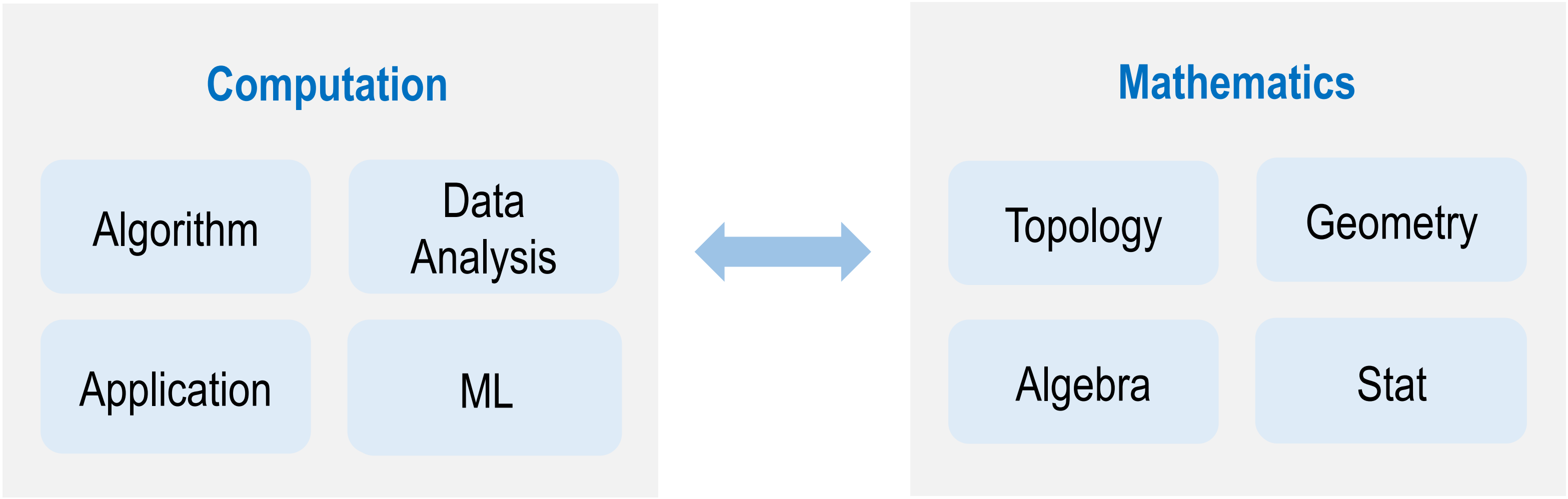
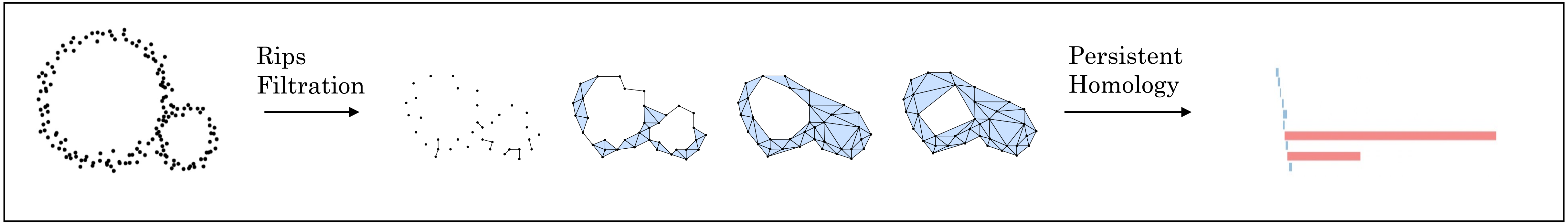
Fast Computation of Zigzag Persistence

