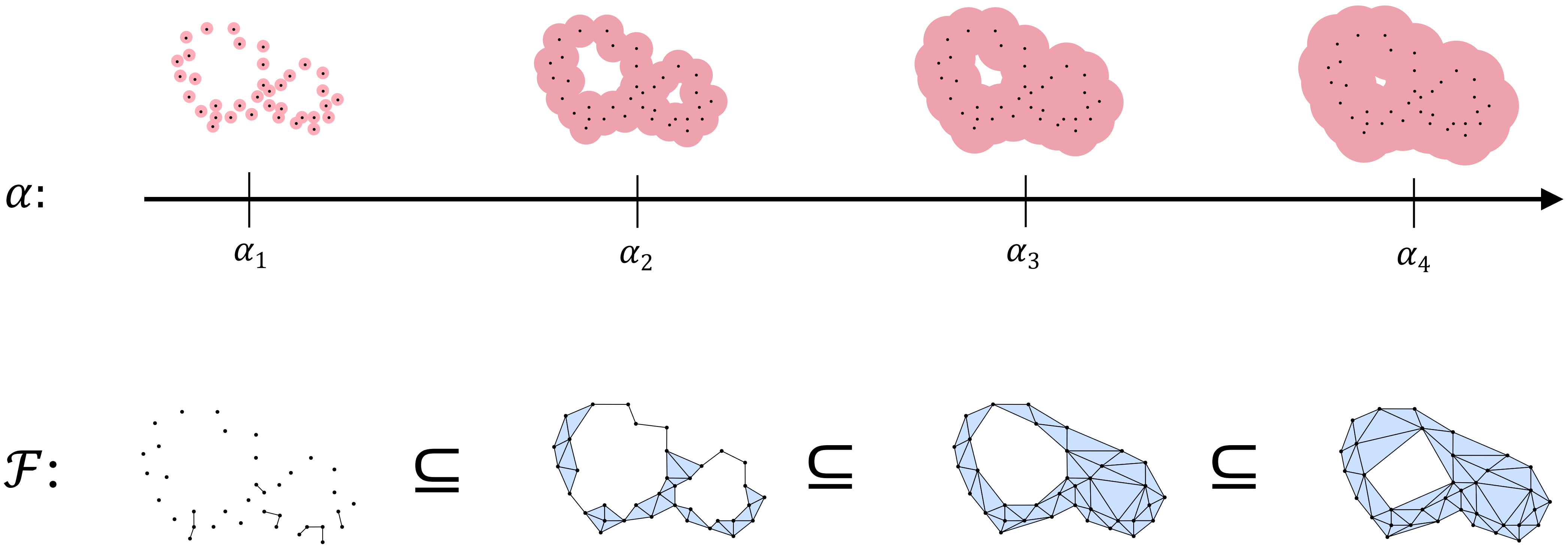
- Filtrations are inputs to the persistent homology pipeline that we want to formalize
- But still we need to formally define a PD on a filtration of simplicial complexes
- Previously, we only saw some examples of PD on a sequence of "growing spaces", which are not exactly a filtration of complexes.

PD for Filtration

- Filtrations are inputs to the persistent homology pipeline that we want to formalize
- But still we need to formally define a PD on a filtration of simplicial complexes
- Previously, we only saw some examples of PD on a sequence of "growing spaces", which are not exactly a filtration of complexes.
- Moreover, we haven't really formally defined a PD on a growing space other than showing some examples

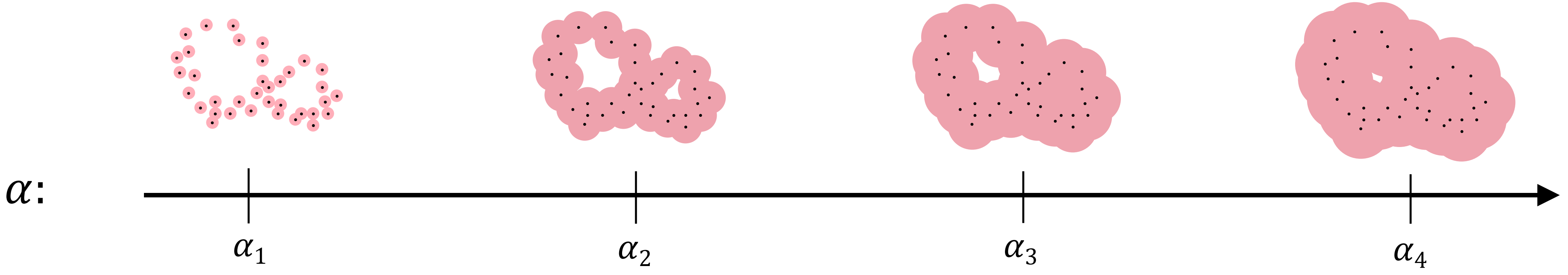
PD for Filtration

- Eventually, we will show that, PDs can be formally defined on both a "growing space" (which is continuous) and a "filtration of complexes" (which is discrete).

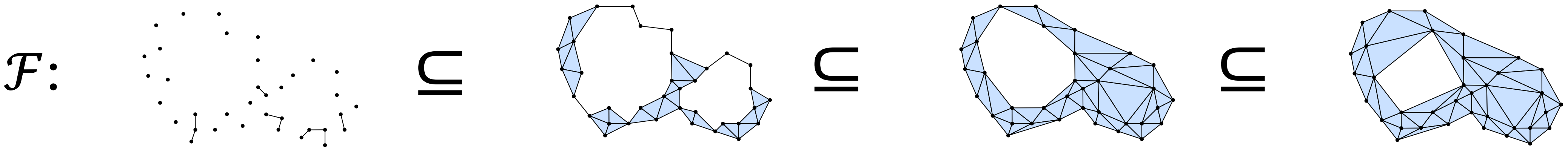


PD for Filtration

- Eventually, we will show that, PDs can be formally defined on both a "growing space" (which is continuous) and a "filtration of complexes" (which is discrete).
- We sometimes call the former one a "continuous" filtration and latter a "discrete" filtration (by default, a "filtration" without modifiers is always a discrete one).



"Continuous" filtration



"Discrete" filtration

PD for Filtration

- However, formally defining PD on a continuous or a discrete filtration needs a lot of mathematics (a lot of algebra, category theory, or quiver theory), which is beyond the scope of the course.
- So to understand the definition of a PD, we shall see how to compute a PD on a discrete filtration.
- Things can get a bit technical from now on.

Simplex-wise Filtration

- For computing persistence diagram, we focus on a special type of filtration.
- **Definition**: A **simplex-wise filtration** is a filtration such that each consecutive complexes differ by only a single simplex, i.e., in

$$\mathcal{F}:\ \emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

for each inclusion $K_{i-1} \subseteq K_i$, we have that K_i is derived from K_{i-1} by inserting a single simplex typically denoted σ_i.

Simplex-wise Filtration

- For computing persistence diagram, we focus on a special type of filtration.
- **Definition**: A **simplex-wise filtration** is a filtration such that each consecutive complexes differ by only a single simplex, i.e., in

$$\mathcal{F}\colon \emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

for each inclusion $K_{i-1} \subseteq K_i$, we have that K_i is derived from K_{i-1} by inserting a single simplex typically denoted σ_i.

- Because of the constructions, we can also consider a simplex-wise filtration

$$\mathcal{F} : \varnothing = K_0 \xhookrightarrow{\sigma_1} K_1 \xhookrightarrow{\sigma_2} \cdots \xhookrightarrow{\sigma_{m-1}} K_{m-1} \xhookrightarrow{\sigma_m} K_m = K$$

as a sequence of simplices $\sigma_0, \sigma_1, \ldots, \sigma_{m-1}$ inserted one by one following the order.

Simplex-wise Filtration

- For computing persistence diagram, we focus on a special type of filtration.
- **Definition**: A **simplex-wise filtration** is a filtration such that each consecutive complexes differ by only a single simplex, i.e., in

$$\mathcal{F}\colon \emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m = K$$

for each inclusion $K_{i-1} \subseteq K_i$, we have that K_i is derived from K_{i-1} by inserting a single simplex typically denoted σ_i.

- Because of the constructions, we can also consider a simplex-wise filtration

$$\mathcal{F} : \emptyset = K_0 \xhookrightarrow{\sigma_1} K_1 \xhookrightarrow{\sigma_2} \cdots \xhookrightarrow{\sigma_{m-1}} K_{m-1} \xhookrightarrow{\sigma_m} K_m = K$$

as a sequence of simplices $\sigma_0, \sigma_1, \dots, \sigma_{m-1}$ inserted one by one following the order.

- **Fact**: Each general filtration (not necessarily simplex-wise) can be made into a simplex-wise one by padding additional complexes (or expanding the inclusions)

Simplex-wise Filtration

- K_0 to K_1: insert vertices t and u and edge tu
- K_1 to K_2: insert edge st
- K_2 to K_3: insert edge su
- K_3 to K_4: insert triangle stu

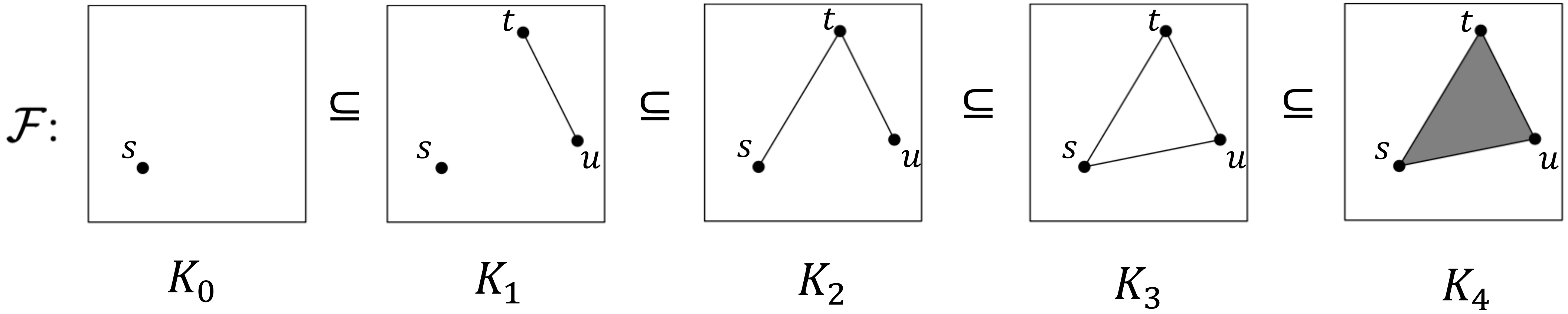
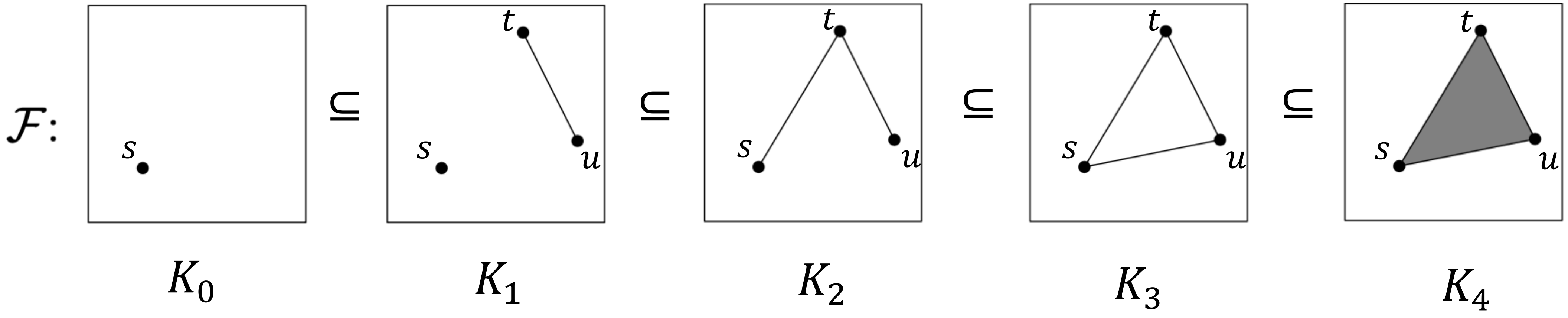


Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

Simplex-wise Filtration

- K_0 to K_1: insert vertices t and u and edge tu
- K_1 to K_2: insert edge st
- K_2 to K_3: insert edge su
- K_3 to K_4: insert triangle stu



- To convert to simplex-wise, only need to add an empty complex at the beginning and insert two additional complexes between K_0 to K_1.

Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

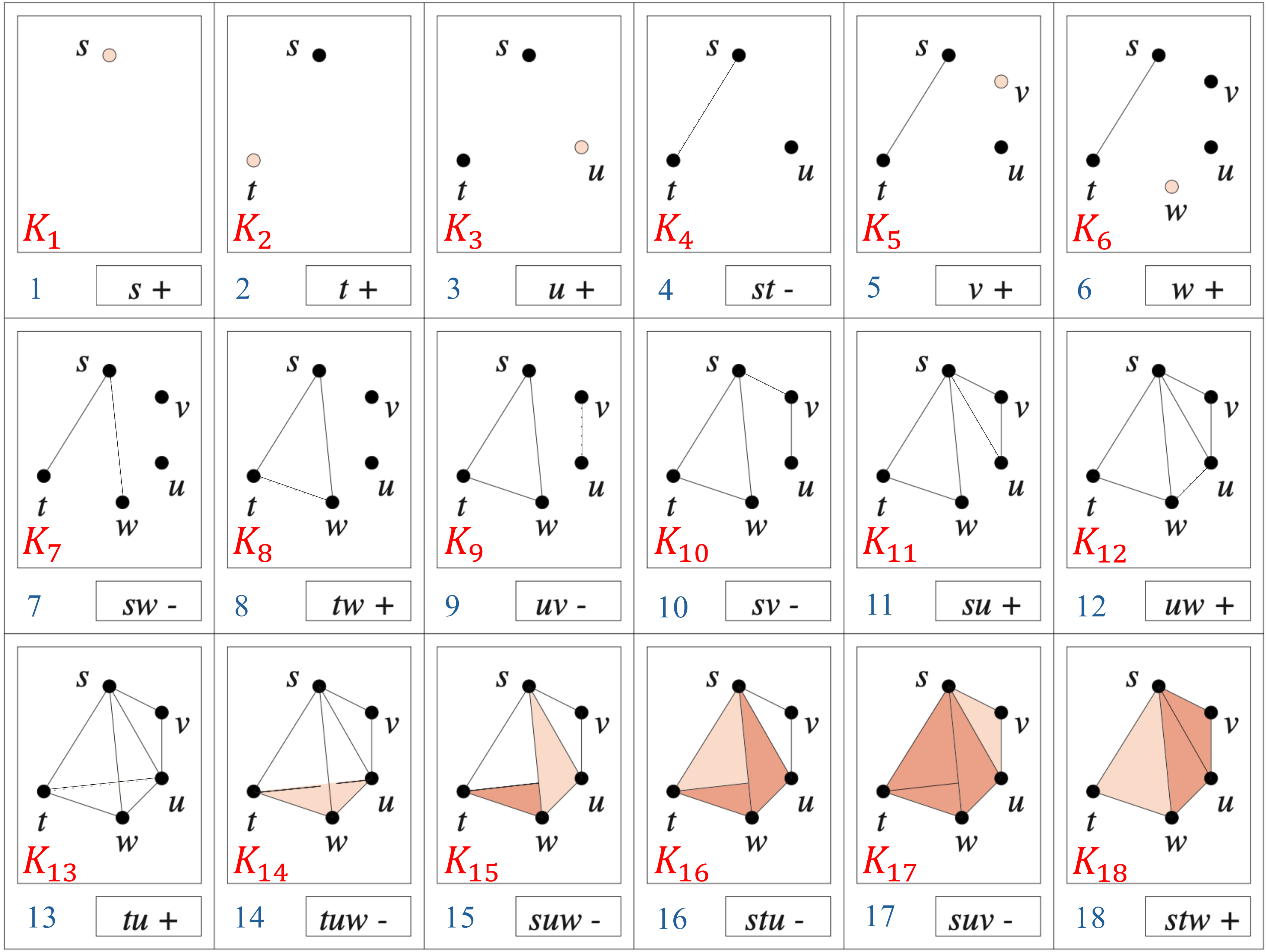


Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

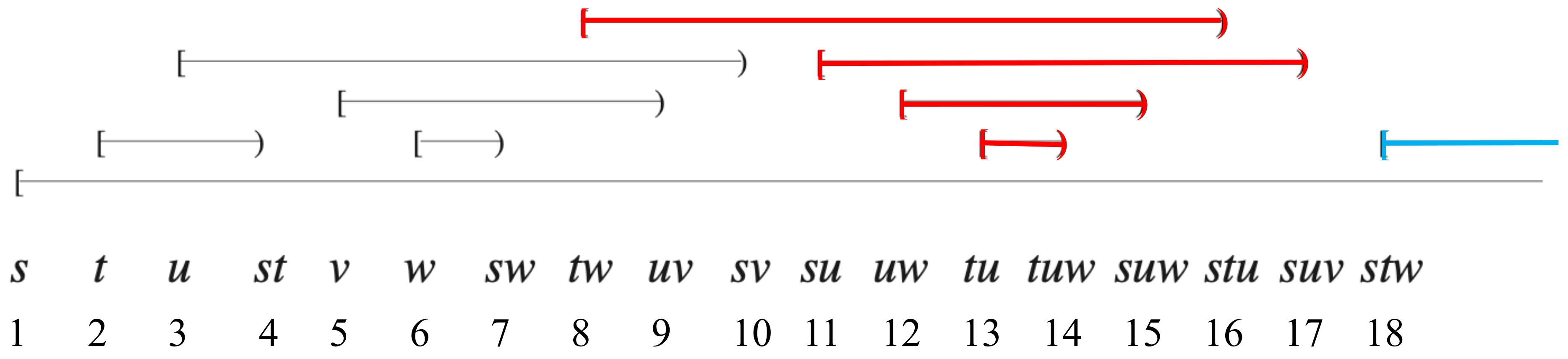
Algorithm

- Notice that the input filtration $\mathcal{F}$ must be simplex-wise

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

set each σ_i in $\mathcal{F}$ as "unpaired"
$\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
for $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
    $z = \partial(\sigma_i)$
    while $z \neq 0$ **do**
        let σ_j be the simplex with maximum index in z
        if σ_j is unpaired **then** break
        $z = z + \zeta(\sigma_j)$
    if $z \neq 0$ **then**
        pair σ_j with σ_i and set σ_j, σ_i as "paired"
        $\zeta(\sigma_j) = z$
        $p =$ dimension of σ_j
        add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
for each each unpaired σ_i **do**
    $p =$ dimension of σ_i
    add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

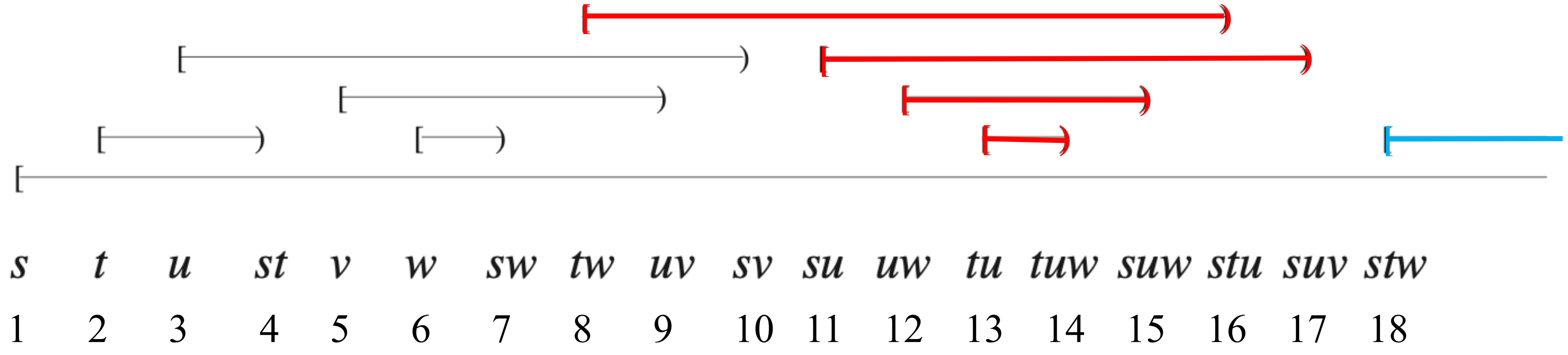
Resulting PD



- Black: PD_0
- Red: PD_1
- Blue: PD_2

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

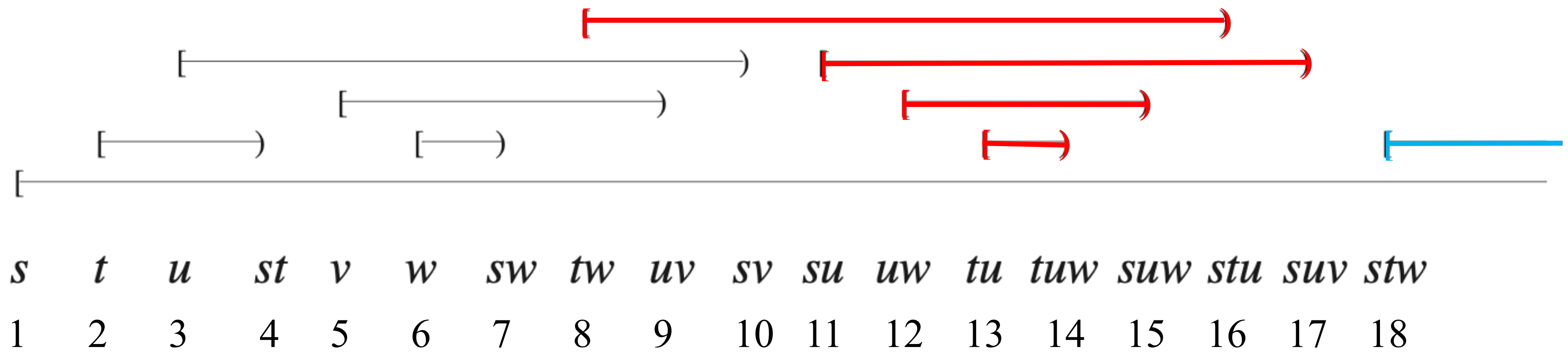
Resulting PD



- Notice: instead of drawing each pair of birth / death as a point on 2D plane, we just let each pair of birth and death form an interval, indicating the "time" in which a certain homology hole persists (will see examples later)

