A Fast Algorithm for Computing Zigzag Representatives

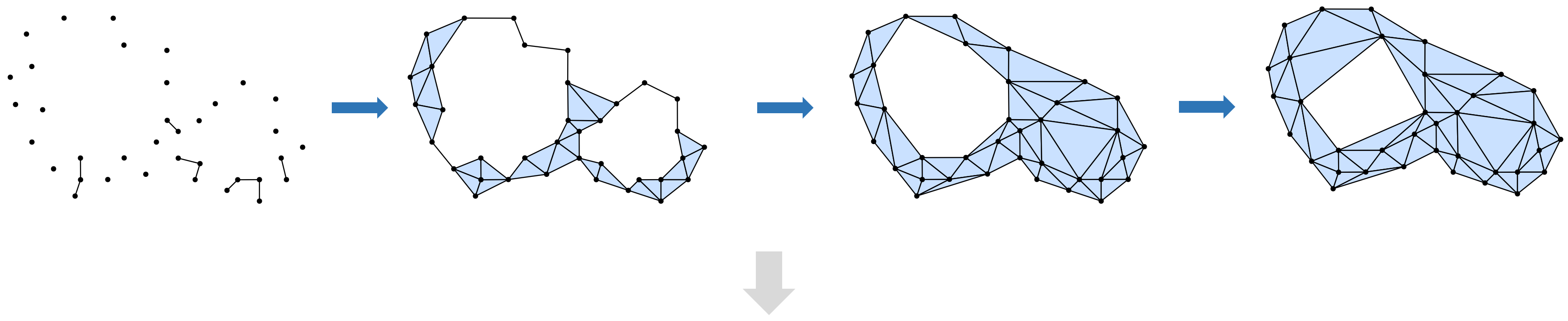
SODA 25'

Tamal K. Dey, Purdue University

***Tao Hou**, University of Oregon*

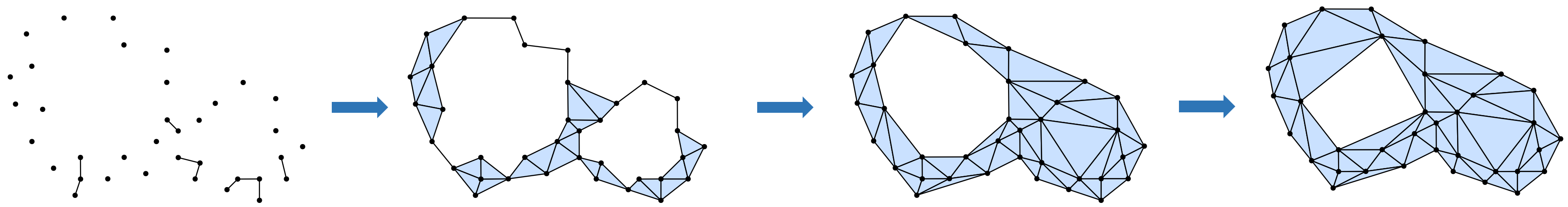
Dmitriy Morozov, Lawrence Berkeley National Laboratory

Persistent homology (non-zigzag version)



- As we add each simplex in the sequence, the homology of the complex changes, with:
  - Birth: betti number increased by 1
  - Death: betti number decreased by 1

Persistent homology (non-zigzag version)



- As we add each simplex in the sequence, the homology of the complex changes, with:
  - Birth: betti number increased by 1
  - Death: betti number decreased by 1
- The birth and death points can be canonically paired, resulting in persistence barcode:

Persistent homology: Pipeline

Simplex-wise filtration:

$$\mathcal{F} : K_0 \xhookrightarrow{\sigma_0} K_1 \xhookrightarrow{\sigma_1} \cdots \xhookrightarrow{\sigma_{m-2}} K_{m-1} \xhookrightarrow{\sigma_{m-1}} K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}(\mathcal{F}) : \mathsf{H}(K_0) \xrightarrow{\psi_0^*} \mathsf{H}(K_1) \xrightarrow{\psi_1^*} \cdots \xrightarrow{\psi_{m-2}^*} \mathsf{H}(K_{m-1}) \xrightarrow{\psi_{m-1}^*} \mathsf{H}(K_m)$$

$$\Downarrow$$

Interval decomposition: [Gabriel 72]

$$\mathsf{H}(\mathcal{F}) = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$$

$$\Downarrow$$

persistence barcode:

$$\mathsf{Pers}(\mathcal{F}) = \{[b_\alpha, d_\alpha] \mid \alpha \in \mathcal{A}\}$$

Zigzag persistence

Zigzag filtration:

$$\mathcal{F} : K_0 \overset{\sigma_0}{\longleftrightarrow} K_1 \overset{\sigma_1}{\longleftrightarrow} \cdots \overset{\sigma_{m-2}}{\longleftrightarrow} K_{m-1} \overset{\sigma_{m-1}}{\longleftrightarrow} K_m$$

Zigzag persistence

Zigzag filtration:

$$\mathcal{F} : K_0 \xleftrightarrow{\sigma_0} K_1 \xleftrightarrow{\sigma_1} \cdots \xleftrightarrow{\sigma_{m-2}} K_{m-1} \xleftrightarrow{\sigma_{m-1}} K_m$$

$$K_i \xhookrightarrow{\sigma_i} K_{i+1} \textbf{ or } K_i \xhookleftarrow{\sigma_i} K_{i+1}$$

Zigzag persistence

Zigzag filtration:

$$\mathcal{F} : K_0 \overset{\sigma_0}{\longleftrightarrow} K_1 \overset{\sigma_1}{\longleftrightarrow} \cdots \overset{\sigma_{m-2}}{\longleftrightarrow} K_{m-1} \overset{\sigma_{m-1}}{\longleftrightarrow} K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}(\mathcal{F}) : \mathsf{H}(K_0) \overset{\psi_0^*}{\longleftrightarrow} \mathsf{H}(K_1) \overset{\psi_1^*}{\longleftrightarrow} \cdots \overset{\psi_{m-2}^*}{\longleftrightarrow} \mathsf{H}(K_{m-1}) \overset{\psi_{m-1}^*}{\longleftrightarrow} \mathsf{H}(K_m)$$

$$\Downarrow$$

Interval decomposition: [Gabriel 72]

$$\mathsf{H}(\mathcal{F}) = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$$

$$\Downarrow$$

persistence barcode:

$$\mathsf{Pers}(\mathcal{F}) = \{[b_\alpha, d_\alpha] \mid \alpha \in \mathcal{A}\}$$

Non-zigzag vs. zigzag: Computing

Non-zigzag [ELZ2000]

integer YOUNGEST (simplex σ^j)
  $\Lambda = \{\sigma \in \partial_{k+1}(\sigma^j) \mid \sigma$ positive$\}$;
  loop
    $i = \max(\Lambda)$;
    if T$[i]$ is unoccupied then
      store j and Λ in T$[i]$; exit
    endif;
    $\Lambda = \Lambda + \Lambda^i$
  forever;
  return i.

Zigzag [CdSM2009]

Case f_i: We compute the representation of the boundary of simplex σ in terms of the cycles Z_i, and then reduce the result among the boundaries, obtaining: $\partial\sigma = Z_i v = Z_i(B_i u + v')$. There are two possibilities:

Birth: If $v' = 0$, then $\partial\sigma$ is already a boundary, and addition of σ creates a new cycle, for example, $C_i u - \sigma$. We append this cycle to the matrix Z_i, and we append $i+1$ to both the birth vector $\mathbf{b}_i$ and the index vector $\mathbf{idx}_i$ to get $\mathbf{b}_{i+1}$ and $\mathbf{idx}_{i+1}$, respectively.

Death: If $v' \neq 0$, then let j be the row of the lowest non-zero element in vector v'. We output a pair $(\mathbf{b}_i[j], i)$. We append vector v' to the matrix B_i, and the corresponding chain $(C_i u - \sigma)$ to the matrix C_i to obtain matrices B_{i+1} and C_{i+1}, respectively.