Tao Hou
School of Computing, DePaul University
ESA 2022

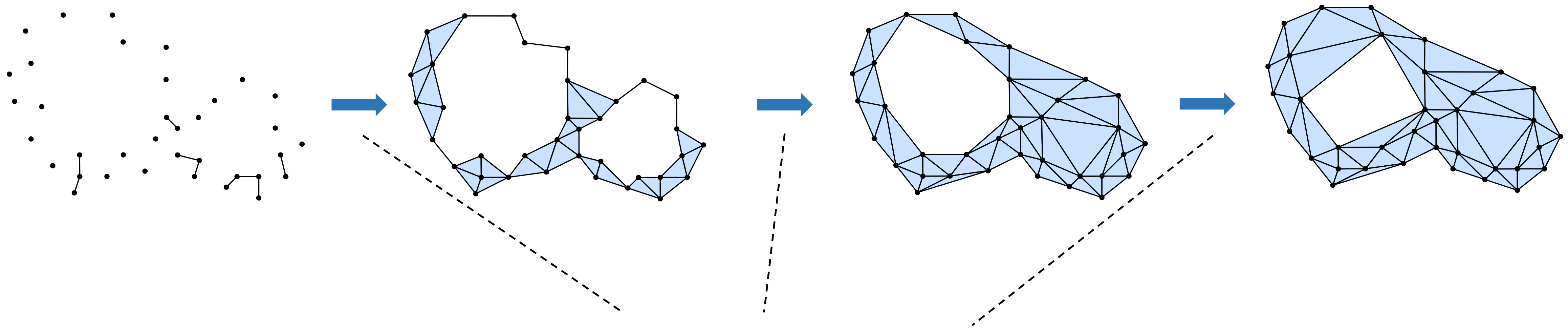
Joint work with **Tamal K. Dey**

Topological data analysis (TDA)





Persistent homology



Expand each arrow into a sequence of additions of a single simplex

Filtration: a sequence of additions of a single simplex

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

Standard persistence: Pipeline

Standard 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:

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

$$\Downarrow$$

Interval decomposition: **[Gabriel 72]**

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

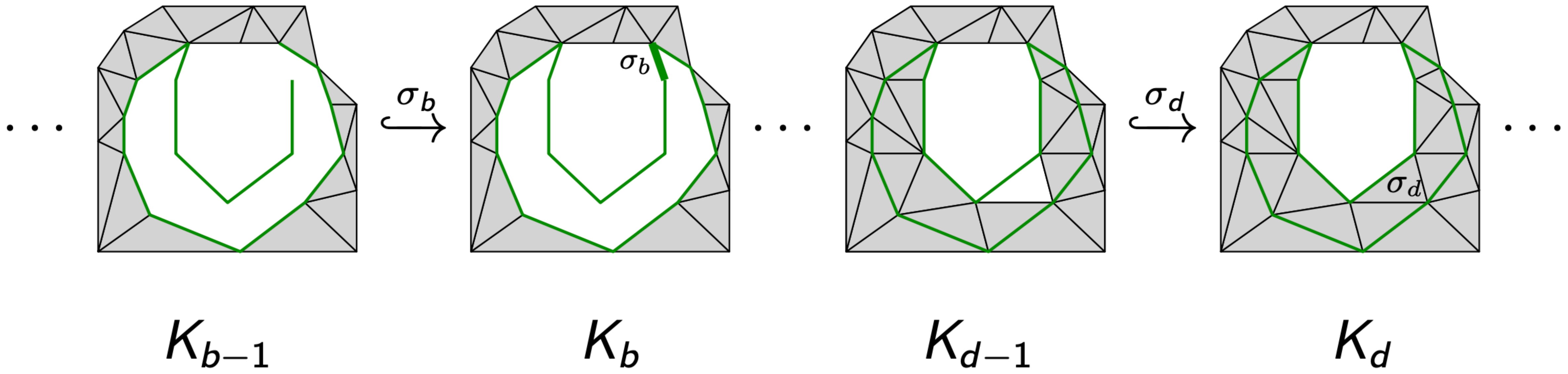
$$\Downarrow$$

p-th persistence barcode:

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

starts and ends with **indices in the filtration**

Persistent homology: example



An interval: $[b, d-1]$

Standard persistence

Standard filtration:

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

$$\Downarrow$$

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: **[Gabriel 72]**

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

$$\Downarrow$$

p-th persistence barcode:

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

Zigzag persistence

Zigzag filtration:

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

$$K_i \overset{\sigma_i}{\hookrightarrow} K_{i+1} \text{ or } K_i \overset{\sigma_i}{\hookleftarrow} K_{i+1}$$

Gunnar Carlsson and Vin de Silva. **Zigzag persistence**.
Foundations of Computational Mathematics, 10(4):367–405, 2010.

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:

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

$$\Downarrow$$

Interval decomposition: **[Gabriel 72]**

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

$$\Downarrow$$

p-th persistence barcode:

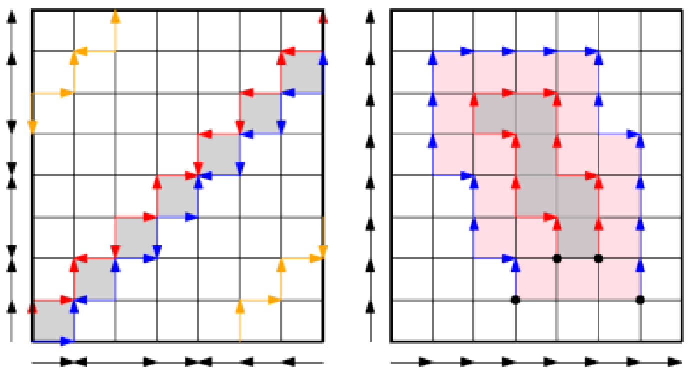
$$\mathsf{Pers}_p(\mathcal{F}) = \{[b_\alpha, d_\alpha] \mid \alpha \in \mathcal{A}\}$$

Peter Gabriel. Unzerlegbare Darstellungen I. Manuscripta Mathematica, 6(1):71–103, 1972.