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

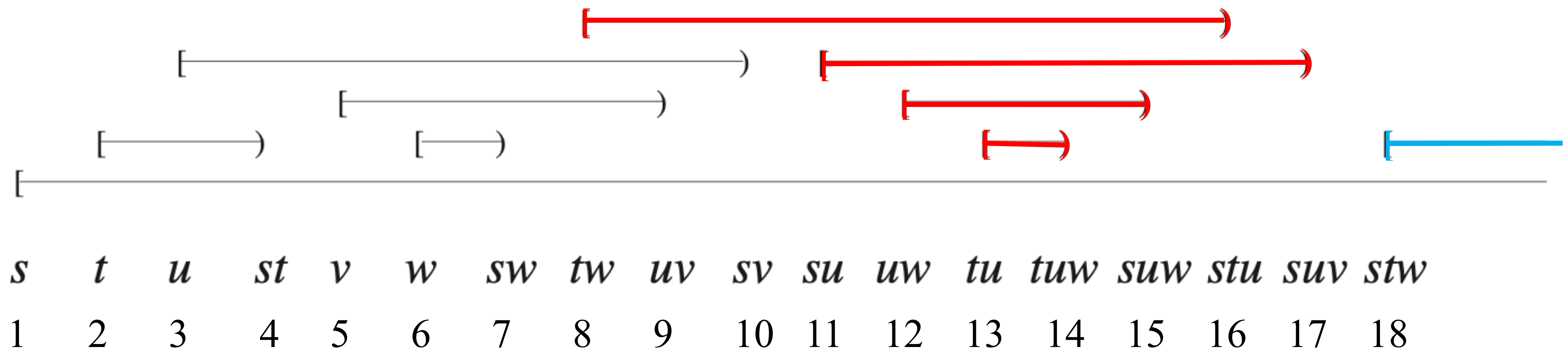
Resulting PD



- Notice: instead of drawing each pair of birth / death as a point on 2D plane, we just let each pair of birth and death form an interval, indicating the "time" in which a certain homology hole persists (will see examples later)
- The above is also called the **persistence barcode**

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

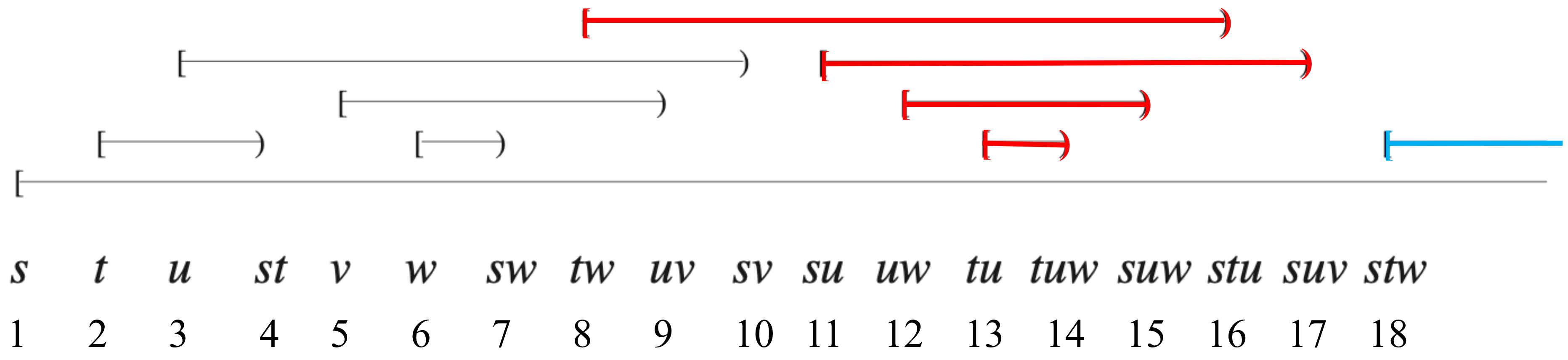
Resulting PD



- Notice: instead of drawing each pair of birth / death as a point on 2D plane, we just let each pair of birth and death form an interval, indicating the "time" in which a certain homology hole persists (will see examples later)
- The above is also called the **persistence barcode**
- So persistence barcodes and persistence diagrams are just the same things displayed in different ways (we sometimes also use the two terms interchangeably)

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

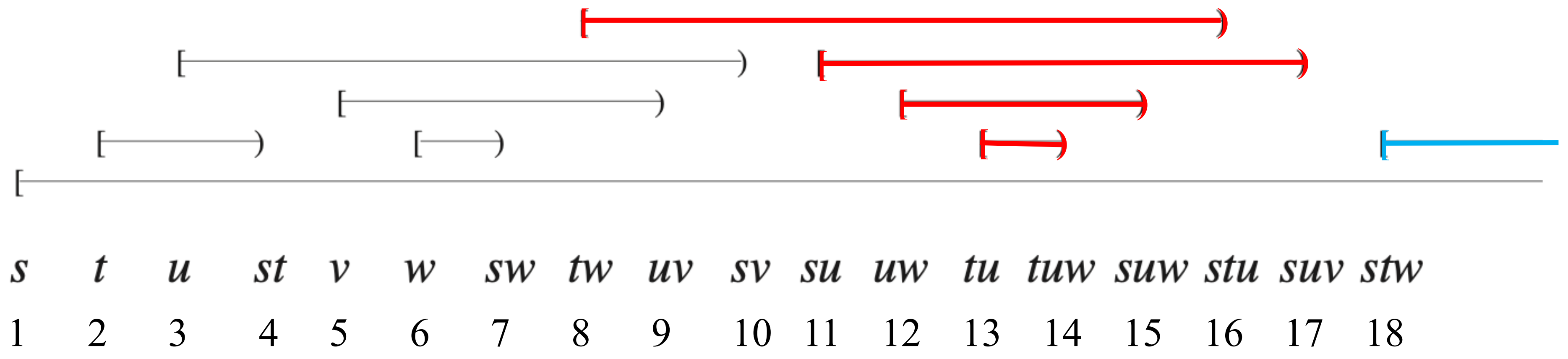
Resulting PD



- Notice: instead of drawing each pair of birth / death as a point on 2D plane, we just let each pair of birth and death form an interval, indicating the "time" in which a certain homology hole persists (will see examples later)
- The above is also called the **persistence barcode**
- So persistence barcodes and persistence diagrams are just the same things displayed in different ways (we sometimes also use the two terms interchangeably)
- Also notice: In persistence barcode, we always draw each interval as left-closed, right open (there is a technical reason for this but explaining this a little beyond scope)

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

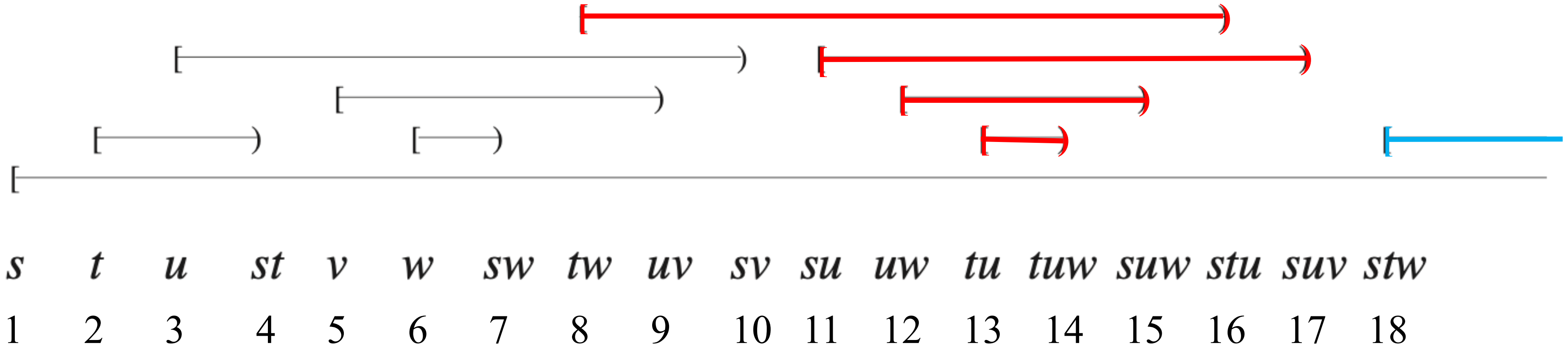
Resulting PD



- We also notice that the cycle recorded in the "ζ table" indeed captures the homology hole born and died with a birth-death interval in the barcode (point in the PD)

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

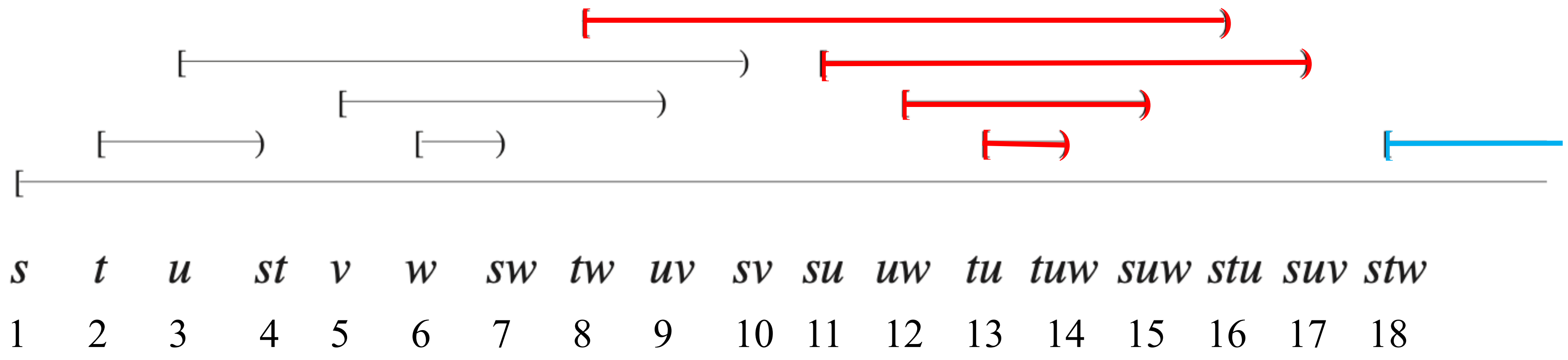
Resulting PD



- We also notice that the cycle recorded in the "ζ table" indeed captures the homology hole born and died with a birth-death interval in the barcode (point in the PD)
- i.e., for an interval $[b, d)$, $\zeta[\sigma_b]$ represents the homology feature born at the index b and dying at index d.