Case g_i: There are once again two possibilities:

Birth: There is no cycle in matrix Z_i that contains simplex σ. Let j be the index of the first column in C_i that contains σ, let l be the index of the row of the lowest non-zero element in $B_i[j]$.

1. Prepend $D_i C_i[j]$ to Z_i to get Z_i'. Prepend $i+1$ to the birth vector $\mathbf{b}_i$ to get $\mathbf{b}_{i+1}$.
2. Let $c = C_i[j][\sigma]$ be the coefficient of σ in the chain $C_i[j]$. Let $\mathbf{r}_\sigma$ be the row of σ in matrix C_i. We prepend the row $-\mathbf{r}_\sigma/c$ to the matrix B_i to get B_i'.
3. Subtract $(\mathbf{r}_\sigma[k]/c) \cdot C_i[j]$ from every column $C_i[k]$ to get C_i'.
4. Subtract $(B_i'[k][l]/B_i'[j][l]) \cdot B_i'[j]$ from every other column $B_i'[k]$.
5. Drop row l and column j from B_i' to get B_{i+1}, drop column l from Z_i', and drop column j from C_i to get C_{i+1}.
6. Reduce Z_{i+1} initially set to Z_i':
   1: **while** $\exists\, k < j$ s.t. low $Z_{i+1}[j] =$ low $Z_{i+1}[k]$ **do**
   2: $s =$ low $Z_{i+1}[j]$, $s_k^j = Z_{i+1}[j][s]/Z_{i+1}[k][s]$
   3: $Z_{i+1}[j] = Z_{i+1}[j] - s_k^j \cdot Z_{i+1}[k]$
   4: In B_{i+1}, add row j multiplied by s_k^j to row k

We set the index $\mathbf{idx}_{i+1}$ of the prepended cycle to be 1, and increase the index of every other column by 1. Figure 5 illustrates the changes made in this case.

Death: Let $Z_i[j]$ be the first cycle that contains simplex σ. Output $(\mathbf{b}_i[j], i)$.

1. Change basis to remove σ from matrix Z_i:
   1: **for** increasing $k > j$ s.t. $\sigma \in Z_i[k]$ **do**
   2: Let $\sigma_j^k = Z_i[k][\sigma]/Z_i[j][\sigma]$
   3: $Z_{i+1}[k] = Z_i[k] - \sigma_j^k \cdot Z_i[j]$
   4: In B_i, add row k multiplied by σ_j^k to row j
   5: **if** low $Z_{i+1}[k] >$ low $Z_i[k]$ **then**
   6: $j = k$
2. Subtract cycle $(C_i[k][\sigma]/Z_i[j][\sigma]) \cdot Z_i[j]$ from every chain $C_i[k]$.
3. Drop $Z_{i+1}[j]$, the corresponding entry in vectors $\mathbf{b}_i$ and $\mathbf{idx}_i$, row j from B_i, row σ from C_i and Z_i (as well as row and column of σ from D_i).

We increase the index of every column by 1, $\mathbf{idx}_{i+1}(l) = \mathbf{idx}_i(l) + 1$.

Representatives for persistent homology

Recall:

Filtration:

$$\mathcal{F} : K_0 \xleftrightarrow{\sigma_0} K_1 \xleftrightarrow{\sigma_1} \cdots \xleftrightarrow{\sigma_{m-2}} K_{m-1} \xleftrightarrow{\sigma_{m-1}} K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}(\mathcal{F}) : \mathsf{H}(K_0) \xleftrightarrow{\psi_0^*} \mathsf{H}(K_1) \xleftrightarrow{\psi_1^*} \cdots \xleftrightarrow{\psi_{m-2}^*} \mathsf{H}(K_{m-1}) \xleftrightarrow{\psi_{m-1}^*} \mathsf{H}(K_m)$$

$$\Downarrow$$

Interval decomposition:

$$\mathsf{H}(\mathcal{F}) = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$$

An interval module over [*b*, *d*]:

$$\begin{array}{ccccccccccccccccccccc} & & & & & & [b, & & b+1, & & \cdots, & & d] & & & & & & \\ 0 & \leftrightarrow & \cdots & \leftrightarrow & 0 & \leftrightarrow & \mathbb{Z}_2 & \xleftrightarrow{=} & \mathbb{Z}_2 & \xleftrightarrow{=} & \cdots & \xleftrightarrow{=} & \mathbb{Z}_2 & \leftrightarrow & 0 & \leftrightarrow & \cdots & \leftrightarrow & 0 \end{array}$$

Representatives for persistent homology

Recall:

Filtration:

$$\mathcal{F} : K_0 \overset{\sigma_0}{\longleftrightarrow} K_1 \overset{\sigma_1}{\longleftrightarrow} \cdots \overset{\sigma_{m-2}}{\longleftrightarrow} K_{m-1} \overset{\sigma_{m-1}}{\longleftrightarrow} K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}(\mathcal{F}) : \mathsf{H}(K_0) \overset{\psi_0^*}{\longleftrightarrow} \mathsf{H}(K_1) \overset{\psi_1^*}{\longleftrightarrow} \cdots \overset{\psi_{m-2}^*}{\longleftrightarrow} \mathsf{H}(K_{m-1}) \overset{\psi_{m-1}^*}{\longleftrightarrow} \mathsf{H}(K_m)$$

$$\Downarrow$$

Interval decomposition:

$$\mathsf{H}(\mathcal{F}) = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$$

An interval module over [*b*, *d*]:

$$\begin{array}{ccccccccccccccccccc} & & & & & & [b, & & b+1, & & \cdots, & & d] & & & & & & \\ 0 & \leftrightarrow & \cdots & \leftrightarrow & 0 & \leftrightarrow & \mathbb{Z}_2 & \overset{=}{\longleftrightarrow} & \mathbb{Z}_2 & \overset{=}{\longleftrightarrow} & \cdots & \overset{=}{\longleftrightarrow} & \mathbb{Z}_2 & \leftrightarrow & 0 & \leftrightarrow & \cdots & \leftrightarrow & 0 \end{array}$$

Definition. Given the decomposition into interval modules, a *representative* for an interval [*b*, *d*] is a sequence of cycles which form the interval module over [*b*, *d*]

- aka. homology class of cycle at each index generates the one-dimensional vector space at the index

Representatives for persistent homology

Recall:

Filtration:

$$\mathcal{F} : K_0 \overset{\sigma_0}{\longleftrightarrow} K_1 \overset{\sigma_1}{\longleftrightarrow} \cdots \overset{\sigma_{m-2}}{\longleftrightarrow} K_{m-1} \overset{\sigma_{m-1}}{\longleftrightarrow} K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}(\mathcal{F}) : \mathsf{H}(K_0) \overset{\psi_0^*}{\longleftrightarrow} \mathsf{H}(K_1) \overset{\psi_1^*}{\longleftrightarrow} \cdots \overset{\psi_{m-2}^*}{\longleftrightarrow} \mathsf{H}(K_{m-1}) \overset{\psi_{m-1}^*}{\longleftrightarrow} \mathsf{H}(K_m)$$

$$\Downarrow$$

Interval decomposition:

$$\mathsf{H}(\mathcal{F}) = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$$

An interval module over [*b*, *d*]:

$$\begin{array}{ccccccccccccccccc} & & & & & [b, & & b+1, & & \cdots, & & d] & & & & & \\ 0 & \leftrightarrow & \cdots & \leftrightarrow & 0 & \leftrightarrow & \mathbb{Z}_2 & \overset{=}{\longleftrightarrow} & \mathbb{Z}_2 & \overset{=}{\longleftrightarrow} & \cdots & \overset{=}{\longleftrightarrow} & \mathbb{Z}_2 & \leftrightarrow & 0 & \leftrightarrow & \cdots \leftrightarrow 0 \end{array}$$

Definition. Given the decomposition into interval modules, a *representative* for an interval [*b*, *d*] is a sequence of cycles which form the interval module over [*b*, *d*]

- aka. homology class of cycle at each index generates the one-dimensional vector space at the index