Applications of Zigzag persistence

- In time varying settings: functions, point cloud, vector field
  - G. Carlsson, V. de Silva, and D. Morozov. Zigzag persistent homology and real-valued functions. SoCG 2009.
  - W. Kim and F. Mémoli. Spatiotemporal persistent homology for dynamic metric spaces. DCG 2020.
  - T. Dey, M. Lipinsky, M. Mrozek, R. Slechta. Tracking dynamical features via continuation and persistence. SoCG 2022.

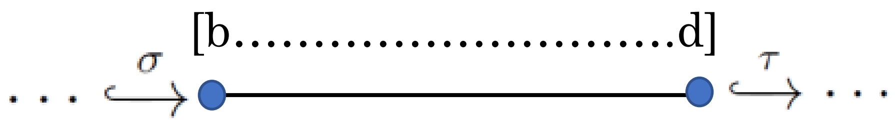
- In multiparameter persistence



Non-Zigzag vs. Zigzag persistence

Bars in non-zigzag: 1 type

- closed-open



Bars in zigzag: 4 types

- closed-open
- closed-closed
- open-closed
- open-open

Simplices(σ) in zigzag: insertion($\downarrow\sigma$), deletion($\uparrow\sigma$), repeated($\downarrow\sigma$)

$$\mathcal{F}: \varnothing = K_0 \leftrightarrow \cdots \xhookrightarrow{\downarrow\sigma} \cdots \xhookleftarrow{\uparrow\sigma} \cdots \xhookrightarrow{\downarrow\sigma} \cdots \leftrightarrow K_m = \varnothing$$

Complexities of persistence computing

	Theoretical	In Practice
Standard	$O(m^\omega)$	*Various optimizations*
Zigzag	$O(m^\omega)$	*Much slower*

$\omega \approx 2.37286$, matrix multiplication exponent

- Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. **Topological persistence and simplification**. 2000.
- Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. **Zigzag persistent homology and real-valued functions**. 2009.
- Nikola Milosavljevi´c, Dmitriy Morozov, and Primoz Skraba. **Zigzag persistent homology in matrix multiplication time**. 2011.

Overview of FastZigzag

- Input zigzag filtration

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

- Convert to a non-zigzag filtration of same length
  - *Linear time* • *Very Fast*

$$\mathcal{F}' : K'_0 \overset{\sigma'_0}{\hookrightarrow} K'_1 \overset{\sigma'_1}{\hookrightarrow} \cdots \overset{\sigma'_{m-1}}{\hookrightarrow} K'_m$$

- Compute barcode for non-zigzag filtration $\mathcal{F}'$
  - *Fast software [Gudhi, Phat, Dionysus etc.]*

- Convert barcode of $\mathcal{F}'$ to that of $\mathcal{F}$
  - $O(1)$ *conversion per bar*

Conversion in FastZigzag

1. Convert input zigzag to a non-repetitive zigzag filtration of same length
2. Convert non-repetitive zigzag to an up-down filtration of same length
3. Convert up-down filtration to an extended filtration of same length
4. Convert extended filtration to a non-zigzag filtration of same length

Non-repetitive filtration: A simplex is added at most one time

$$\mathcal{F} : \varnothing = K_0 \leftrightarrow \cdots \xhookrightarrow{\sigma} \cdots \xhookleftarrow{\sigma} \cdots \xhookrightarrow{\sigma} \cdots \leftrightarrow K_m = \varnothing$$

Repetitive

Conversions 1,2,3,4:

- Done by a simple linear scan of the input filtration

  Linear time, Very Fast
- Simple mapping rules (bijections) for barcodes

1. Input $\Longrightarrow$ Non-repetitive (same length)

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

$$\Downarrow$$

$$\hat{\mathcal{F}} : \varnothing = \hat{K}_0 \xleftrightarrow{\hat{\sigma}_0} \hat{K}_1 \xleftrightarrow{\hat{\sigma}_1} \cdots \xleftrightarrow{\hat{\sigma}_{m-1}} \hat{K}_m = \varnothing$$