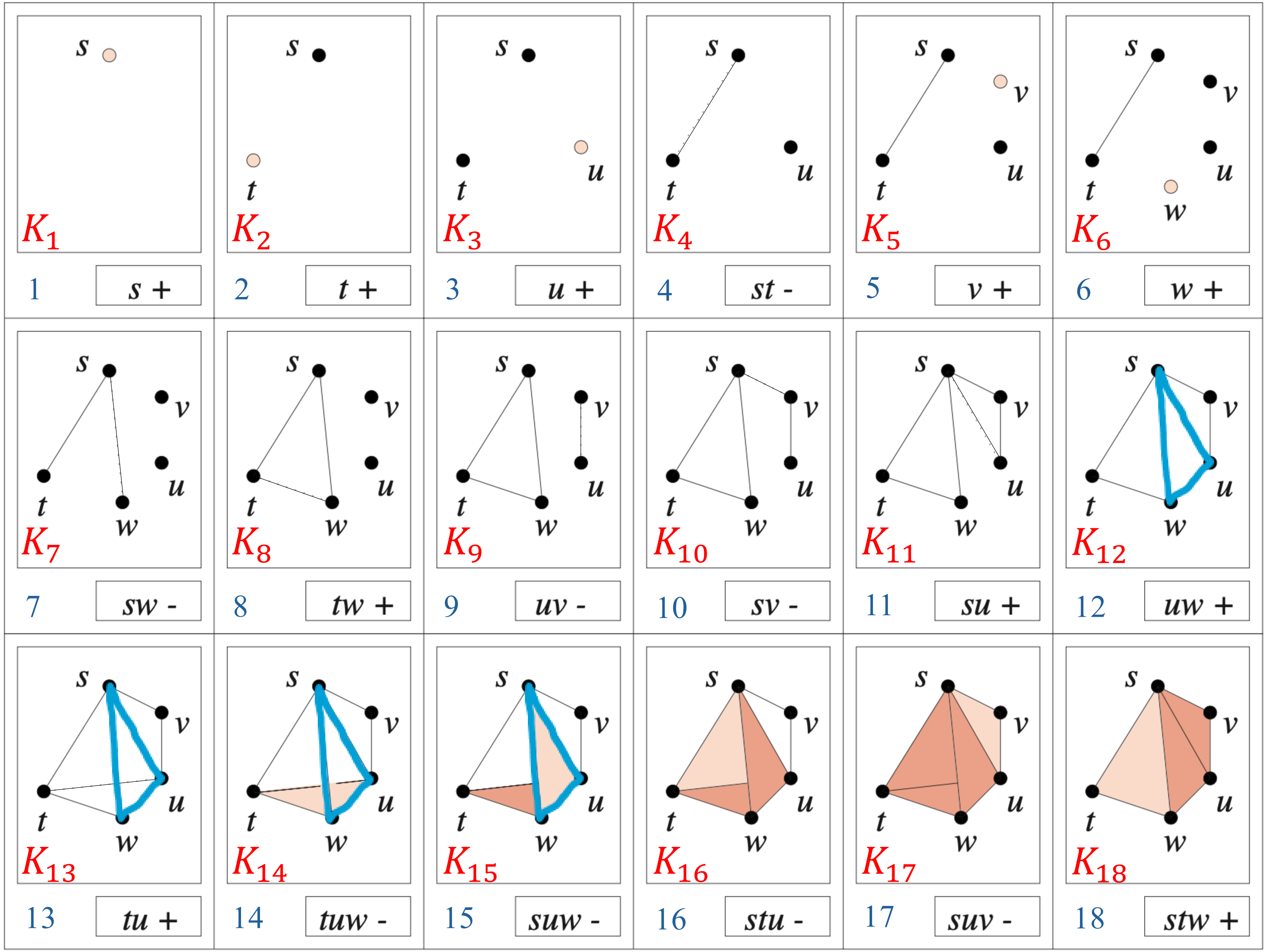
Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

Resulting PD



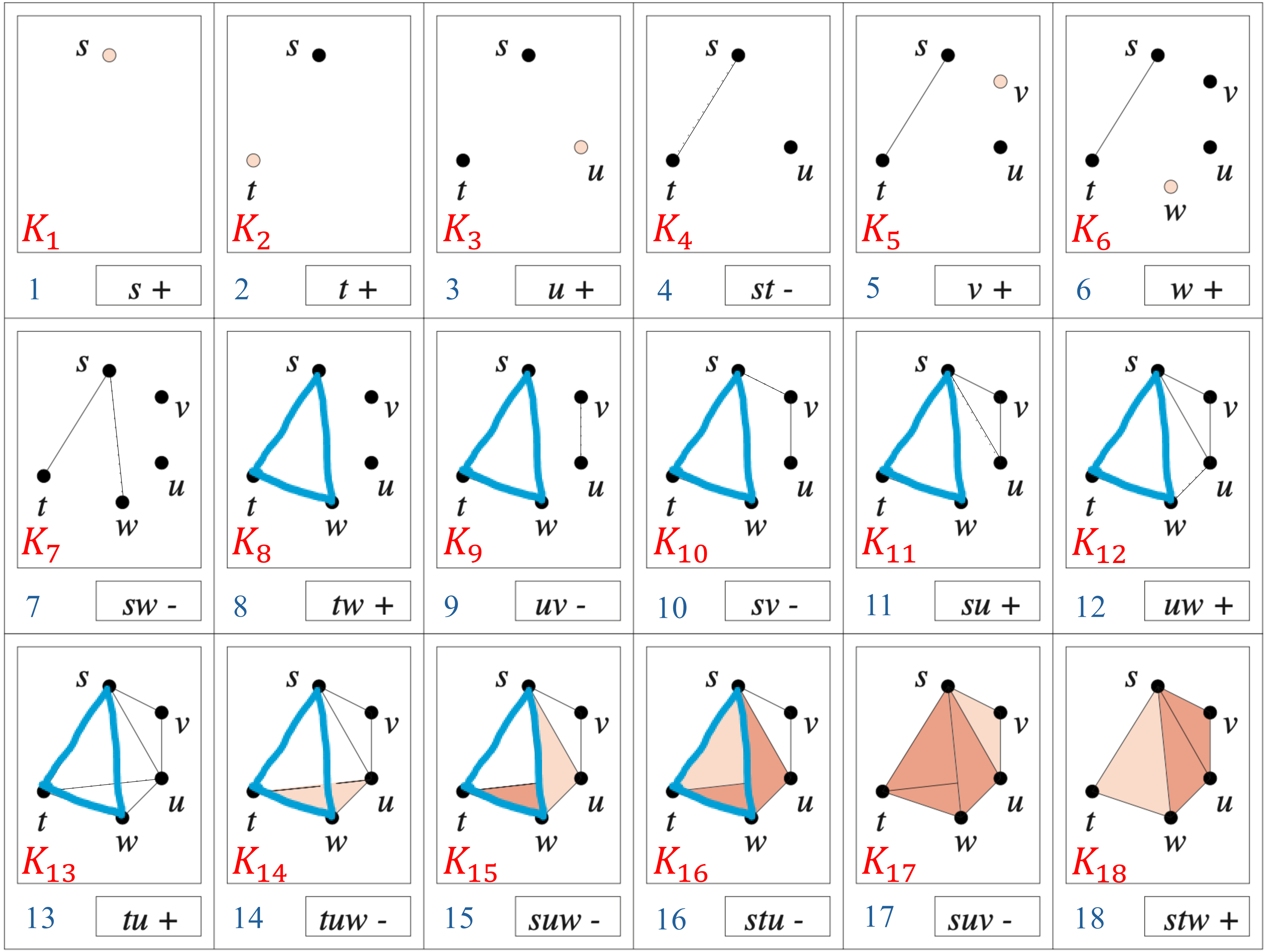
- We also notice that the cycle recorded in the "ζ table" indeed captures the homology hole born and died with a birth-death interval in the barcode (point in the PD)
- i.e., for an interval $[b, d)$, $\zeta[\sigma_b]$ represents the homology feature born at the index b and dying at index d.
- This $\zeta[\sigma_b]$ is also called the **representative** for the interval $[b, d)$.

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.



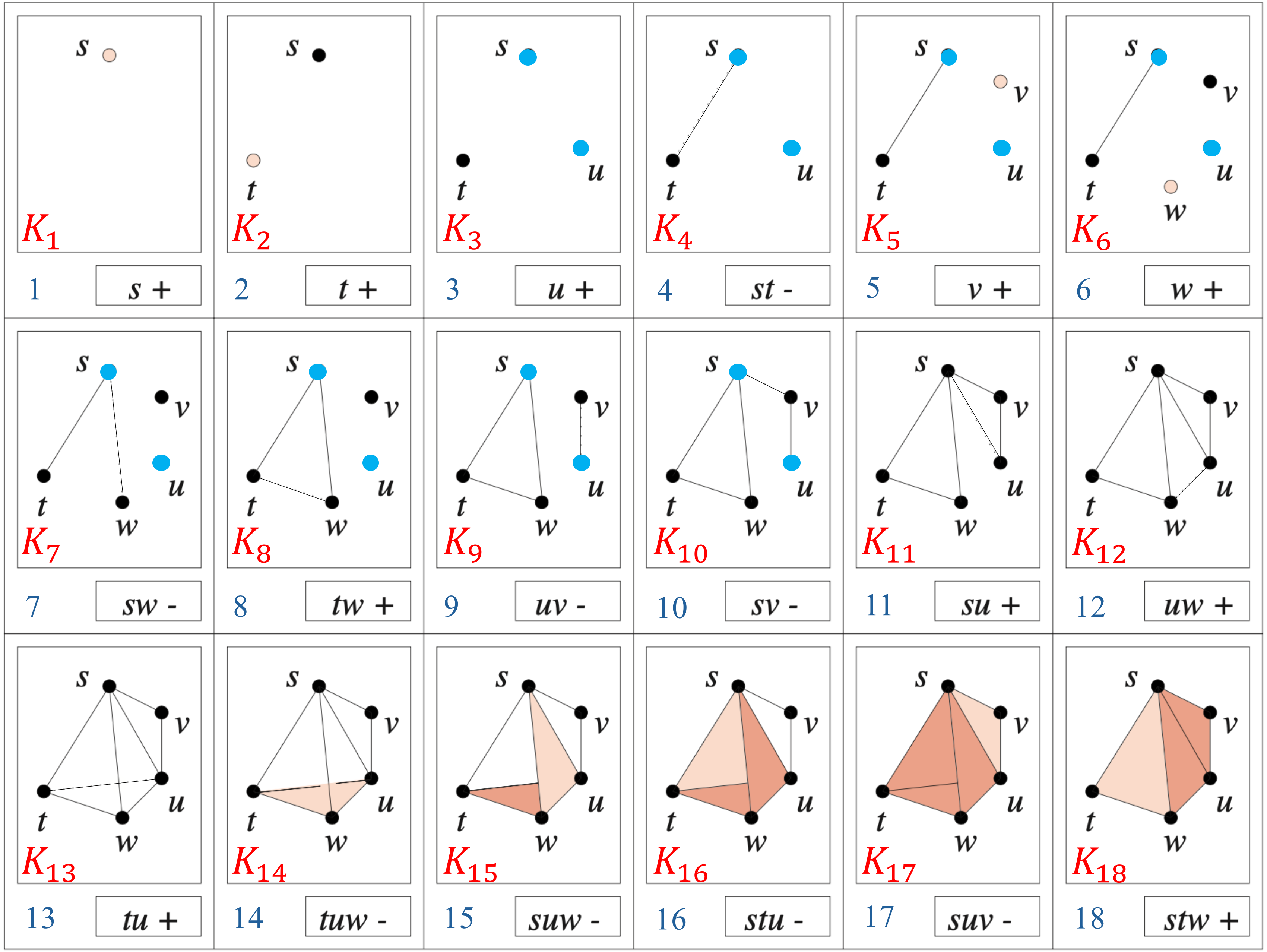
1d hole captured by interval $[12,15) \in PD_1$

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.



1d hole captured by interval $[8,16) \in PD_1$

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.