Remark. Representatives for all of the intervals form a *compatible basis* for each homology group in the persistence module

- aka. not only we have a basis at each homology group, but also *the basic elements at consecutive groups map to each other* by the induced homomorphism

Representatives for persistent homology

Recall:

Filtration:

$$\mathcal{F} : K_0 \overset{\sigma_0}{\longleftrightarrow} K_1 \overset{\sigma_1}{\longleftrightarrow} \cdots \overset{\sigma_{m-2}}{\longleftrightarrow} K_{m-1} \overset{\sigma_{m-1}}{\longleftrightarrow} K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}(\mathcal{F}) : \mathsf{H}(K_0) \overset{\psi_0^*}{\longleftrightarrow} \mathsf{H}(K_1) \overset{\psi_1^*}{\longleftrightarrow} \cdots \overset{\psi_{m-2}^*}{\longleftrightarrow} \mathsf{H}(K_{m-1}) \overset{\psi_{m-1}^*}{\longleftrightarrow} \mathsf{H}(K_m)$$

$$\Downarrow$$

Interval decomposition:

$$\mathsf{H}(\mathcal{F}) = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$$

An interval module over [*b*, *d*]:

$$\begin{array}{ccccccccccccccccc} & & & & & [b, & & b+1, & & \cdots, & & d] & & & & & \\ 0 & \to & \cdots & \to & 0 & \to & \mathbb{Z}_2 & \xrightarrow{\approx} & \mathbb{Z}_2 & \xrightarrow{\approx} & \cdots & \xrightarrow{\approx} & \mathbb{Z}_2 & \to & 0 & \to \cdots \to & 0 \end{array}$$

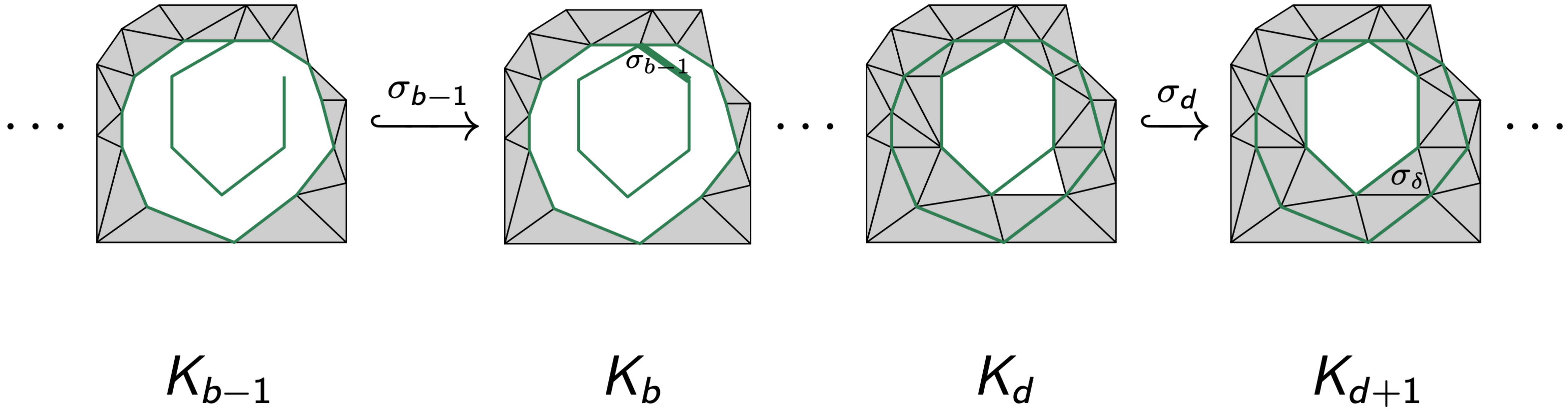
Definition. Given the decomposition into interval modules, a *representative* for an interval [*b*, *d*] is a sequence of cycles which form the interval module over [*b*, *d*]

- aka. homology class of cycle at each index generates the one-dimensional vector space at the index

Definition. A representative for an interval [*b*, *d*] in the *non-zigzag* persistence is just a cycle *z*

- born in K_b (aka. in K_b but not in K_{b-1}) and
- dying entering K_{d+1} (aka. not a boundary in K_d but becomes a boundary in K_{d+1})

Representatives for non-zigzag: Example



Zigzag representatives

- Representative for non-zigzag contains a single cycle: all inclusion maps are forward
- Not true for zigzag persistence: a representative for zigzag is a sequence of cycles generating an interval module

Zigzag representatives

- Representative for non-zigzag contains a single cycle: all inclusion maps are forward
- Not true for zigzag persistence: a representative for zigzag is a sequence of cycles generating an interval module

Definition. A *representative* for $[b, d]$ is a sequence of cycles

$$\mathsf{rep} = \{z_\alpha \in \mathsf{Z}(K_\alpha) \,|\, \alpha \in [b, d]\}$$

such that for every $b \leq \alpha < d$,

$$\text{either } [z_\alpha] \mapsto [z_{\alpha+1}] \text{ or } [z_{\alpha+1}] \leftarrow\!\shortmid [z_\alpha] \text{ by } \psi_\alpha^*.$$

Zigzag representatives

- Representative for non-zigzag contains a single cycle: all inclusion maps are forward
- Not true for zigzag persistence: a representative for zigzag is a sequence of cycles generating an interval module

Definition. A *representative* for $[b, d]$ is a sequence of cycles

$$\mathsf{rep} = \{z_\alpha \in \mathsf{Z}(K_\alpha) \mid \alpha \in [b, d]\}$$

such that for every $b \leq \alpha < d$,

$$\text{either } [z_\alpha] \mapsto [z_{\alpha+1}] \text{ or } [z_{\alpha+1}] \leftarrow\!\shortmid [z_\alpha] \text{ by } \psi^*_\alpha.$$

Furthermore:

Birth condition:

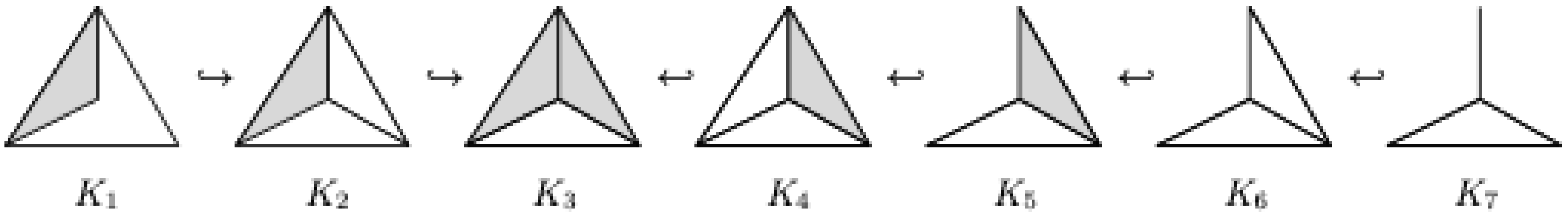
- $\psi^*_{b-1} : \mathsf{H}(K_{b-1}) \to \mathsf{H}(K_b)$ is forward: $z_b \in \mathsf{Z}(K_b) \setminus \mathsf{Z}(K_{b-1})$;
- $\psi^*_{b-1} : \mathsf{H}(K_{b-1}) \leftarrow \mathsf{H}(K_b)$ is backward: $[z_b]$ is the non-zero element in $\mathsf{Ker}(\psi^*_{b-1})$.

Death condition:

- $\psi^*_d : \mathsf{H}(K_d) \leftarrow \mathsf{H}(K_{d+1})$ is backward: $z_d \in \mathsf{Z}(K_d) \setminus \mathsf{Z}(K_{d+1})$;
- $\psi^*_d : \mathsf{H}(K_d) \to \mathsf{H}(K_{d+1})$ is forward: $[z_d]$ is the non-zero element in $\mathsf{Ker}(\psi^*_d)$.