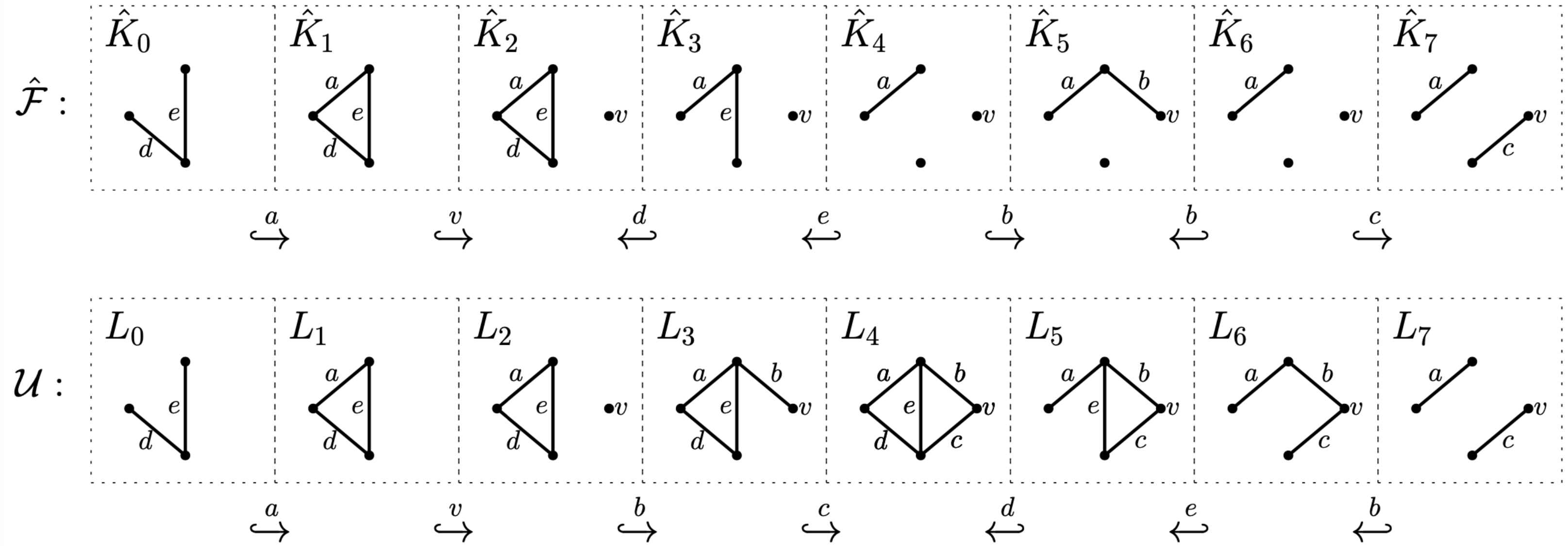
- Idea: Treat each repeatedly added simplex as a new copy
- Barcodes stay the same
- Details given later

2. Non-repetitive $\Longrightarrow$ Up-down (same length)

$$\hat{\mathcal{F}} : \varnothing = \hat{K}_0 \xleftrightarrow{\hat{\sigma}_0} \hat{K}_1 \xleftrightarrow{\hat{\sigma}_1} \cdots \xleftrightarrow{\hat{\sigma}_{m-1}} \hat{K}_m = \varnothing$$

$$\mathcal{U} : \varnothing = L_0 \xhookrightarrow{\tau_0} \cdots \xhookrightarrow{\tau_{n-1}} L_n \xhookleftarrow{\tau_n} \cdots \xhookleftarrow{\tau_{2n-1}} L_{2n} = \varnothing \qquad (m = 2n)$$

List the additions in $\hat{\mathcal{F}}$ first and then the deletions in $\hat{\mathcal{F}}$, following the orders in $\hat{\mathcal{F}}$



2. Non-repetitive $\Longrightarrow$ Up-down

Mayer-Vietoris Diamond [Carlsson, de-Silva, 2010]

$$\mathcal{F}_1: \quad K_0 \longleftrightarrow \cdots \longleftrightarrow K_{j-1} \xhookrightarrow{\sigma} K_j \xhookleftarrow{\tau} K_{j+1} \longleftrightarrow \cdots \longleftrightarrow K_m$$

$$\mathcal{F}_2: \quad K_0 \longleftrightarrow \cdots \longleftrightarrow K_{j-1} \xhookleftarrow{\tau} K_j' \xhookrightarrow{\sigma} K_{j+1} \longleftrightarrow \cdots \longleftrightarrow K_m$$

- $\mathcal{F}_1$ to $\mathcal{F}_2$: Outward switch
- $\mathcal{F}_2$ to $\mathcal{F}_1$: Inward switch

Proposition. [Dey-Hou, 2022]
An up-down filtration $\mathcal{U}$ can be derived from the non-repetitive filtration $\hat{\mathcal{F}}$ by a sequence of inward switches s.t.