- 0d hole captured by interval $[3,10) \in PD_0$, which is the gap between s and u.
- The gap disappears when the two points become connected

Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

More interpretations of the algorithm:

- When processing each σ_i, if the while loop ends with $z = 0$, then the simplex σ_i is called **positive**

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

set each σ_i in $\mathcal{F}$ as "unpaired"
$\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
for $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
    $z = \partial(\sigma_i)$
    while $z \neq 0$ **do**
        let σ_j be the simplex with maximum index in z
        if σ_j is unpaired **then** break
        $z = z + \zeta(\sigma_j)$
    if $z \neq 0$ **then**
        pair σ_j with σ_i and set σ_j, σ_i as "paired"
        $\zeta(\sigma_j) = z$
        $p =$ dimension of σ_j
        add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
for each each unpaired σ_i **do**
    $p =$ dimension of σ_i
    add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

More interpretations of the algorithm:

- When processing each σ_i, if the while loop ends with $z = 0$, then the simplex σ_i is called **positive**
- It means that inserting σ_i creates a new homology hole

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

set each σ_i in $\mathcal{F}$ as "unpaired"
$\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
for $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
    $z = \partial(\sigma_i)$
    while $z \neq 0$ **do**
        let σ_j be the simplex with maximum index in z
        if σ_j is unpaired **then** break
        $z = z + \zeta(\sigma_j)$
    if $z \neq 0$ **then**
        pair σ_j with σ_i and set σ_j, σ_i as "paired"
        $\zeta(\sigma_j) = z$
        $p =$ dimension of σ_j
        add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
for each each unpaired σ_i **do**
    $p =$ dimension of σ_i
    add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

- E.g., inserting $\sigma_8 = tw$ creates the blue 1d hole

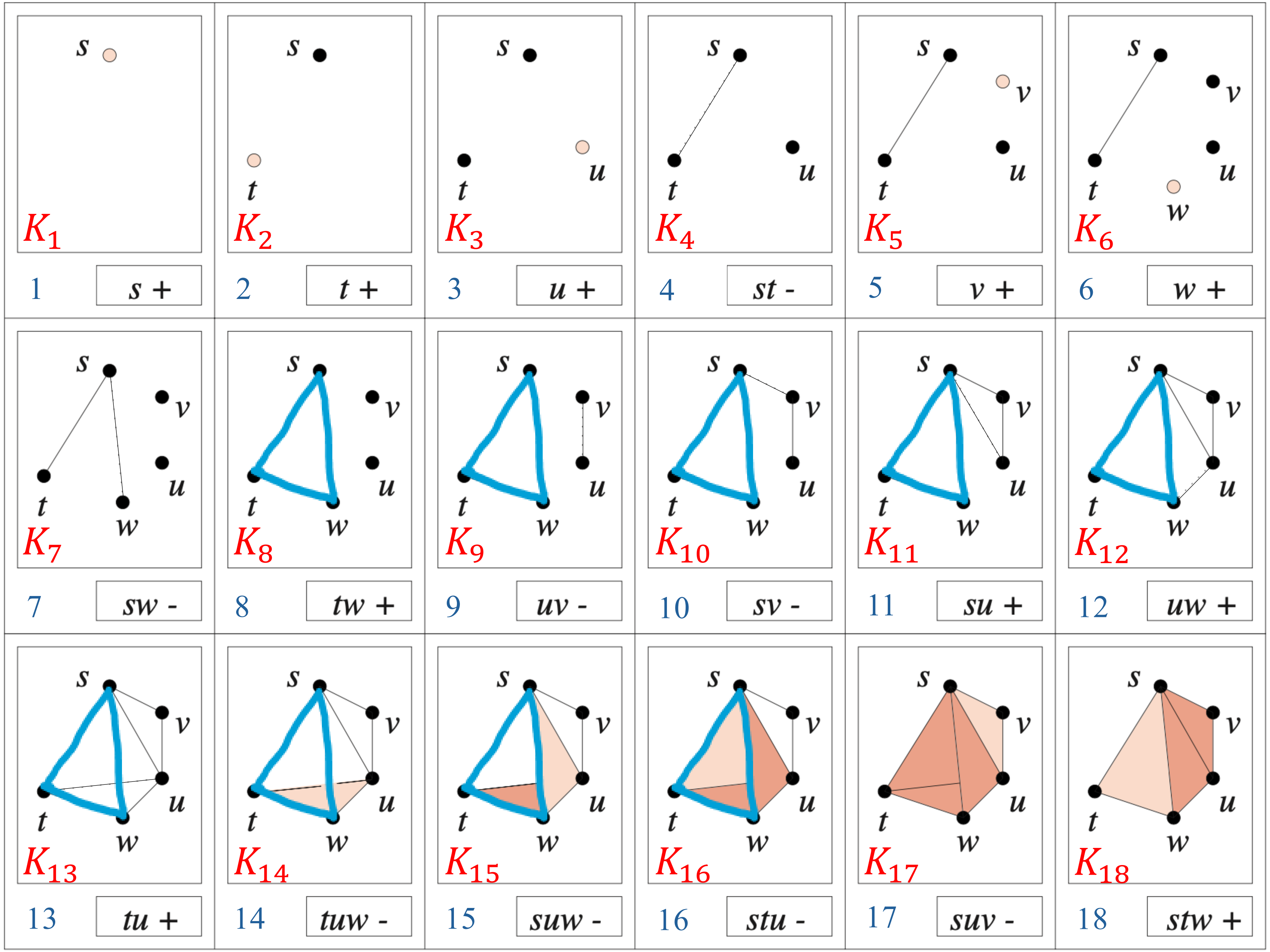


Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

- If the while loop ends with $z \neq 0$, then the simplex σ_i is called **negative**

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

set each σ_i in $\mathcal{F}$ as "unpaired"
$\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
for $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
  $z = \partial(\sigma_i)$
  while $z \neq 0$ **do**
    let σ_j be the simplex with maximum index in z
    if σ_j is unpaired **then** break
    $z = z + \zeta(\sigma_j)$
  if $z \neq 0$ **then**
    pair σ_j with σ_i and set σ_j, σ_i as "paired"
    $\zeta(\sigma_j) = z$
    $p =$ dimension of σ_j
    add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
for each each unpaired σ_i **do**
  $p =$ dimension of σ_i
  add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

- If the while loop ends with $z \neq 0$, then the simplex σ_i is called **negative**
- It means that inserting σ_i creates a homology hole die (becomes trivial)

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

set each σ_i in $\mathcal{F}$ as "unpaired"
$\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
for $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
    $z = \partial(\sigma_i)$
    while $z \neq 0$ **do**
        let σ_j be the simplex with maximum index in z
        if σ_j is unpaired **then** break
        $z = z + \zeta(\sigma_j)$
    if $z \neq 0$ **then**
        pair σ_j with σ_i and set σ_j, σ_i as "paired"
        $\zeta(\sigma_j) = z$
        $p =$ dimension of σ_j
        add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
for each each unpaired σ_i **do**
    $p =$ dimension of σ_i
    add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

- E.g., inserting $\sigma_{16} = stu$ kills the blue 1d hole

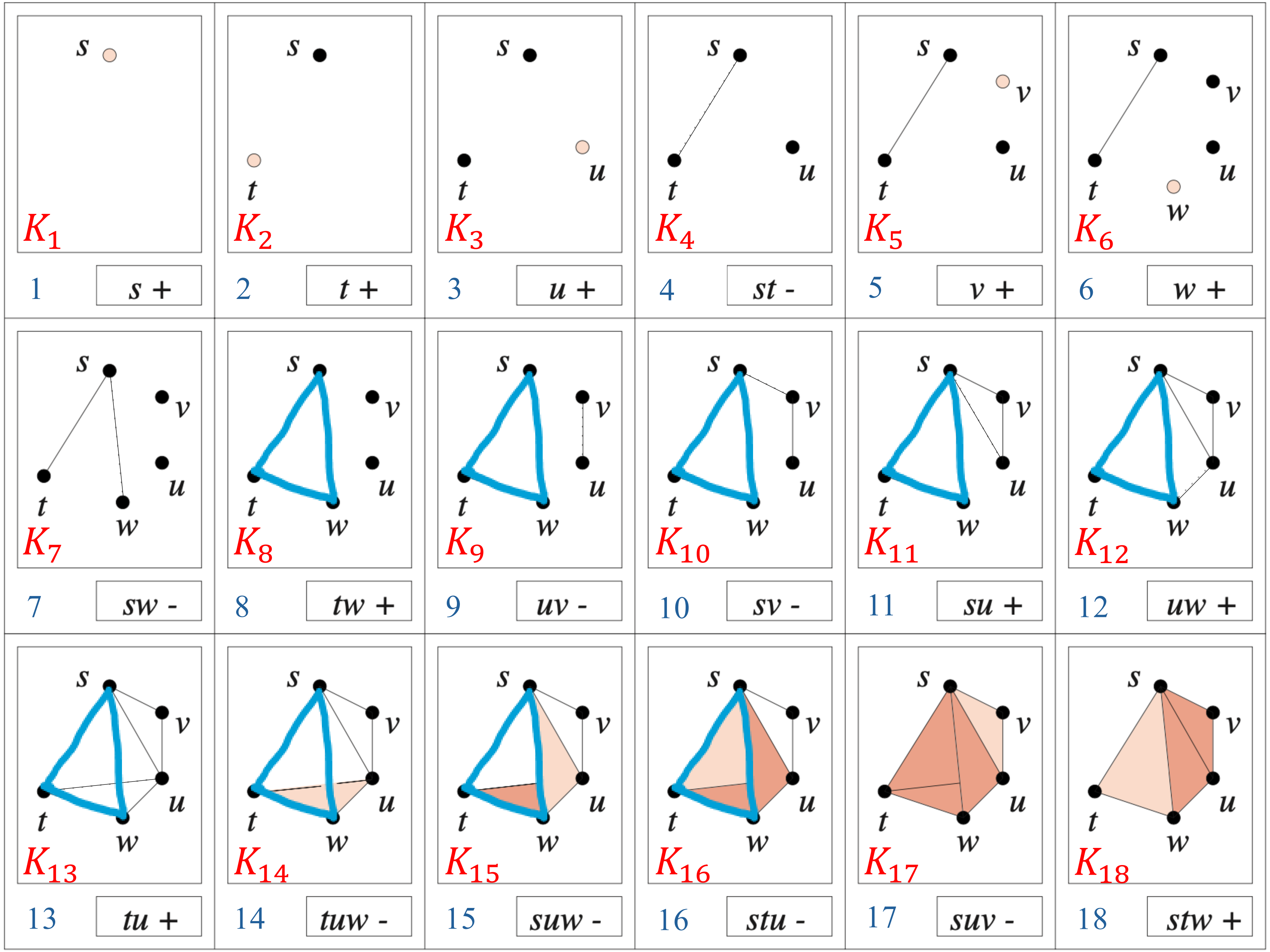


Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

We have that line 10 in the algorithm is always pairing

- a positive simplex σ_j

with

- a negative simplex σ_i

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

1: set each σ_i in $\mathcal{F}$ as "unpaired"
2: $\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
3: **for** $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
4:     $z = \partial(\sigma_i)$
5:     **while** $z \neq 0$ **do**
6:         let σ_j be the simplex with maximum index in z
7:         **if** σ_j is unpaired **then** break
8:         $z = z + \zeta(\sigma_j)$
9:     **if** $z \neq 0$ **then**
10:         pair σ_j with σ_i and set σ_j, σ_i as "paired"
11:         $\zeta(\sigma_j) = z$
12:         $p =$ dimension of σ_j
13:         add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
14: **for each** each unpaired σ_i **do**
15:     $p =$ dimension of σ_i
16:     add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

