Computing Zigzag Persistence on Graphs in Near-Linear Time

Tamal K. Dey and **Tao Hou**

Department of Computer Science
Purdue University

SoCG 2021

Student Presentation

Background: standard persistence

Standard filtration:

$$\mathcal{F} : K_0 \hookrightarrow K_1 \hookrightarrow \cdots \hookrightarrow K_{m-1} \hookrightarrow K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}_p(\mathcal{F}) : \mathsf{H}_p(K_0) \to \mathsf{H}_p(K_1) \to \cdots \to \mathsf{H}_p(K_{m-1}) \to \mathsf{H}_p(K_m)$$

$$\Downarrow$$

Interval decomposition:

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

$$\Downarrow$$

p-th persistence barcode:

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

Background: zigzag persistence

Zigzag filtration:

$$\mathcal{F} : K_0 \leftrightarrow K_1 \leftrightarrow \cdots \leftrightarrow K_{m-1} \leftrightarrow K_m$$

$$\Downarrow$$

Induced module:

$$\mathsf{H}_p(\mathcal{F}) : \mathsf{H}_p(K_0) \leftrightarrow \mathsf{H}_p(K_1) \leftrightarrow \cdots \leftrightarrow \mathsf{H}_p(K_{m-1}) \leftrightarrow \mathsf{H}_p(K_m)$$

$$\Downarrow$$

Interval decomposition:

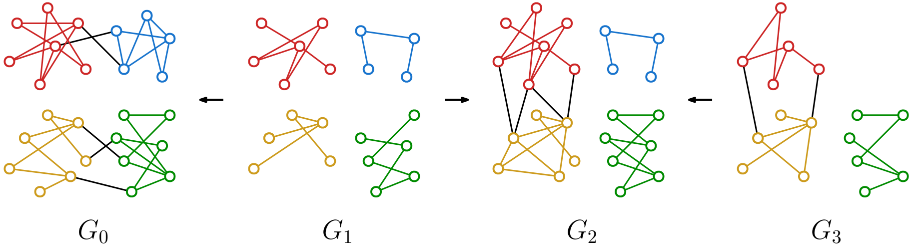
$$\mathsf{H}_p(\mathcal{F}) = \bigoplus_{\alpha \in \mathcal{A}} \mathcal{I}^{[b_\alpha, d_\alpha]}$$

$$\Downarrow$$

p-th persistence barcode:

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

Example of Zigzag Filtration (of Graphs)



Application: *dynamic networks*, etc.

Complexities of persistence computing

	*	Graphs
Standard	m^3, m^ω	$m\,\alpha(m)$
Zigzag	m^3, m^ω	?

m: length of filtration
$\omega < 2.373$: matrix multiplication exponent
$\alpha(m)$: inverse Ackermann function

$O(m^\omega)$ algorithm: Milosavljević, Morozov, and Skraba.
Zigzag persistent homology in matrix multiplication time. 2011.