- $\exists$ a bijection between $\mathsf{Pers}_*(\hat{\mathcal{F}})$ and $\mathsf{Pers}_*(\mathcal{U})$

Barcode bijection between $\mathcal{U}$ and $\hat{\mathcal{F}}$

Simple takeaway: Corresponding intervals have **same** creator and destroyer simplices

$$[b \quad \cdots\cdots\cdots \quad d]$$

$$\mathcal{U} : L_0 \hookrightarrow L_1 \xhookrightarrow{\sigma} L_2 \hookrightarrow L_3 \hookrightarrow L_4 \hookleftarrow L_5 \hookleftarrow L_6 \xhookleftarrow{\tau} L_7 \hookleftarrow L_8$$

$$\downarrow$$

$$[b' \quad \cdots\cdots \quad d']$$

$$\hat{\mathcal{F}} : \hat{K}_0 \hookrightarrow \hat{K}_1 \hookleftarrow \hat{K}_2 \xhookrightarrow{\sigma} \hat{K}_3 \hookleftarrow \hat{K}_4 \hookrightarrow \hat{K}_5 \xhookleftarrow{\tau} \hat{K}_6 \hookrightarrow \hat{K}_7 \hookleftarrow \hat{K}_8$$

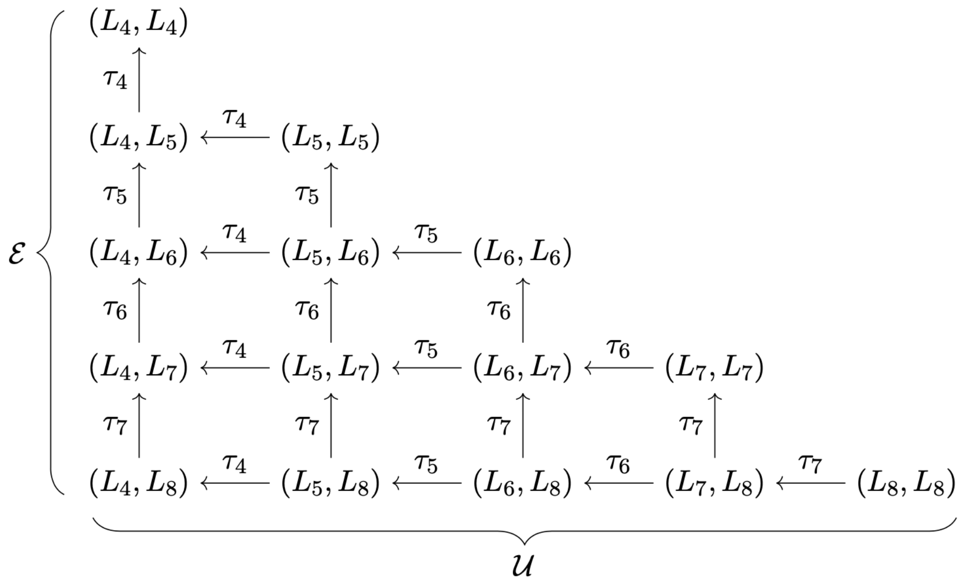
3. Up-down ⟹ Extended (same length)

$$\mathcal{U} : \varnothing = L_0 \overset{\tau_0}{\hookrightarrow} \cdots \overset{\tau_{n-1}}{\hookrightarrow} L_n \overset{\tau_n}{\hookleftarrow} \cdots \overset{\tau_{2n-1}}{\hookleftarrow} L_{2n} = \varnothing$$

⬇

$$\mathcal{E} : \varnothing = L_0 \hookrightarrow \cdots \hookrightarrow L_n = (\hat{K}, L_{2n}) \hookrightarrow (\hat{K}, L_{2n-1}) \hookrightarrow \cdots \hookrightarrow (\hat{K}, L_n) = (\hat{K}, \hat{K})$$

- Use Mayer-Vietoris Pyramid [CdSM09]



- Interval mapping: Corresponding intervals have **same** creator and destroyer simplices

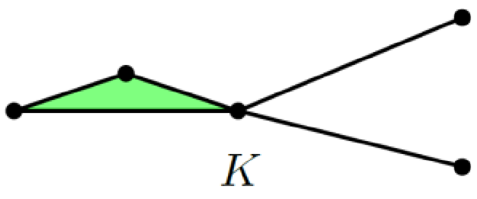
G. Carlsson, V. de Silva, and D. Morozov. Zigzag persistent homology and real-valued functions. SoCG 2009.