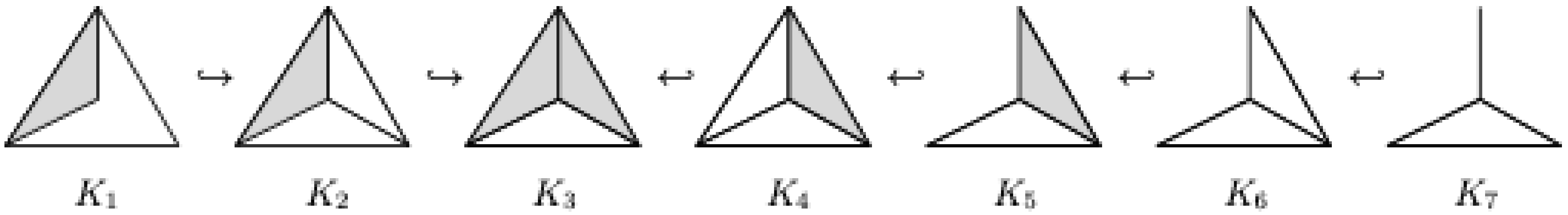
Zigzag representatives: Example

Filtration:



Zigzag representatives: Example

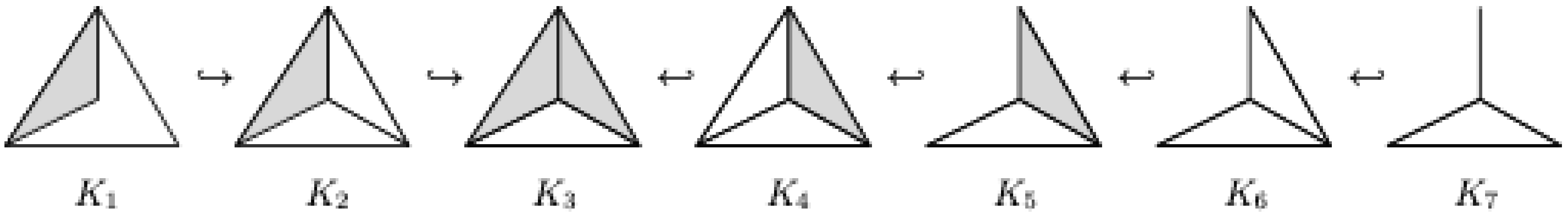
Filtration:



Interval:

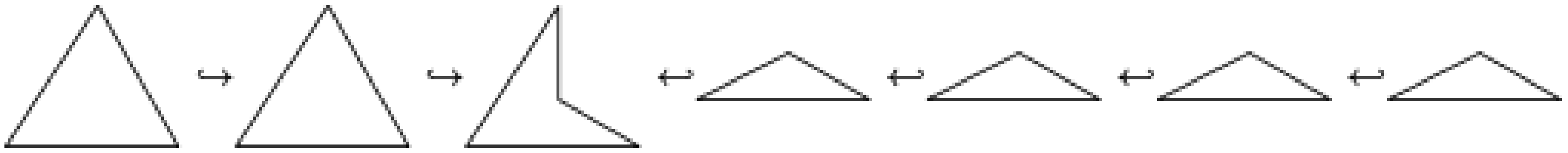
Zigzag representatives: Example

Filtration:



Interval:

Representative:



Contributions

Computing barcodes	Computing representatives
$O(m^{\omega})$ / $O(mn^2)$	$O(m^2n^2)$

m: length of filtration
n: maximum size of complexes

Contributions

Computing barcodes	Computing representatives
$O(m^{\omega})$ / $O(mn^2)$	$O(m^2n^2)$

m: length of filtration
n: maximum size of complexes

Reason for naive $O(m^2n^2)$ complexity (see also [Maria&Outdot2015]):

- During computing the barcode, there are $O(mn)$ summations of intervals and their representatives
- Each representative takes $O(mn)$ space and hence their summation takes $O(mn)$ time

Contributions

Computing barcodes	Computing representatives
$O(m^\omega)$ / $O(mn^2)$	$O(m^2n^2) \implies O(m^2n)$

m: length of filtration
n: maximum size of complexes

Contributions

Computing barcodes	Computing representatives
$O(m^\omega)$ / $O(mn^2)$	$O(m^2n^2)$ $\Longrightarrow$ $O(m^2n)$

m: length of filtration
n: maximum size of complexes

Reason for the improved $O(m^2n)$ complexity:

- There are still $O(mn)$ summations of representatives
- **BUT** each representative takes $O(m)$ space in our algorithm (instead of $O(mn)$) and their summation takes $O(m)$ time

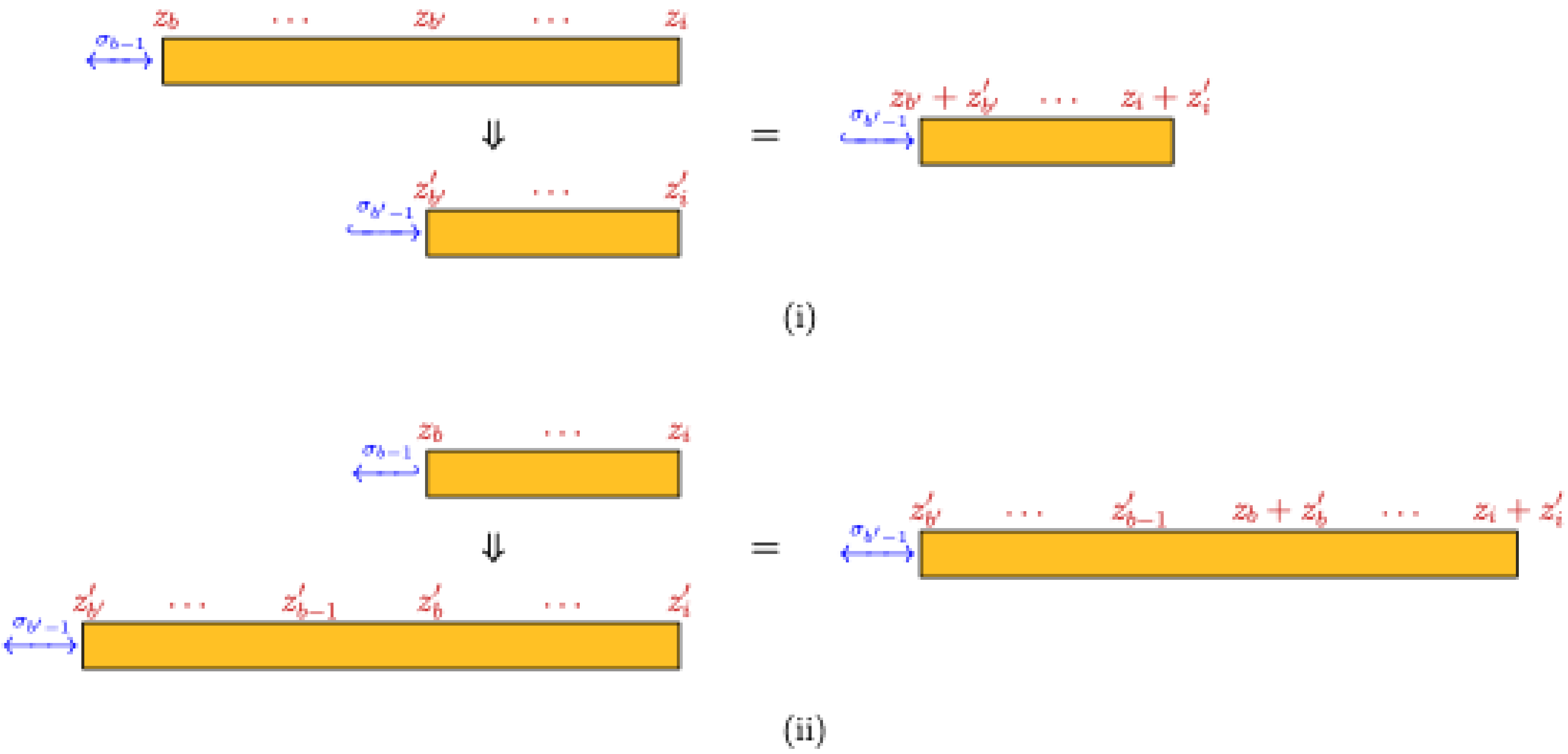
"Naive" $O(m^2n^2)$ algorithm

- Based on a direct maintenance of representatives.
- Scan each $K_i \overset{\sigma_i}{\longleftrightarrow} K_{i+1}$ in the filtration one by one:
  - During the process we maintain a set of "*active*" intervals (those ending with the current index i)
  - When we encounter a birth, we start a new interval $[i+1, i+1]$ and assign it a new representative
  - When we encounter a death, we choose an active interval to kill (which ends with i), and possibly assign the killed interval a new (finalized) representative
  - We also *extend* those active intervals which we do not choose to kill from i to $i+1$, whose representatives may change during extension