Computation: Zigzag vs. Standard

Standard

[ELZ2000]

```
integer YOUNGEST (simplex σ^j)
  Λ = {σ ∈ ∂_{k+1}(σ^j) | σ positive};
  loop
    i = max(Λ);
    if T[i] is unoccupied then
      store j and Λ in T[i]; exit
    endif;
    Λ = Λ + Λ^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: $\quad s =$ low $Z_{i+1}[j]$, $s_k^j = Z_{i+1}[j][s]/Z_{i+1}[k][s]$
   3: $\quad Z_{i+1}[j] = Z_{i+1}[j] - s_k^j \cdot Z_{i+1}[k]$
   4: $\quad$ 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: $\quad$ Let $\sigma_j^k = Z_i[k][\sigma]/Z_i[j][\sigma]$
   3: $\quad Z_{i+1}[k] = Z_i[k] - \sigma_j^k \cdot Z_i[j]$
   4: $\quad$ In B_i, add row k multiplied by σ_j^k to row j
   5: $\quad$ **if** low $Z_{i+1}[k] >$ low $Z_i[k]$ **then**
   6: $\quad\quad j = k$
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, $\mathbf{idx}_{i+1}(l) = \mathbf{idx}_i(l) + 1$.

Contributions

Input:

$\mathcal{F} : \varnothing = G_0 \xleftrightarrow{\sigma_0} G_1 \xleftrightarrow{\sigma_1} \cdots \xleftrightarrow{\sigma_{m-1}} G_m$

$G = \bigcup_{i=0}^{m} G_i$

m: length of filtration, n: size of G

1. Dimension-0: $O(m \log^2 n + m \log m)$; works for any complex
2. Dimension-1: $O(m \log^4 n)$
3. Alexander duality: dimension-$(p-1)$ for $\mathbb{R}^p$-embedded complexes in $O(m \log^2 n + m \log m + n \log n)$ time

Contributions

- Dimension-0: $O(m \log^2 n + m \log m)$

Standard

- Only need to kill the older one when two connected components merge

Zigzag

- Connected components can split into smaller ones because of edge deletion
- Connected components can disappear because of vertex deletion
- Need to pair the merge and disappearing of the components with the split and entering of components

Contributions

- Dimension-1: $O(m \log^4 n)$

Standard

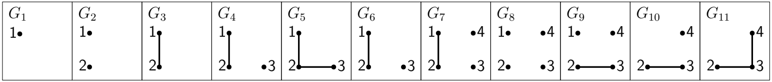
- Every newly created 1-cycles: infinite bars; no pairing

Zigzag

- Edge deletion kills 1-cycles
- Need to properly pair the creation and destruction of 1-cycles

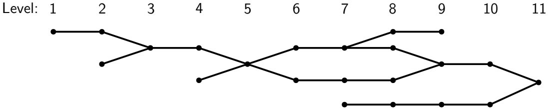
Algorithm for 0-dimension: Barcode graph

Input $\mathcal{F}$:



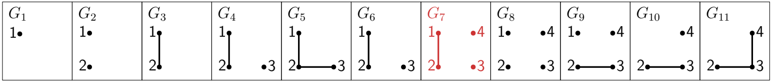
$$\Downarrow$$

Barcode graph $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$:



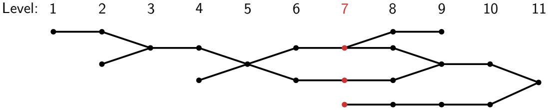
Algorithm for 0-dimension: Barcode graph

Input $\mathcal{F}$:



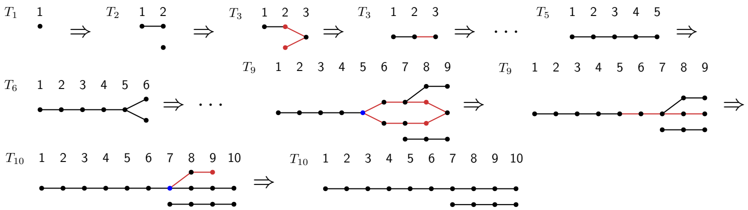
$\Downarrow$

Barcode graph $\mathbb{G}_{\mathrm{B}}(\mathcal{F})$:



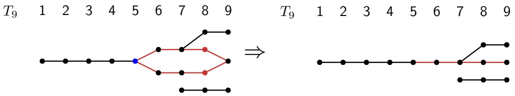
Building *barcode forest* T_i

- Based on [Agarwal, Edelsbrunner, Harer, Wang 2006]
- Build T_{i+1} from T_i under four cases:
  - Entrance
  - Split
  - Departure
  - Merge
- Update T_{i+1} and output the persistence intervals



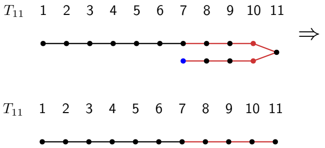
Updating *barcode forest* T_i

- Merge (in the same tree)



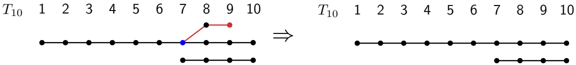
- Glue paths to their nearest common ancestor
- j: level of NCR
  i: current level
- Output: [j+1,i-1]

- Merge (in different trees)



- j: level of the higher root
- Glue paths to their level-j ancestors
- Output: [j,i-1]

- Departure



- Delete the path to its nearest splitting ancestor
- j: level of nearest splitting ancestor
  i: current level
- Output: [j+1,i-1]

Data structures

- Keep track of connectivity of graphs
  - Fully-dynamic connectivity [Holm, De Lichtenberg, Thorup 2001]: $O(\log^2 n)$
- Barcode forest
  - Mergeable trees [Georgiadis, Kaplan, Shafrir, Tarjan, Werneck 2011]: $O(\log m)$

Thus the complexity $O(m \log^2 n + m \log m)$

Correctness proof

Definition (Representatives; see also [Maria&Oudot 2014])

- $\mathcal{M} : V_0 \overset{\psi_0}{\longleftrightarrow} \cdots \overset{\psi_{m-1}}{\longleftrightarrow} V_m$: module induced by a simplex-wise filtration
- $[b, d] \subseteq [1, m]$: an interval