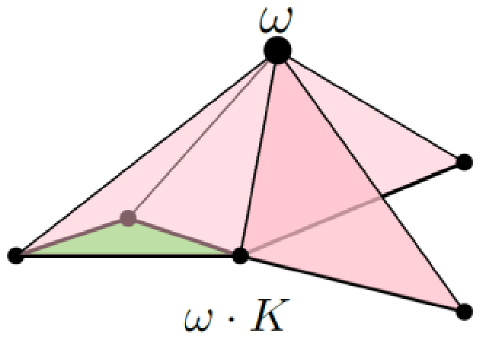
4. Extended $\Longrightarrow$ Non-zigzag (same length)

- Use 'Coning' [CEH06]: No change in barcode

$$\mathcal{E} : \varnothing = L_0 \hookrightarrow \cdots \hookrightarrow L_n = (\hat{K}, L_{2n}) \hookrightarrow (\hat{K}, L_{2n-1}) \hookrightarrow \cdots \hookrightarrow (\hat{K}, L_n) = (\hat{K}, \hat{K})$$

$$\hat{\mathcal{E}} : L_0 \cup \{\omega\} \hookrightarrow \cdots \hookrightarrow L_n \cup \{\omega\} = \hat{K} \cup \omega \cdot L_{2n} \hookrightarrow \hat{K} \cup \omega \cdot L_{2n-1} \hookrightarrow \cdots \hookrightarrow \hat{K} \cup \omega \cdot L_n$$





1. Repetitive ⟹ Non-repetitive (Details)

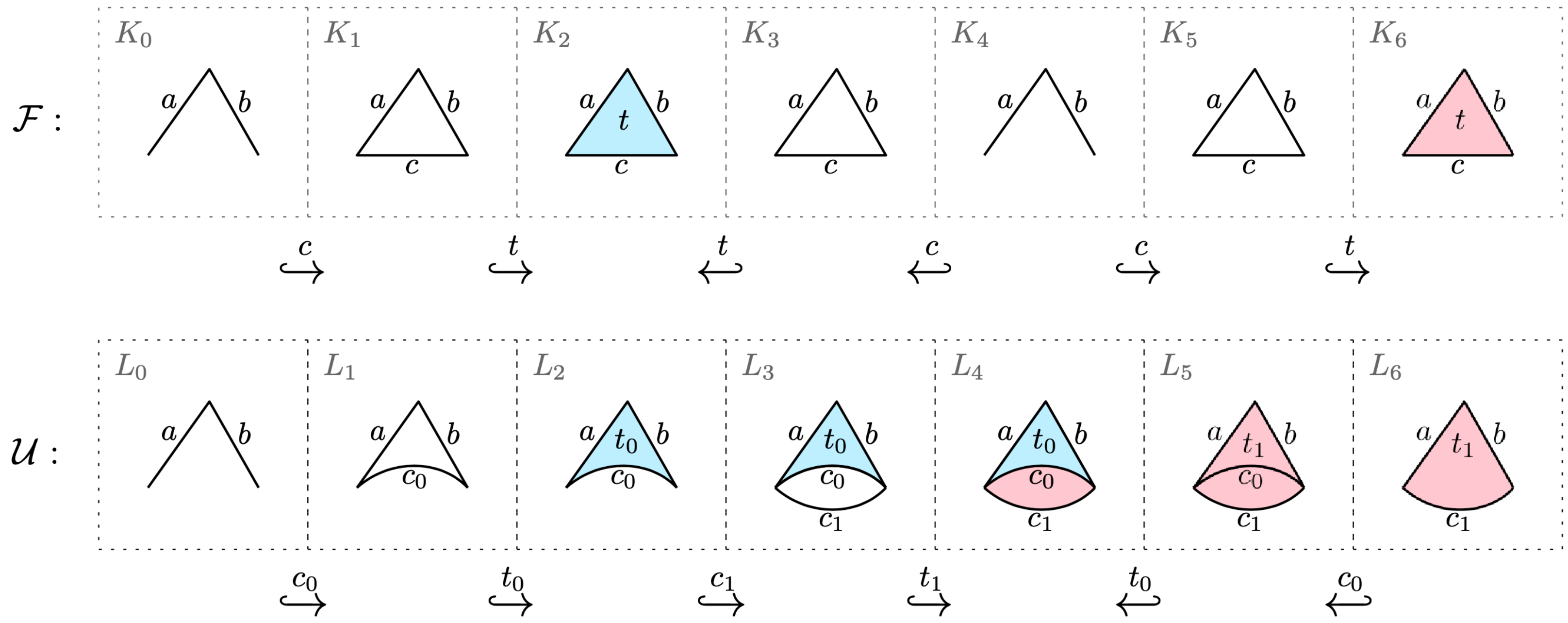
- Treat new occurrence of simplex σ as a new copy