"Naive" $O(m^2n^2)$ algorithm

- Based on a direct maintenance of representatives.
- Scan each $K_i \overset{\sigma_i}{\longleftrightarrow} K_{i+1}$ in the filtration one by one:
  - During the process we maintain a set of "*active*" intervals (those ending with the current index i)
  - When we encounter a birth, we start a new interval $[i+1, i+1]$ and assign it a new representative
  - When we encounter a death, we choose an active interval to kill (which ends with i), and possibly assign the killed interval a new (finalized) representative
  - We also *extend* those active intervals which we do not choose to kill from i to $i+1$, whose representatives may change during extension
- **Observation**: Other than having a new representative for a new starting interval, changing representatives for the intervals are done by **representative summations** (**a critical fact leading to the improvement**)

Illustration of representative summation



Wires and bundles

- Key definitions leading to the improvement. With details omitted:
  - A *wire* is a cycle $\omega_i \in Z(K_i)$ with a *starting index* i
  - A *bundle* is a set of wires
- A key observation: *zigzag representatives in our algorithm can always be generated from bundles of wires*

Wires and bundles

- Key definitions leading to the improvement. With details omitted:
    - A *wire* is a cycle $\omega_i \in Z(K_i)$ with a *starting index* i
    - A *bundle* is a set of wires
- A key observation: *zigzag representatives in our algorithm can always be generated from bundles of wires*

Definition. A *wire* is a cycle $\omega_i \in \mathsf{Z}(K_i)$ with a *starting index* $i \in \mathsf{P}^H(\mathcal{F}) \cup \mathsf{P}^B(\mathcal{F})$ s.t.

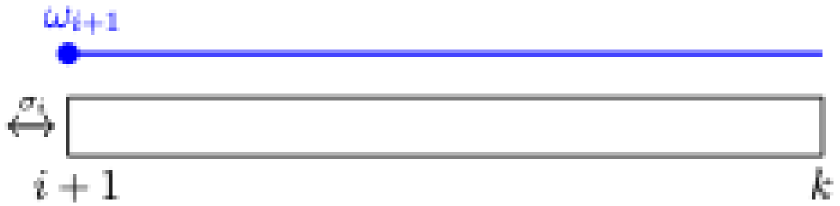
(i) $K_{i-1} \hookrightarrow K_i$ is forward and $\omega_i \in \mathsf{Z}(K_i) \setminus \mathsf{Z}(K_{i-1})$, or

(ii) $K_{i-1} \hookleftarrow K_i$ is backward and $\omega_i \in \mathsf{B}(K_{i-1}) \setminus \mathsf{B}(K_i)$, or

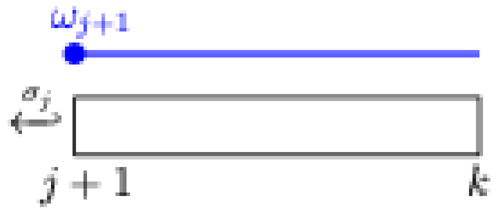
(iii) $K_{i-1} \hookrightarrow K_i$ is forward and $\omega_i \in \mathsf{B}(K_i) \setminus \mathsf{B}(K_{i-1})$.

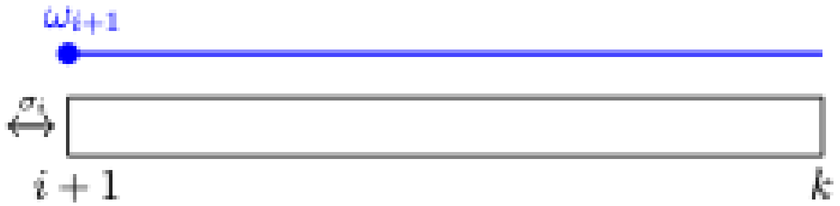
We also say that ω_i is a wire *at* index i. The wires satisfying (i) or (ii) are also called *non-boundary wires* whereas those satisfying (iii) are called *boundary wires*.