The algorithm takes $O(m^3)$ time:

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

set each σ_i in $\mathcal{F}$ as "unpaired"
$\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
for $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
    $z = \partial(\sigma_i)$
    while $z \neq 0$ **do**
        let σ_j be the simplex with maximum index in z
        if σ_j is unpaired **then** break
        $z = z + \zeta(\sigma_j)$
    if $z \neq 0$ **then**
        pair σ_j with σ_i and set σ_j, σ_i as "paired"
        $\zeta(\sigma_j) = z$
        $p =$ dimension of σ_j
        add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
for each each unpaired σ_i **do**
    $p =$ dimension of σ_i
    add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

The algorithm takes $O(m^3)$ time:

First of all, summing two cycles in line 8 takes $O(m)$ time

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

1: set each σ_i in $\mathcal{F}$ as "unpaired"
2: $\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
3: **for** $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
4:     $z = \partial(\sigma_i)$
5:     **while** $z \neq 0$ **do**
6:         let σ_j be the simplex with maximum index in z
7:         **if** σ_j is unpaired **then** break
8:         $z = z + \zeta(\sigma_j)$
9:     **if** $z \neq 0$ **then**
10:         pair σ_j with σ_i and set σ_j, σ_i as "paired"
11:         $\zeta(\sigma_j) = z$
12:         $p =$ dimension of σ_j
13:         add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
14: **for each** each unpaired σ_i **do**
15:     $p =$ dimension of σ_i
16:     add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

The algorithm takes $O(m^3)$ time:

First of all, summing two cycles in line 8 takes $O(m)$ time

- You could either represent a cycle z (a set of simplices) as a 0-1 list where the i-th item is 1 iff σ_i is in z

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathsf{PD}_p(\mathcal{F})$, for each dimension p

1: set each σ_i in $\mathcal{F}$ as "unpaired"
2: $\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
3: **for** $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
4: $\quad z = \partial(\sigma_i)$
5: $\quad$ **while** $z \neq 0$ **do**
6: $\quad\quad$ let σ_j be the simplex with maximum index in z
7: $\quad\quad$ **if** σ_j is unpaired **then** break
8: $\quad\quad z = z + \zeta(\sigma_j)$
9: $\quad$ **if** $z \neq 0$ **then**
10: $\quad\quad$ pair σ_j with σ_i and set σ_j, σ_i as "paired"
11: $\quad\quad \zeta(\sigma_j) = z$
12: $\quad\quad p =$ dimension of σ_j
13: $\quad\quad$ add (j, i) to $\mathsf{PD}_p(\mathcal{F})$
14: **for each** each unpaired σ_i **do**
15: $\quad p =$ dimension of σ_i
16: $\quad$ add (i, ∞) to $\mathsf{PD}_p(\mathcal{F})$

The algorithm takes $O(m^3)$ time:

First of all, summing two cycles in line 8 takes $O(m)$ time

- You could either represent a cycle z (a set of simplices) as a 0-1 list where the i-th item is 1 iff σ_i is in z
- Or represent z as a sorted list of integers such that i is in the list iff σ_i is in z

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\text{PD}_p(\mathcal{F})$, for each dimension p

1: set each σ_i in $\mathcal{F}$ as "unpaired"
2: $\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
3: **for** $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
4:     $z = \partial(\sigma_i)$
5:     **while** $z \neq 0$ **do**
6:         let σ_j be the simplex with maximum index in z
7:         **if** σ_j is unpaired **then** break
8:         $z = z + \zeta(\sigma_j)$
9:     **if** $z \neq 0$ **then**
10:         pair σ_j with σ_i and set σ_j, σ_i as "paired"
11:         $\zeta(\sigma_j) = z$
12:         $p =$ dimension of σ_j
13:         add (j, i) to $\text{PD}_p(\mathcal{F})$
14: **for each** each unpaired σ_i **do**
15:     $p =$ dimension of σ_i
16:     add (i, ∞) to $\text{PD}_p(\mathcal{F})$

The algorithm takes $O(m^3)$ time:

First of all, summing two cycles in line 8 takes $O(m)$ time

- You could either represent a cycle z (a set of simplices) as a 0-1 list where the i-th item is 1 iff σ_i is in z
- Or represent z as a sorted list of integers such that i is in the list iff σ_i is in z

In the worst case, both inner and outer loop iterates $O(m)$ time, and hence $O(m^3)$ oveall

Input: a filtration $\mathcal{F}$ as a sequence of simplices $\sigma_1, \sigma_2, \ldots, \sigma_m$
Output: p-th PD of $\mathcal{F}$, $\mathrm{PD}_p(\mathcal{F})$, for each dimension p

1: set each σ_i in $\mathcal{F}$ as "unpaired"
2: $\zeta =$ a table mapping each σ_i to a cycle $\zeta(\sigma_i)$ initially undefined
3: **for** $\sigma_i = \sigma_1, \sigma_2, \ldots, \sigma_m$ **do**
4:     $z = \partial(\sigma_i)$
5:     **while** $z \neq 0$ **do**
6:         let σ_j be the simplex with maximum index in z
7:         **if** σ_j is unpaired **then** break
8:         $z = z + \zeta(\sigma_j)$
9:     **if** $z \neq 0$ **then**
10:         pair σ_j with σ_i and set σ_j, σ_i as "paired"
11:         $\zeta(\sigma_j) = z$
12:         $p =$ dimension of σ_j
13:         add (j, i) to $\mathrm{PD}_p(\mathcal{F})$
14: **for each** each unpaired σ_i **do**
15:     $p =$ dimension of σ_i
16:     add (i, ∞) to $\mathrm{PD}_p(\mathcal{F})$

PD for General Filtration

- Recall part of the reason we introduce the previous algorithm for computing PD is to formally define PD on a discrete filtration

PD for General Filtration

- Recall part of the reason we introduce the previous algorithm for computing PD is to formally define PD on a discrete filtration
- But the algorithm only takes a simplex-wise filtration as input, so technically we only defined PD on a simplex-wise filtration

PD for General Filtration

- Recall part of the reason we introduce the previous algorithm for computing PD is to formally define PD on a discrete filtration
- But the algorithm only takes a simplex-wise filtration as input, so technically we only defined PD on a simplex-wise filtration
- We shall now define PD on a general filtration not necessarily simplex-wise

PD for General Filtration

- Recall part of the reason we introduce the previous algorithm for computing PD is to formally define PD on a discrete filtration
- But the algorithm only takes a simplex-wise filtration as input, so technically we only defined PD on a simplex-wise filtration
- We shall now define PD on a general filtration not necessarily simplex-wise
- The process is as follows:
  1. "**Expand**" the general filtration $\mathcal{F}$ into a simplex-wise one $\mathcal{F}'$
     - Aka, for an inclusion in $\mathcal{F}$ which inserts k number of simplices, we convert it into k inclusions each inserting a single simplex

PD for General Filtration

- Recall part of the reason we introduce the previous algorithm for computing PD is to formally define PD on a discrete filtration
- But the algorithm only takes a simplex-wise filtration as input, so technically we only defined PD on a simplex-wise filtration
- We shall now define PD on a general filtration not necessarily simplex-wise
- The process is as follows:
  1. "**Expand**" the general filtration $\mathcal{F}$ into a simplex-wise one $\mathcal{F}'$
     - Aka, for an inclusion in $\mathcal{F}$ which inserts k number of simplices, we convert it into k inclusions each inserting a single simplex
  2. Compute $\mathrm{PD}(\mathcal{F}')$

PD for General Filtration

- Recall part of the reason we introduce the previous algorithm for computing PD is to formally define PD on a discrete filtration
- But the algorithm only takes a simplex-wise filtration as input, so technically we only defined PD on a simplex-wise filtration
- We shall now define PD on a general filtration not necessarily simplex-wise
- The process is as follows:
  1. "**Expand**" the general filtration $\mathcal{F}$ into a simplex-wise one $\mathcal{F}'$
     - Aka, for an inclusion in $\mathcal{F}$ which inserts k number of simplices, we convert it into k inclusions each inserting a single simplex
  2. Compute $\mathrm{PD}(\mathcal{F}')$
  3. Convert $\mathrm{PD}(\mathcal{F}')$ into $\mathrm{PD}(\mathcal{F})$ by "**contracting**" each interval $\mathrm{PD}(\mathcal{F}')$ based on the correspondence between

PD for General Filtration

- Recall part of the reason we introduce the previous algorithm for computing PD is to formally define PD on a discrete filtration
- But the algorithm only takes a simplex-wise filtration as input, so technically we only defined PD on a simplex-wise filtration
- We shall now define PD on a general filtration not necessarily simplex-wise
- The process is as follows:
  1. "**Expand**" the general filtration $\mathcal{F}$ into a simplex-wise one $\mathcal{F}'$
     - Aka, for an inclusion in $\mathcal{F}$ which inserts k number of simplices, we convert it into k inclusions each inserting a single simplex
  2. Compute $\mathrm{PD}(\mathcal{F}')$
  3. Convert $\mathrm{PD}(\mathcal{F}')$ into $\mathrm{PD}(\mathcal{F})$ by "**contracting**" each interval $\mathrm{PD}(\mathcal{F}')$ based on the correspondence between
  4. During the contraction, some intervals in $\mathrm{PD}(\mathcal{F}')$ may disappear (birth and death coincide)