$$\mathcal{F} : \varnothing = K_0 \leftrightarrow \cdots \xhookrightarrow{\sigma} \cdots \xhookleftarrow{\sigma} \cdots \xhookrightarrow{\sigma} \cdots \xhookleftarrow{\sigma} \cdots \leftrightarrow K_m = \varnothing$$

$$\Downarrow$$

$$\hat{\mathcal{F}} : \varnothing = \hat{K}_0 \leftrightarrow \cdots \xhookrightarrow{\hat{\sigma}_1} \cdots \xhookleftarrow{\hat{\sigma}_1} \cdots \xhookrightarrow{\hat{\sigma}_2} \cdots \xhookleftarrow{\hat{\sigma}_2} \cdots \leftrightarrow \hat{K}_m = \varnothing$$

- Copies of same simplex shall occur in same complex in up-down: use Δ-complex [Hatcher02]



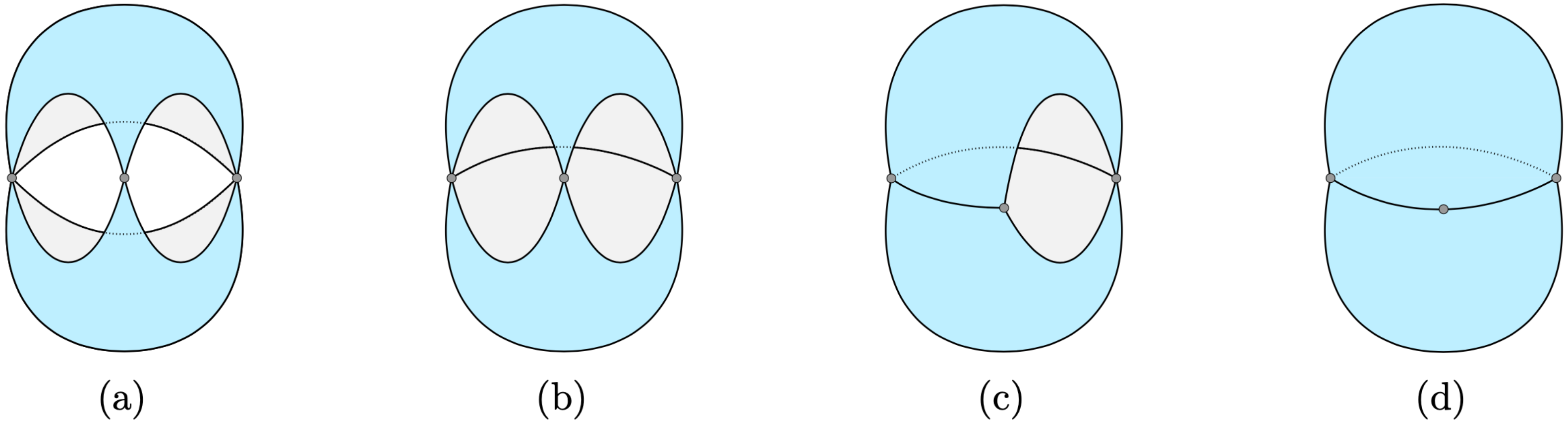
Δ-complex

- Building blocks: **Cells**

Definition. A Δ-complex is defined recursively with dimension:

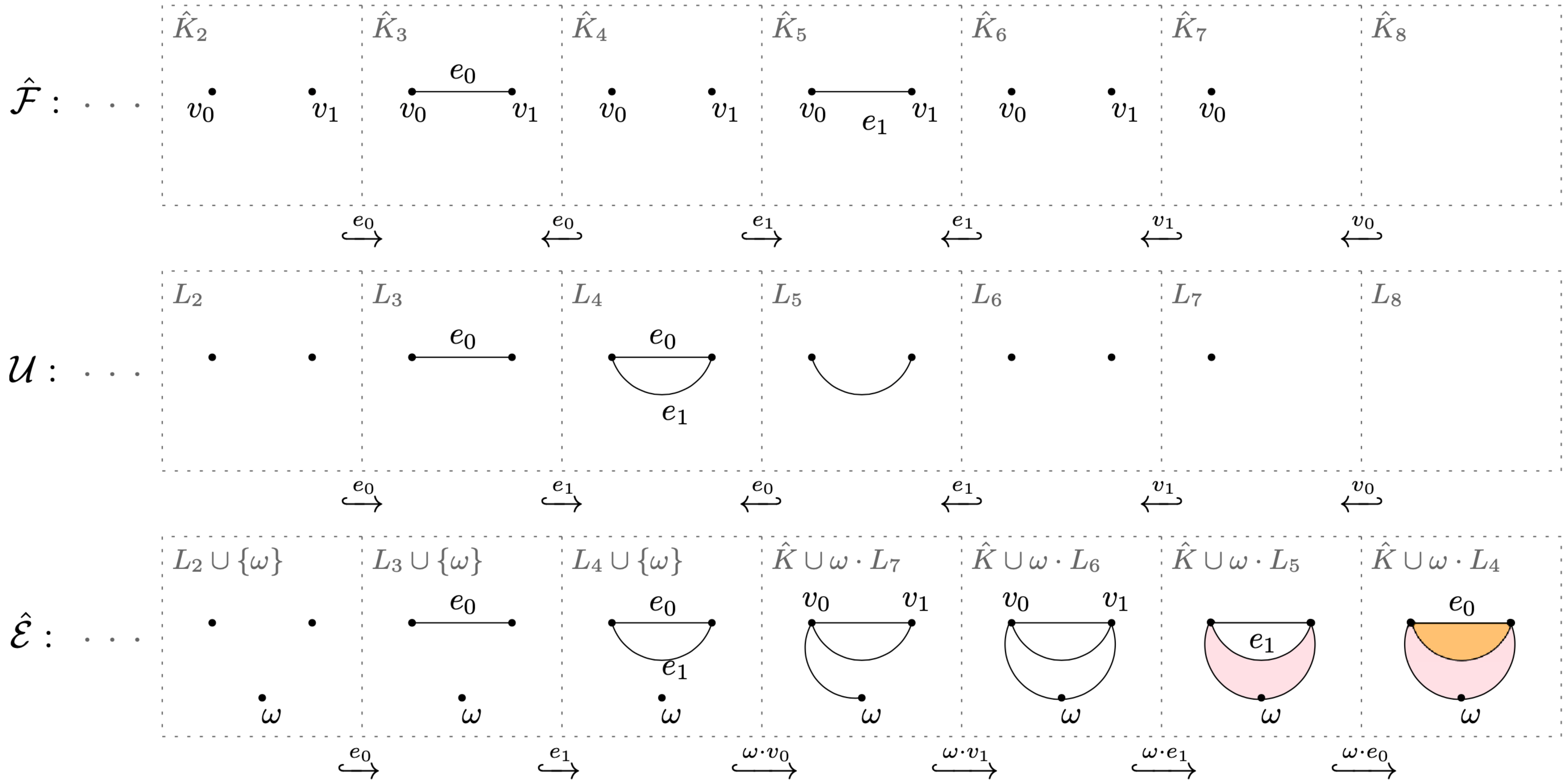
1. Δ-complex K^0 is a set of points, each called a 0-cell.

2. Δ-complex K^p, $p \geq 1$, is a quotient space of a Δ-complex K^{p-1} with standard p-simplices where quotienting is realized by a homeomorphism $h : \partial(\sigma) \to K^{p-1}$ that maps each proper face of σ onto a cell in K^{p-1}.



Two triangles sharing 0,1,2,3 edges

Overall Conversions



Software **FZZ** using **Phat** software for non-zigzag (https://github.com/taohou01/fzz)

FZZ vs. Dionysus2, Gudhi

No.	Length	D	Rep	MaxK	T_{Dio2}	T_{GUDHI}	T_{FZZ}	SU
1	5,260,700	5	1.0	883,350	2h02m46.0s	–	8.9s	873
2	5,254,620	4	1.0	1,570,326	19m36.6s	–	11.0s	107
3	5,539,494	5	1.3	1,671,047	3h05m00.0s	45m47.0s	3m20.8s	13.7
4	5,660,248	4	2.0	1,385,979	2h59m57.0s	29m46.7s	4m59.5s	6.0
5	5,327,422	4	3.5	760,098	43m54.8s	10m35.2s	3m32.1s	3.0
6	5,309,918	3	5.1	523,685	5h46m03.0s	1h32m37.0s	19m30.2s	4.7
7	5,357,346	3	7.3	368,830	3h37m54.0s	57m28.4s	30m25.2s	1.9
8	6,058,860	4	9.1	331,211	53m21.2s	7m19.0s	3m44.4s	2.0
9	5,135,720	3	21.9	11,859	23.8s	15.6s	8.6s	1.9
10	5,110,976	3	27.7	11,435	36.2s	39.9s	8.5s	4.3
11	5,811,310	4	44.2	7,782	38.5s	36.9s	23.9s	1.5

All run on Intel(R), Core™, i5-9500 CPU@3.00GHz, 16GB memory, Linux OS

Thank you!