Representatives as bundles of wires

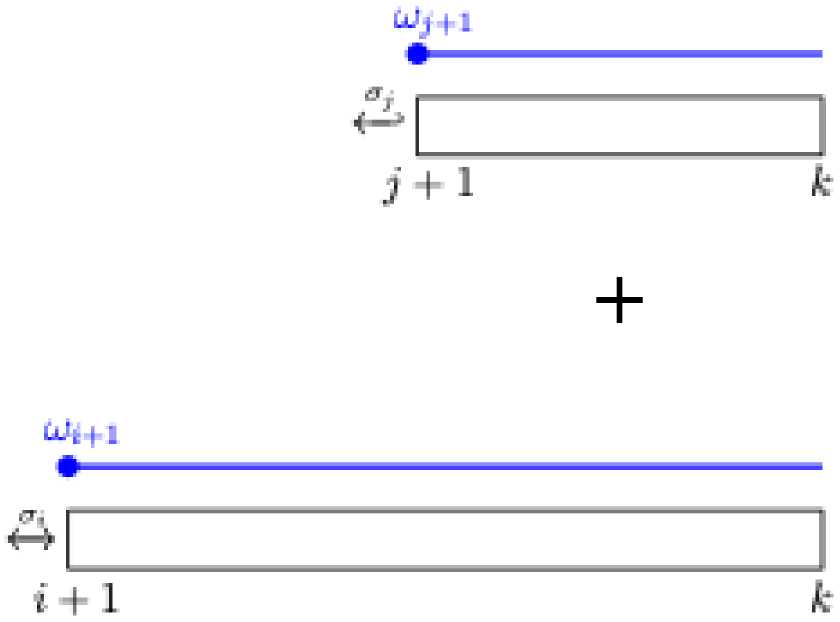


Representatives as bundles of wires

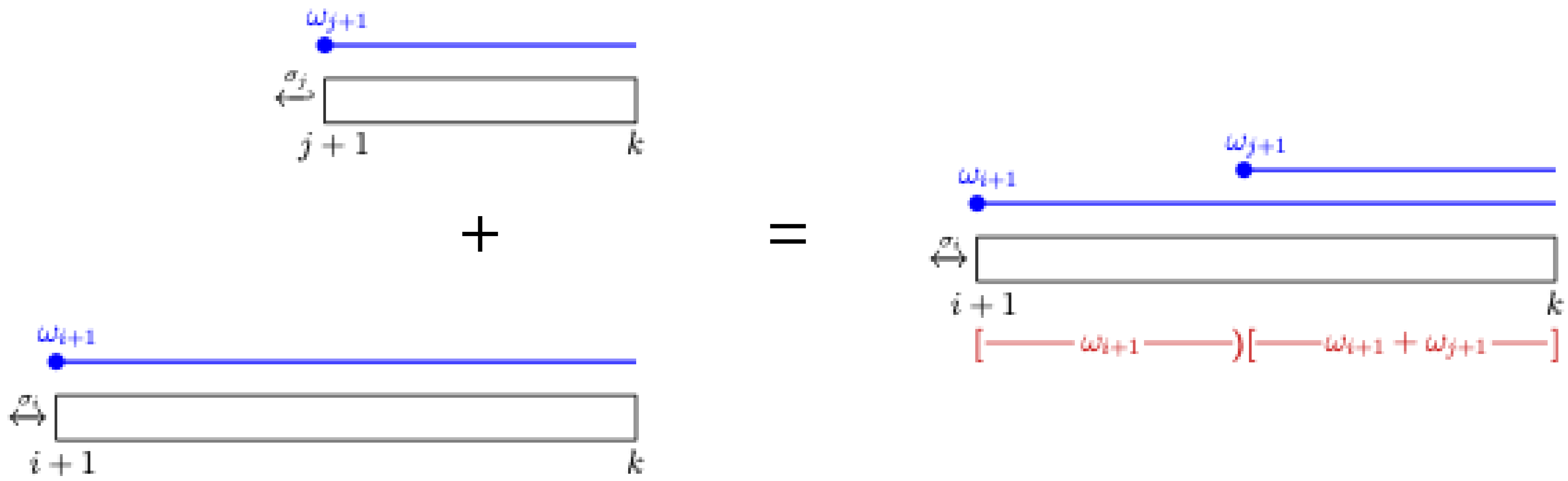




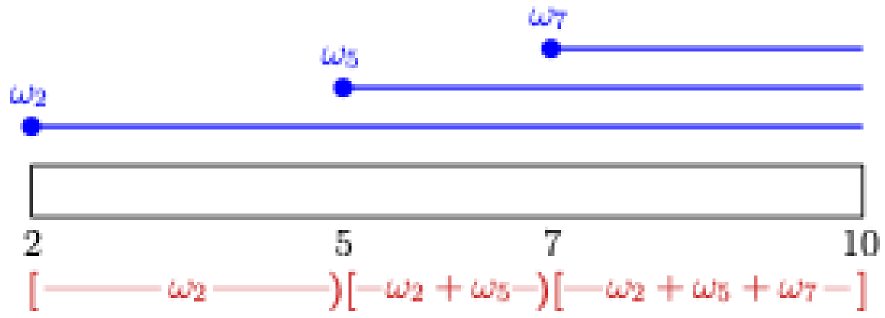
Representatives as bundles of wires

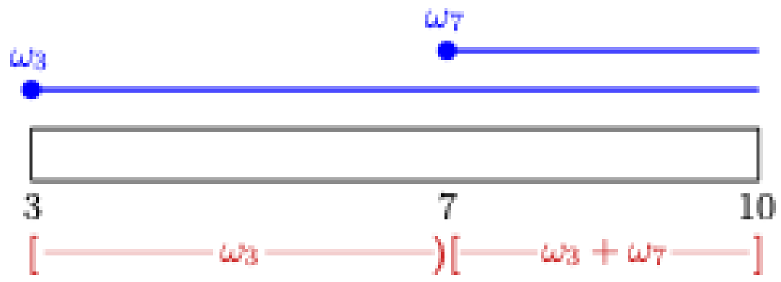


Representatives as bundles of wires

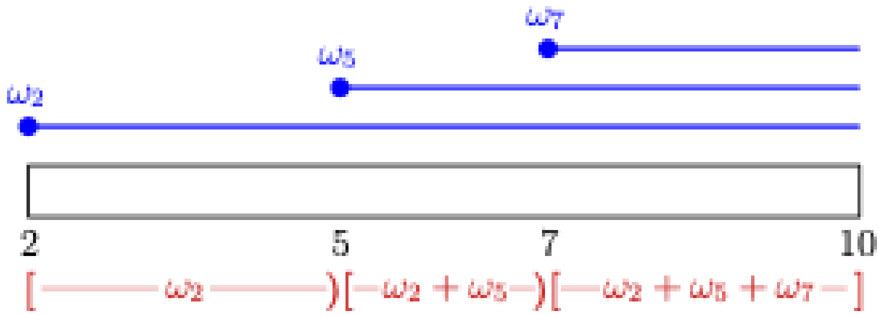


Representatives as bundles of wires

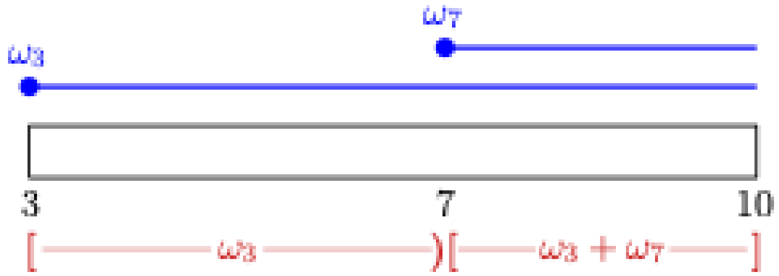




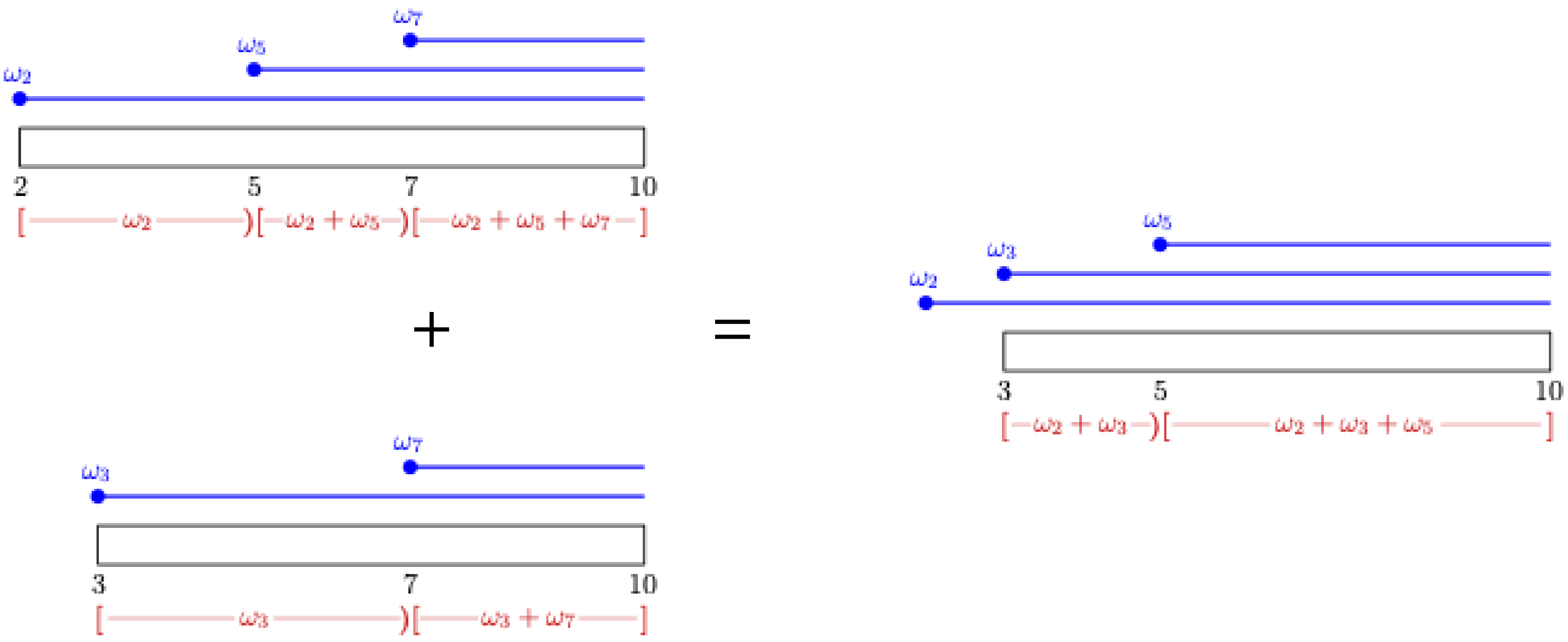
Representatives as bundles of wires



+



Representatives as bundles of wires



Bundles and wires for improvement

- **Fact** [DHM25]: *zigzag representatives in our algorithm can always be generated from bundles of wires*

Theorem. *There is a wire bundle* $W = \{w_\iota \mid \iota \in \mathsf{P}^H(\mathcal{F}) \cup \mathsf{P}^B(\mathcal{F})\}$ *so that a representative for any* $[b, d] \in \mathsf{Pers}^H(\mathcal{F}) \cup \mathsf{Pers}^B(\mathcal{F})$ *is generated by a wire bundle that is a subset of* W*.*

Bundles and wires for improvement

- **Fact** [DHM25]: *zigzag representatives in our algorithm can always be generated from bundles of wires*

Theorem. *There is a wire bundle* $W = \{w_\iota \mid \iota \in \mathsf{P}^H(\mathcal{F}) \cup \mathsf{P}^B(\mathcal{F})\}$ *so that a representative for any* $[b, d] \in \mathsf{Pers}^H(\mathcal{F}) \cup \mathsf{Pers}^B(\mathcal{F})$ *is generated by a wire bundle that is a subset of* W*.*

- **Fact**: Different wires start with different indices, so we store each representative as a bundle which is nothing but a set of $O(m)$ indices

Bundles and wires for improvement

- **Fact** [DHM25]: *zigzag representatives in our algorithm can always be generated from bundles of wires*

Theorem. *There is a wire bundle* $W = \{w_\iota \mid \iota \in \mathsf{P}^H(\mathcal{F}) \cup \mathsf{P}^B(\mathcal{F})\}$ *so that a representative for any* $[b,d] \in \mathsf{Pers}^H(\mathcal{F}) \cup \mathsf{Pers}^B(\mathcal{F})$ *is generated by a wire bundle that is a subset of* W*.*

- **Fact**: Different wires start with different indices, so we store each representative as a bundle which is nothing but a set of $O(m)$ indices
- **Fact**: Summing two representatives boils down to symmetric difference of two bundles (sets) of indices

Proposition. *Let* $[b,i], [b',i] \in \mathsf{Pers}^H(\mathcal{F}_i) \cup \mathsf{Pers}^B(\mathcal{F}_i)$ *and* $b \prec b'$*. Suppose that* W *and* W' *generate a representative for* $[b,i]$ *and* $[b',i]$ *respectively. Then, the sum* $W \boxplus W'$ *generates a representative for* $[b',i] \in \mathsf{Pers}^H(\mathcal{F}_i) \cup \mathsf{Pers}^B(\mathcal{F}_i)$*.*

Bundles and wires for improvement

- **Fact** [DHM25]: *zigzag representatives in our algorithm can always be generated from bundles of wires*

> **Theorem.** *There is a wire bundle* $W = \{w_\iota \mid \iota \in \mathsf{P}^H(\mathcal{F}) \cup \mathsf{P}^B(\mathcal{F})\}$ *so that a representative for any* $[b,d] \in \mathsf{Pers}^H(\mathcal{F}) \cup \mathsf{Pers}^B(\mathcal{F})$ *is generated by a wire bundle that is a subset of* W.

- **Fact**: Different wires start with different indices, so we store each representative as a bundle which is nothing but a set of $O(m)$ indices
- **Fact**: Summing two representatives boils down to symmetric difference of two bundles (sets) of indices

> **Proposition.** *Let* $[b,i], [b',i] \in \mathsf{Pers}^H(\mathcal{F}_i) \cup \mathsf{Pers}^B(\mathcal{F}_i)$ *and* $b \prec b'$. *Suppose that* W *and* W' *generate a representative for* $[b,i]$ *and* $[b',i]$ *respectively. Then, the sum* $W \boxplus W'$ *generates a representative for* $[b',i] \in \mathsf{Pers}^H(\mathcal{F}_i) \cup \mathsf{Pers}^B(\mathcal{F}_i)$.

- Summing two representatives now takes $O(m)$ time $\Longrightarrow O(mn)$ summations take $O(m^2 n)$ time

Generating representative from bundle

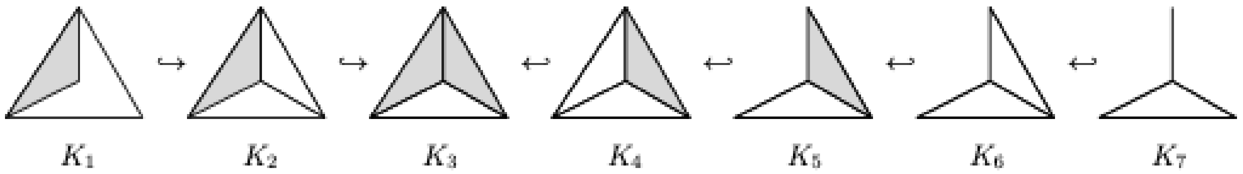
- Can be done in $O(mn)$ time:

Algorithm 1 (ExtRep: Extracting representative from bundle). *Let $W = \{\omega_{\iota_1}, \ldots, \omega_{\iota_\ell}\}$ be a wire bundle where $\iota_1 < \cdots < \iota_\ell$ and let* rep *be the representative for an interval $[b, d]$ generated by W. We can assume $\iota_\ell \leq d$ because wires in W with indices greater than d do not contribute to a cycle in* rep. *Moreover, let ι_k be the last index in $\iota_1, \ldots, \iota_\ell$ no greater than b. We have that $z = \sum_{j=\iota_1}^{\iota_k} \omega_j$ is the cycle at indices $[b, \iota_{k+1})$ in* rep. *We then let λ iterate over $k+1, \ldots, \ell-1$. For each λ, we add ω_{ι_λ} to z, and the resulting z is the cycle at indices $[\iota_\lambda, \iota_{\lambda+1})$ in* rep. *Finally, we add ω_{ι_ℓ} to z, and z is the cycle at indices $[\iota_\ell, d]$ in* rep. *Since at every $\lambda \in [k+1, \ell]$, we add at most one cycle to another cycle, the whole process involves $O(m)$ chain summations.*

- Since there are $O(m)$ intervals, the overall complexity of our algorithm is $O(m^2 n)$

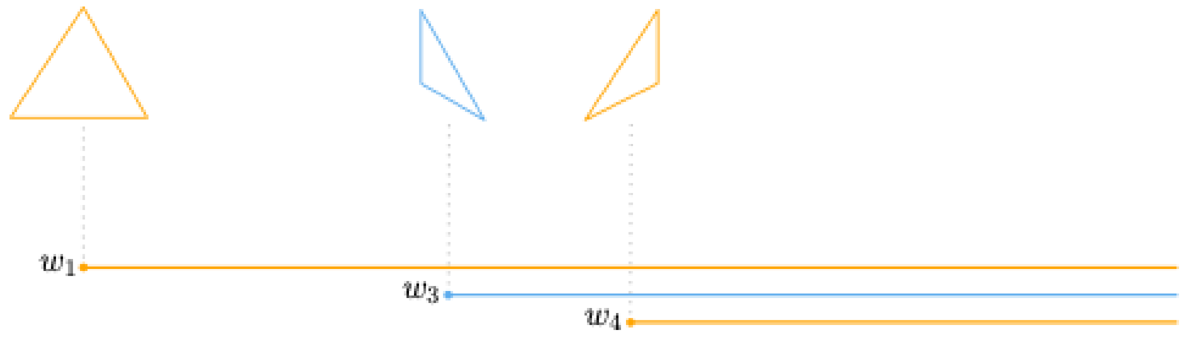
Representative from bundle: Example

Filtration:

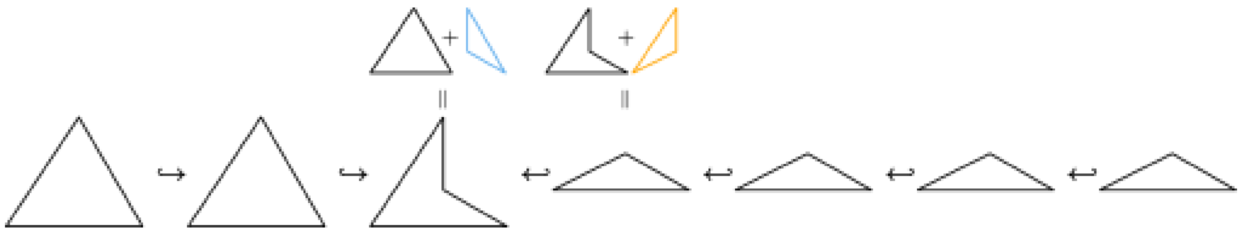


Interval:

Bundle:



Representative:



Implementation (and intro. of *boundary modules*)

Two keys to implement in $O(m^2 n)$ time:

1) Maintain two *pivoted matrices* Z and B whose columns have distinct pivots such that:
   - Columns of Z form a basis for $H(K_i)$
   - Columns of B form a basis for $B(K_i)$
2) We also need to make sure that each column of Z equals the last cycle in the representative generated by the wire bundle we maintain

Implementation (and intro. of *boundary modules*)

Two keys to implement in $O(m^2 n)$ time:

1) Maintain two *pivoted matrices* *Z* and *B* whose columns have distinct pivots such that:
   - Columns of *Z* form a basis for $H(K_i)$
   - Columns of *B* form a basis for $B(K_i)$
2) We also need to make sure that each column of *Z* equals the last cycle in the representative generated by the wire bundle we maintain

- But to make sure pivots are distinct, we cannot avoid summations of *B* to *Z*, making point (2) impossible to hold

Implementation (and intro. of *boundary modules*)

Two keys to implement in $O(m^2n)$ time:

1) Maintain two *pivoted matrices* Z and B whose columns have distinct pivots such that:
   - Columns of Z form a basis for $H(K_i)$
   - Columns of B form a basis for $B(K_i)$
2) We also need to make sure that each column of Z equals the last cycle in the representative generated by the wire bundle we maintain

- But to make sure pivots are distinct, we cannot avoid summations of B to Z, making point (2) impossible to hold
- Unless we **make the columns of *B* also correspond to bundles**! (which are another types of bundles called the *boundary bundles*)

Boundary module

Filtration:

$$\mathcal{F} : K_0 \xleftrightarrow{\sigma_0} K_1 \xleftrightarrow{\sigma_1} \cdots \xleftrightarrow{\sigma_{m-2}} K_{m-1} \xleftrightarrow{\sigma_{m-1}} K_m$$

$$\Downarrow$$

Induced *boundary* module:

$$\mathsf{B}(\mathcal{F}) : \mathsf{B}(K_0) \xleftrightarrow{\psi_0^\#} \mathsf{B}(K_1) \xleftrightarrow{\psi_1^\#} \cdots \xleftrightarrow{\psi_{m-2}^\#} \mathsf{B}(K_{m-1}) \xleftrightarrow{\psi_{m-1}^\#} \mathsf{B}(K_m)$$

- Apply the boundary functor B (instead of the homology functor H)
- Connecting homomorphisms $\psi_i^\#$ are chain maps
- Can still define interval decomposition, representatives, wire and bundles

Boundary module

Filtration:

$$\mathcal{F} : K_0 \overset{\sigma_0}{\longleftrightarrow} K_1 \overset{\sigma_1}{\longleftrightarrow} \cdots \overset{\sigma_{m-2}}{\longleftrightarrow} K_{m-1} \overset{\sigma_{m-1}}{\longleftrightarrow} K_m$$

$$\Downarrow$$

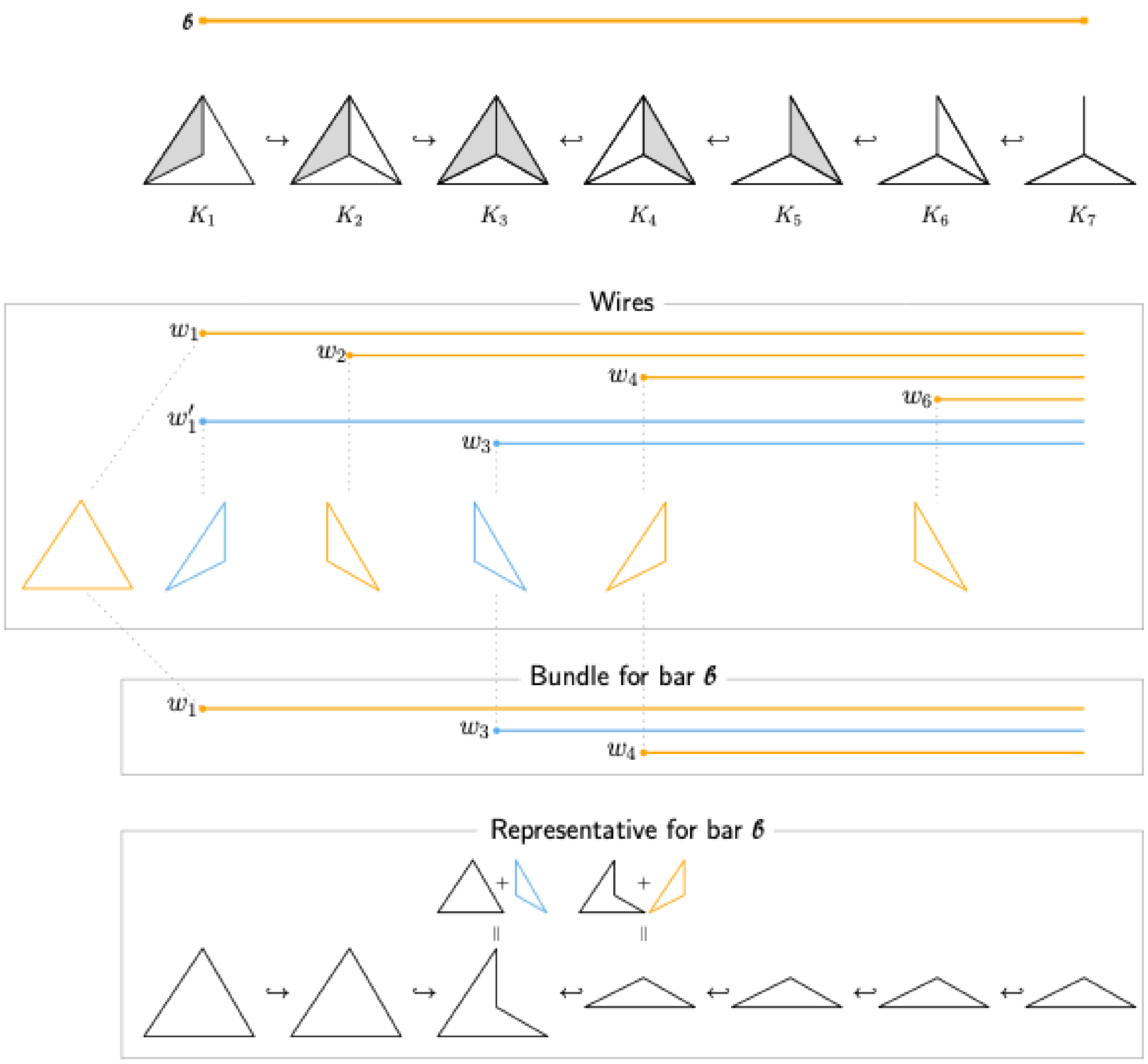
Induced *boundary* module:

$$\mathsf{B}(\mathcal{F}) : \mathsf{B}(K_0) \overset{\psi_0^\#}{\longleftrightarrow} \mathsf{B}(K_1) \overset{\psi_1^\#}{\longleftrightarrow} \cdots \overset{\psi_{m-2}^\#}{\longleftrightarrow} \mathsf{B}(K_{m-1}) \overset{\psi_{m-1}^\#}{\longleftrightarrow} \mathsf{B}(K_m)$$

- Apply the boundary functor B (instead of the homology functor H)
- Connecting homomorphisms $\psi_i^\#$ are chain maps
- Can still define interval decomposition, representatives, wire and bundles
- Let each column of *B* equal the last cycle in the representative (in boundary moudle) generated by the wire bundle we maintain
- We can then add columns in *B* to *Z*
- So representatives come from a mix of wires from the homology module and the boundary module

Wires for the interval $\mathscr{b}$ come from both the homology and boundary modules:

- Orange: homology module
- Blue: boundary module



Thank you!