Expand general filtration to simplex wise

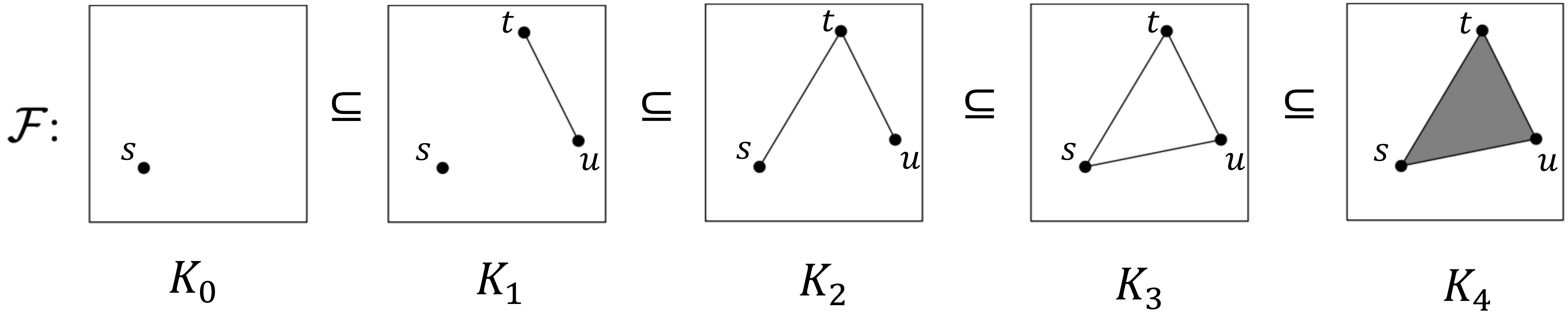


Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

Expand general filtration to simplex wise

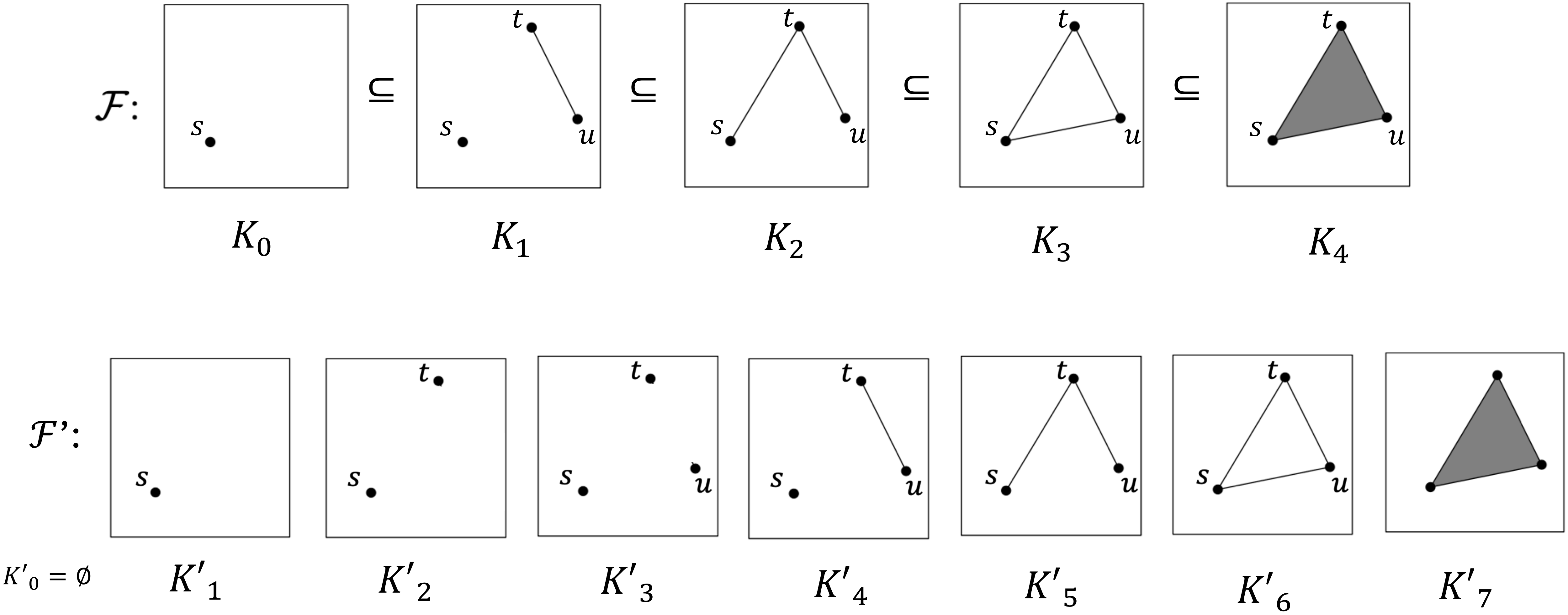


Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

Expand general filtration to simplex wise

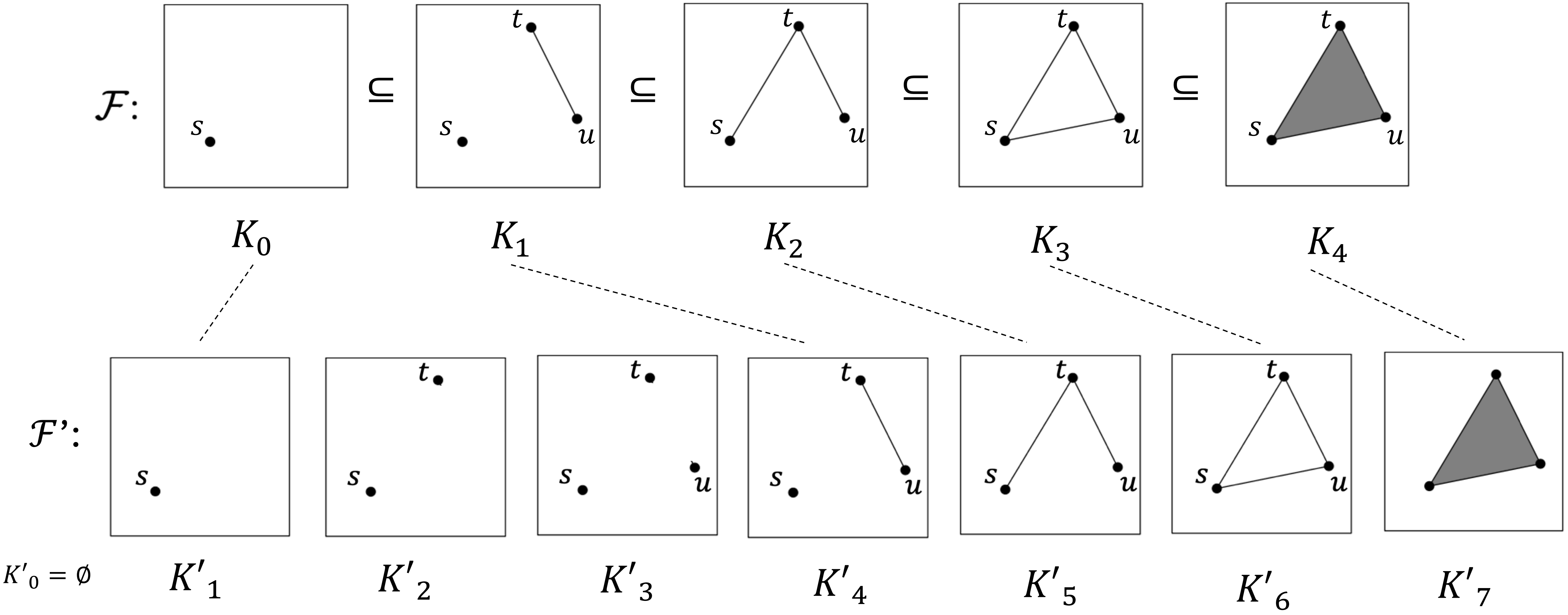


Image source: Patrick Schnider. Introduction to Topological Data Analysis Lecture Notes FS 2023

Expand general filtration to simplex wise

- Another interactive example for correspondence between a general filtration and its simplex-wise version: https://iuricichf.github.io/ICT/algorithm.html

PD for General Filtration

$$\mathcal{F} : K_0 \overset{}{\dashrightarrow} K_1 \dashrightarrow K_2 \dashrightarrow K_3 \dashrightarrow K_4 \dashrightarrow K_5 \dashrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow K'_3 \hookrightarrow K'_4 \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow K'_9 \hookrightarrow K'_{10} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

"Contracting" $[4,10) \in PD(\mathcal{F}')$ into one for $PD(\mathcal{F})$:

PD for General Filtration

$$\mathcal{F} : K_0 \dashrightarrow K_1 \dashrightarrow K_2 \dashrightarrow K_3 \dashrightarrow K_4 \dashrightarrow K_5 \dashrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow \boxed{K'_3 \hookrightarrow K'_4} \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow K'_9 \hookrightarrow K'_{10} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

"Contracting" $[4,10) \in PD(\mathcal{F}')$ into one for $PD(\mathcal{F})$:

- $[4,10) \in PD(\mathcal{F}')$ is born in K'_4, which is when go from K'_3 to K'_4 in $\mathcal{F}'$

PD for General Filtration

$$\mathcal{F} : K_0 \hookrightarrow \boxed{K_1 \hookrightarrow K_2} \hookrightarrow K_3 \hookrightarrow K_4 \hookrightarrow K_5 \hookrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow \boxed{K'_3 \hookrightarrow K'_4} \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow K'_9 \hookrightarrow K'_{10} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

"Contracting" $[4,10) \in PD(\mathcal{F}')$ into one for $PD(\mathcal{F})$:

- $[4,10) \in PD(\mathcal{F}')$ is born in K'_4, which is when go from K'_3 to K'_4 in $\mathcal{F}'$
- In $\mathcal{F}$, the homology feature is born when we go from K_1 to K_2, aka in K_2

PD for General Filtration

$$\mathcal{F} : K_0 \hookrightarrow \boxed{K_1 \hookrightarrow K_2} \hookrightarrow K_3 \hookrightarrow K_4 \hookrightarrow K_5 \hookrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow \boxed{K'_3 \hookrightarrow K'_4} \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow K'_9 \hookrightarrow K'_{10} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

"Contracting" $[4,10) \in PD(\mathcal{F}')$ into one for $PD(\mathcal{F})$:

- $[4,10) \in PD(\mathcal{F}')$ is born in K'_4, which is when go from K'_3 to K'_4 in $\mathcal{F}'$
- In $\mathcal{F}$, the homology feature is born when we go from K_1 to K_2, aka in K_2
- So the birth of the corresponding interval in $PD(\mathcal{F})$ is 2

PD for General Filtration

$$\mathcal{F} : K_0 \hookrightarrow K_1 \hookrightarrow K_2 \hookrightarrow K_3 \hookrightarrow K_4 \hookrightarrow K_5 \hookrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow K'_3 \hookrightarrow K'_4 \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow \boxed{K'_9 \hookrightarrow K'_{10}} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

"Contracting" $[4,10) \in PD(\mathcal{F}')$ into one for $PD(\mathcal{F})$:

- $[4,10) \in PD(\mathcal{F}')$ dies in K'_{10}, which specifically is when go from K'_9 to K'_{10}

PD for General Filtration

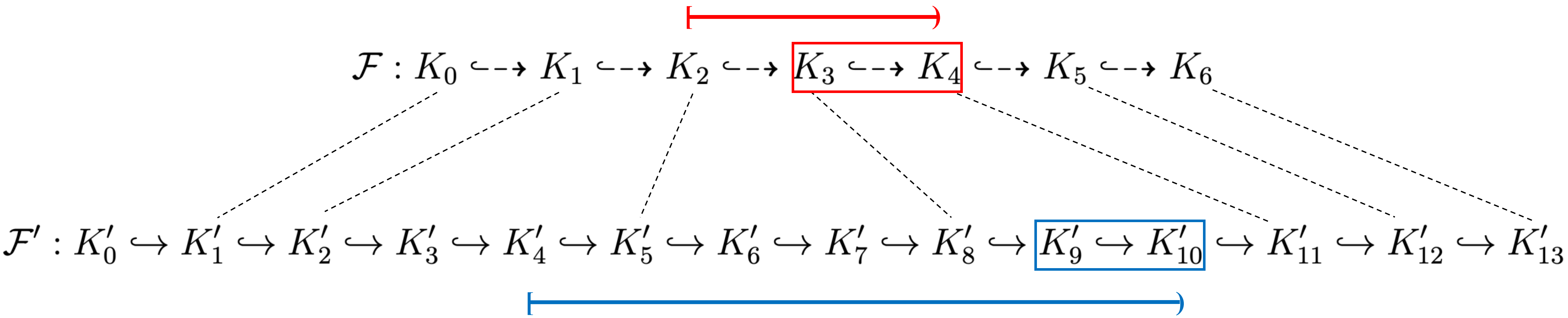
$$\mathcal{F} : K_0 \hookrightarrow K_1 \hookrightarrow K_2 \hookrightarrow \boxed{K_3 \hookrightarrow K_4} \hookrightarrow K_5 \hookrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow K'_3 \hookrightarrow K'_4 \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow \boxed{K'_9 \hookrightarrow K'_{10}} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