Representatives for $[b, d]$: a sequence $\{\alpha_i \in V_i \mid i \in [b, d]\}$ s.t.

1. Classes are connected: $\forall\, i \in [b, d-1]$, $\alpha_i \mapsto \alpha_{i+1}$ or $\alpha_i \mapsfrom \alpha_{i+1}$ by ψ_i
2. Birth end condition:
   $\psi_{b-1} : V_{b-1} \to V_b$: α_b is not in $\mathrm{im}(\psi_{b-1})$
   $\psi_{b-1} : V_{b-1} \leftarrow V_b$: α_b the non-zero element in $\ker(\psi_{b-1})$
3. Death end condition: symmetric to previous

Correctness proof

Definition (Representatives; see also [Maria&Oudot 2014])

- $\mathcal{M} : V_0 \overset{\psi_0}{\longleftrightarrow} \cdots \overset{\psi_{m-1}}{\longleftrightarrow} V_m$: module induced by a simplex-wise filtration
- $[b, d] \subseteq [1, m]$: an interval

Representatives for $[b, d]$: a sequence $\{\alpha_i \in V_i \mid i \in [b, d]\}$ s.t.

1. Classes are connected: $\forall\, i \in [b, d-1]$, $\alpha_i \mapsto \alpha_{i+1}$ or $\alpha_i \leftarrow\!\shortmid\, \alpha_{i+1}$ by ψ_i
2. Birth end condition:
   $\psi_{b-1} : V_{b-1} \rightarrow V_b$: α_b is not in $\mathrm{im}(\psi_{b-1})$
   $\psi_{b-1} : V_{b-1} \leftarrow V_b$: α_b the non-zero element in $\ker(\psi_{b-1})$
3. Death end condition: symmetric to previous

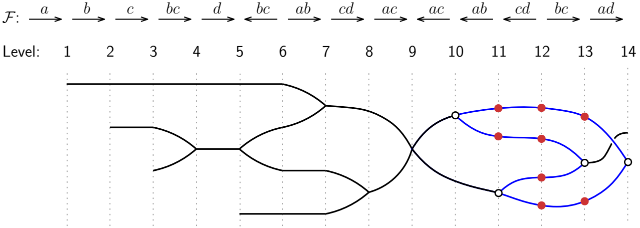
$$\begin{array}{lccccccccc} \mathcal{M} : & V_0 & \xrightarrow{\psi_0} & V_1 & \xleftarrow{\psi_1} & V_2 & \xrightarrow{\psi_2} & V_3 & \xrightarrow{\psi_3} & V_4 \\ & & & [\alpha_1 & \longleftarrow\!\shortmid & \alpha_2 & \longmapsto & \alpha_3] & \longmapsto & 0 \end{array}$$

Correctness proof

Proposition

Each interval produced by the algorithm admits a sequence of representatives.

Interval: $[11, 13]$



Correctness proof

Proposition

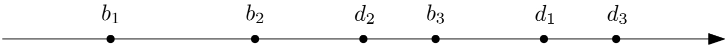
- $\mathcal{M}$: *module induced from a simplex-wise zigzag filtration*
- $\pi : \mathsf{P}(\mathcal{M}) \to \mathsf{N}(\mathcal{M})$: *a bijection*

If: $\forall\, b \in \mathsf{P}(\mathcal{M})$, $[b, \pi(b)]$ *has a sequence of representatives*

Then: $\mathsf{Pers}(\mathcal{M}) = \{[b, \pi(b)] \mid b \in \mathsf{P}(\mathcal{M})\}$

* $\mathsf{P}(\mathcal{M})$, **positive indices**: all starts of intervals

* $\mathsf{N}(\mathcal{M})$, **negative indices**: all ends of intervals



$[b_1, d_1], [b_2, d_2], [b_3, d_3]$ have representatives

$$\Downarrow$$

$$\mathsf{Pers}(M) = \{[b_1, d_1], [b_2, d_2], [b_3, d_3]\}$$

Correctness proof

Theorem

The algorithm computes the 0-th barcode for a given zigzag filtration.

Algorithm for 1-dimension

Algorithm for 1-dimension: pairing

$\mathcal{U}_i$: unpaired positive indices

$\mathcal{U}_0 := \varnothing$

for $i := 0, \ldots, m-1$:

  if $G_i \xrightarrow{\sigma_i} G_{i+1}$ provides positive index $i+1$:

    $\mathcal{U}_{i+1} := \mathcal{U}_i \cup \{i+1\}$

  else if $G_i \xleftarrow{\sigma_i} G_{i+1}$ provides negative index i:

    pair i with a $j_* \in \mathcal{U}_i$ based on the Pairing Principle

    output interval $[j_*, i]$

    $\mathcal{U}_{i+1} := \mathcal{U}_i \setminus \{j_*\}$

  else:

    $\mathcal{U}_{i+1} := \mathcal{U}_i$

for each $j \in \mathcal{U}_m$:

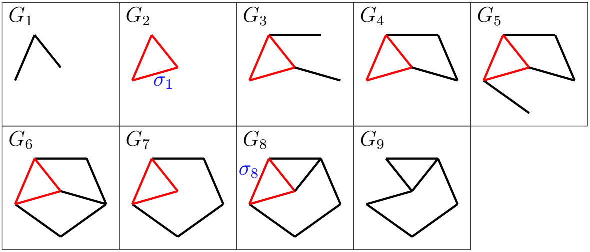
  output interval $[j, m]$

Pairing Principle

For each iteration providing negative index i, let J_i consist of every $j \in \mathcal{U}_i$ s.t. $\exists$ 1-cycle z:

- $z \subseteq G_k$ *for every* $k \in [j, i]$
- z *contains both* σ_{j-1} *and* σ_i

Then, $J_i \neq \varnothing$ *and we pair i with the smallest index* j_* *in* J_i.



$\mathcal{U}_8 = \{2, 6, 8\}$

$J_8 = \{2, 6, 8\}$

Interval: $[2, 8]$

Implementing Pairing Principle

- Reduce the pairing to finding the maximum weight of edges on a path in a minimum spanning forest (details in paper)
- Use Dynamic MSF data structure [Holm, De Lichtenberg, Thorup 2001] to achieve the complexity

Thank You