"Contracting" $[4,10) \in PD(\mathcal{F}')$ into one for $PD(\mathcal{F})$:

- $[4,10) \in PD(\mathcal{F}')$ dies in K'_{10}, which specifically is when go from K'_9 to K'_{10}
- In $\mathcal{F}$, the homology feature dies when we go from K_3 to K_4, aka in K_4

PD for General Filtration



"Contracting" $[4,10) \in PD(\mathcal{F}')$ into one for $PD(\mathcal{F})$:

- $[4,10) \in PD(\mathcal{F}')$ dies in K'_{10}, which specifically is when go from K'_9 to K'_{10}
- In $\mathcal{F}$, the homology feature dies when we go from K_3 to K_4, aka in K_4
- So the corresponding interval in $PD(\mathcal{F})$ is $[2,4)$

PD for General Filtration

$$\mathcal{F} : K_0 \hookrightarrow K_1 \hookrightarrow K_2 \hookrightarrow K_3 \hookrightarrow K_4 \hookrightarrow K_5 \hookrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow K'_3 \hookrightarrow K'_4 \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow K'_9 \hookrightarrow K'_{10} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

$[5,8) \in PD(\mathcal{F}')$ does not correspond to any interval in $PD(\mathcal{F})$:

- In $\mathcal{F}$, the homology feature is born in K_3 and dies also K_3 (so it's ephemeral)

PD for General Filtration

$$\mathcal{F} : K_0 \hookrightarrow K_1 \hookrightarrow K_2 \hookrightarrow K_3 \hookrightarrow K_4 \hookrightarrow K_5 \hookrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow K'_3 \hookrightarrow K'_4 \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow K'_9 \hookrightarrow K'_{10} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

Another way to view it:

- Consider the actual "indices" included in an interval in $PD(\mathcal{F}')$, e.g., $[4,10) \in PD(\mathcal{F}')$ contains indices 4,5,…,9

PD for General Filtration

$$\mathcal{F} : K_0 \dashrightarrow K_1 \dashrightarrow K_2 \dashrightarrow K_3 \dashrightarrow K_4 \dashrightarrow K_5 \dashrightarrow K_6$$

$$\mathcal{F}' : K'_0 \hookrightarrow K'_1 \hookrightarrow K'_2 \hookrightarrow K'_3 \hookrightarrow K'_4 \hookrightarrow K'_5 \hookrightarrow K'_6 \hookrightarrow K'_7 \hookrightarrow K'_8 \hookrightarrow K'_9 \hookrightarrow K'_{10} \hookrightarrow K'_{11} \hookrightarrow K'_{12} \hookrightarrow K'_{13}$$

Another way to view it:

- Consider the actual "indices" included in an interval in $PD(\mathcal{F}')$, e.g., $[4,10) \in PD(\mathcal{F}')$ contains indices 4,5,…,9
- Take the corresponding indices in $\mathcal{F}$ and get the left-closed, right-open interval for $PD(\mathcal{F})$, e.g., the corresponding indices for 4,5,…,9 in $\mathcal{F}$ are 2,3, so the interval in $PD(\mathcal{F})$ is $[2,4)$

- For the previous simplex-wise filtration, we can skip some complexes and renumber them

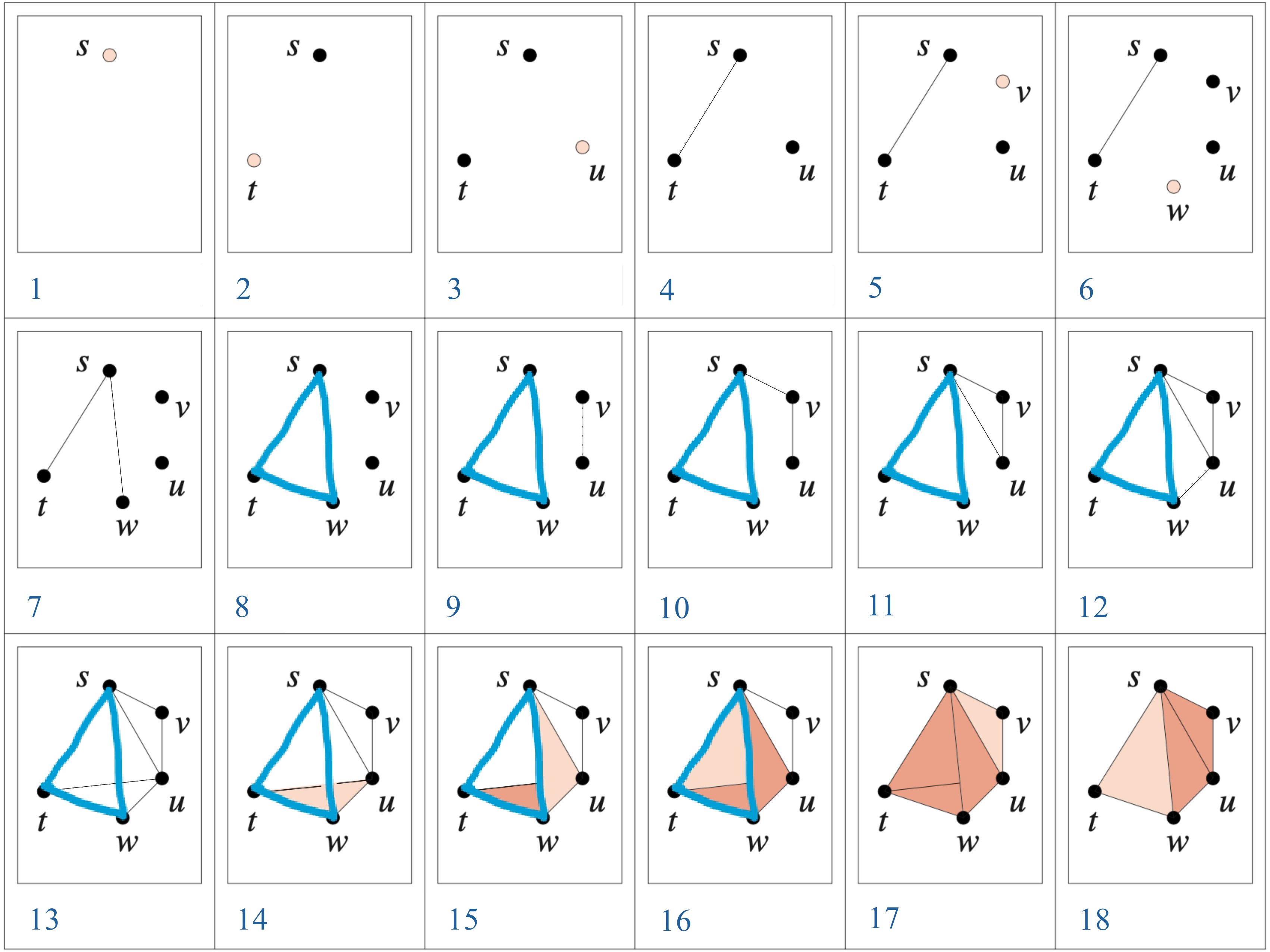


Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.

- For the previous simplex-wise filtration, we can skip some complexes and renumber them
- Then [8,16) in the simplex-wise filtration becomes [5,10) in the non-simplex-wise
- But they are essential "same" interval (representatives are the same)

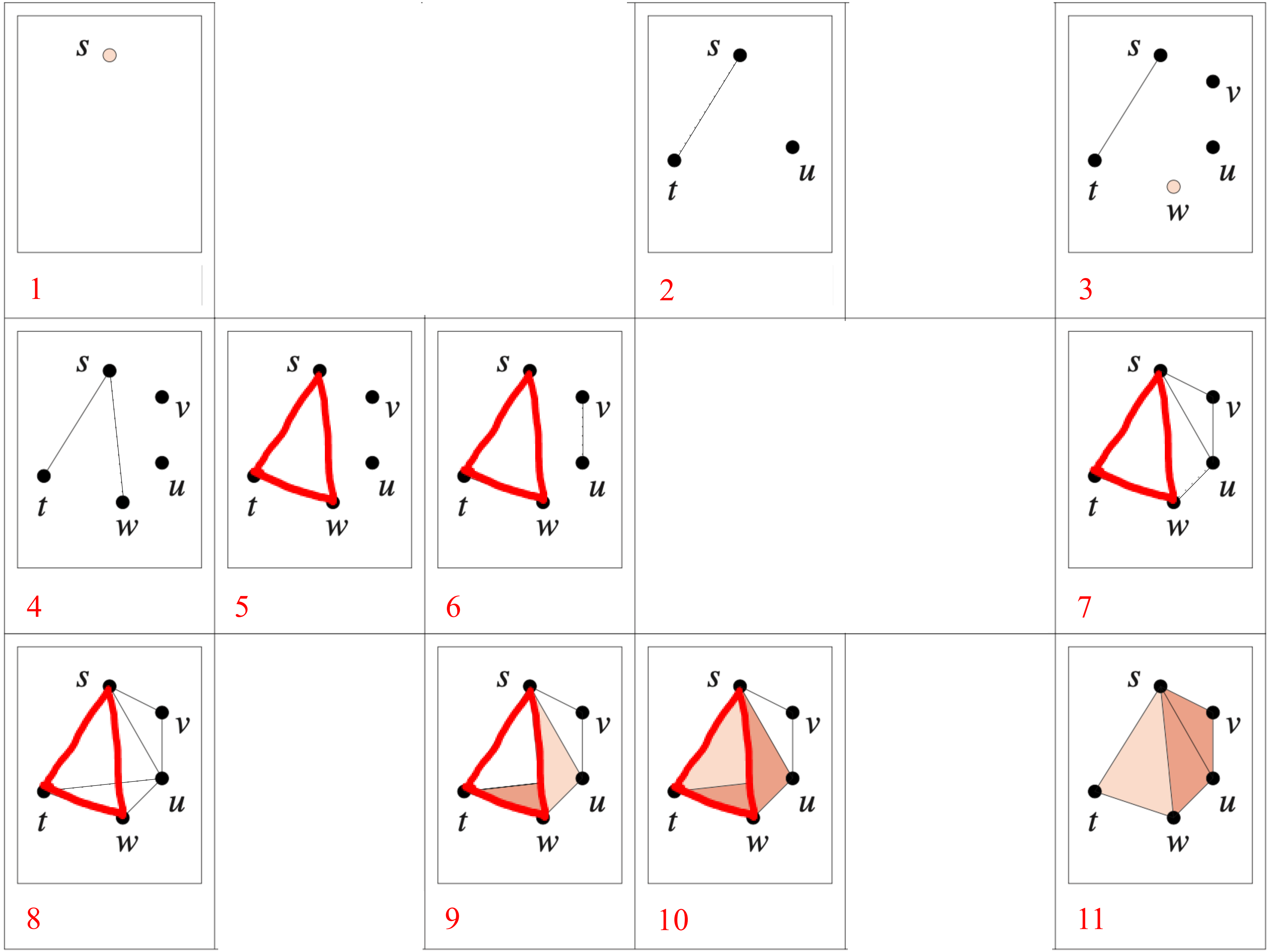


Image source: Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification.