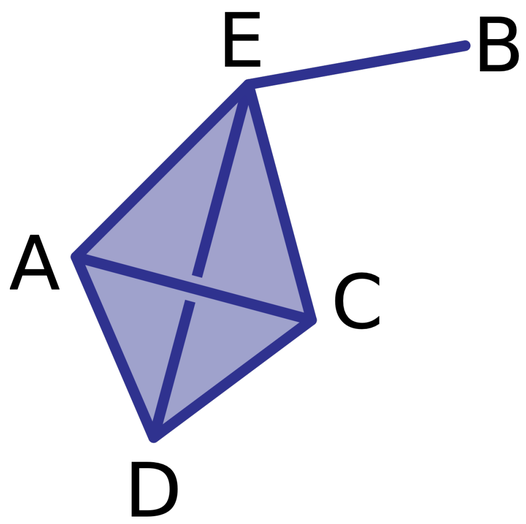
Directional Transform

Dr. Elena Xinyi Wang

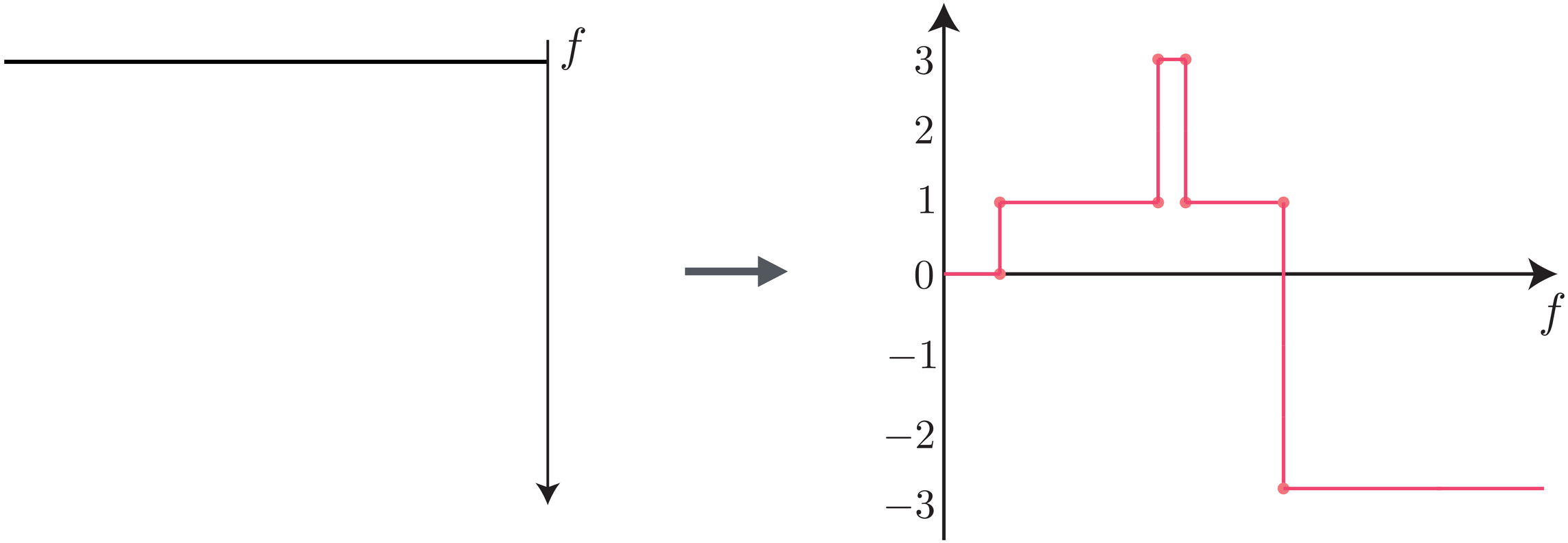
November 18, 2026

Generalized Euler Characteristic

For any simplicial complex K where n_p is the number of p-dimensional simplices, the Euler characteristic is $\chi(K) = \sum_p (-1)^p n_p$



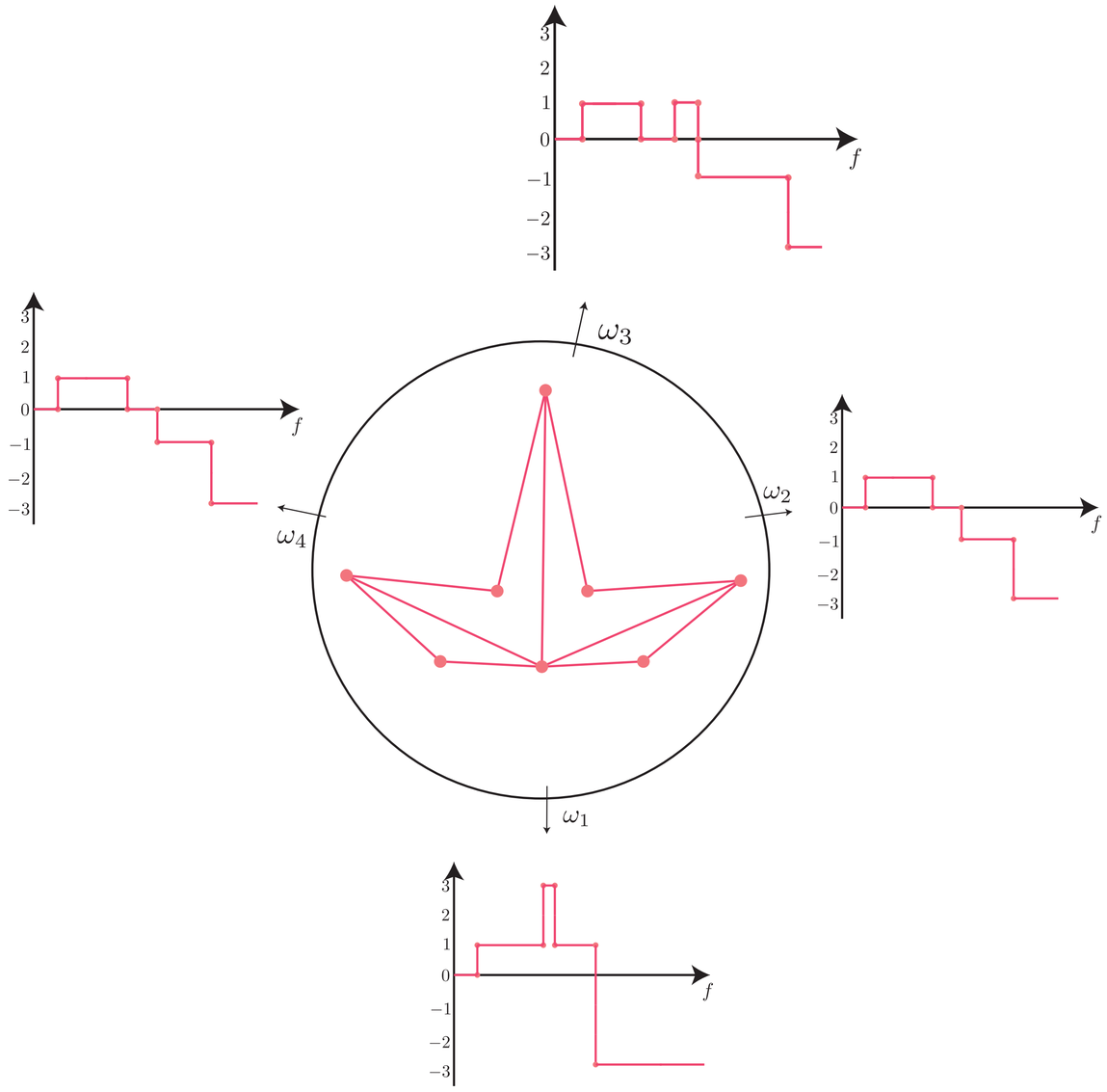
The Euler Characteristic as a Function



Euler Characteristic Curve: $\mathbb{R} \to \mathbb{Z}$

$$a \mapsto \chi(K_a)$$

Euler Characteristic Transform (ECT)



Data: $\boldsymbol{D} \subseteq \mathbb{R}^2$ and

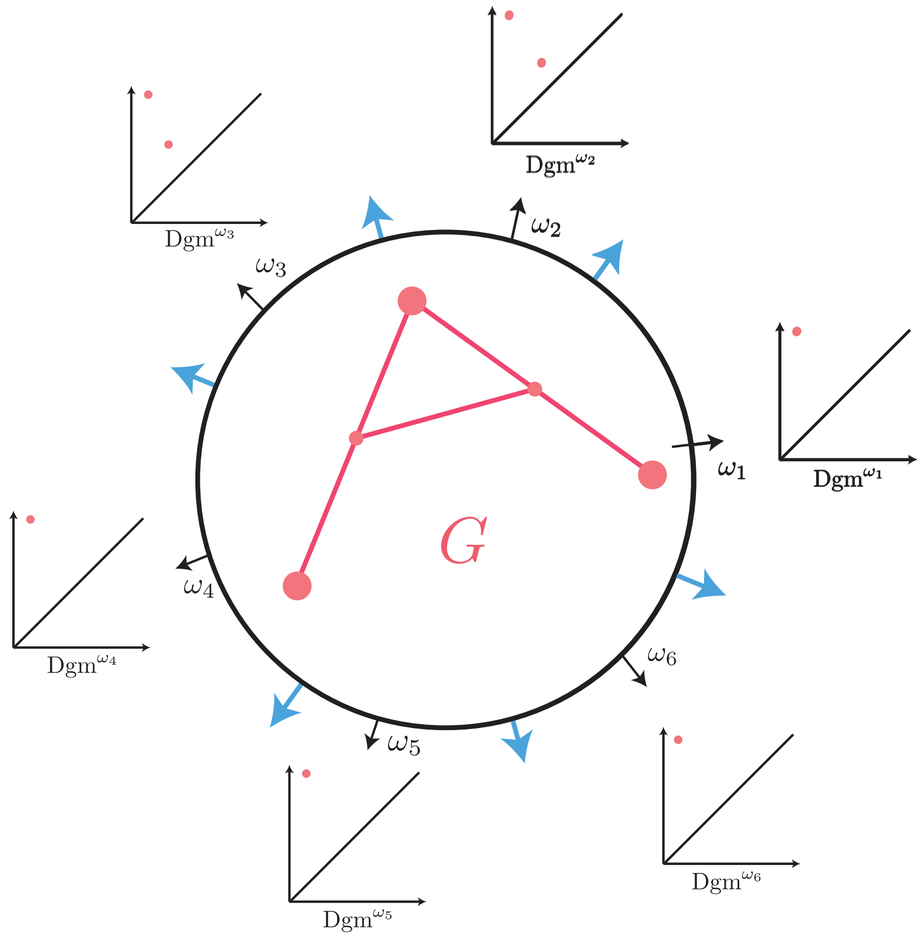
Direction $\boldsymbol{\omega} \in \mathbb{S}^1$

$$\mathrm{ECT}(\boldsymbol{D}) : \mathbb{S}^1 \to \mathrm{Func}(\mathbb{R}, \mathbb{Z})$$

$$\omega \mapsto \{a \mapsto \chi(f_\omega(-\infty, a])\}$$

Persistent Homology Transform

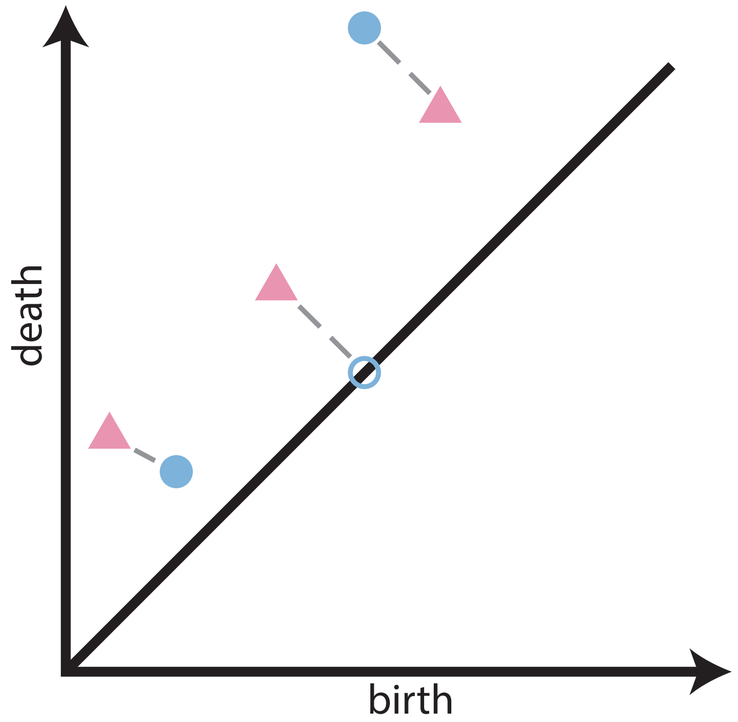
If we compute the persistence diagrams from all directions of $\mathbb{S}^1$, we obtain a family of diagrams (vineyards), called the **Persistent Homology Transform (PHT).**



- Formally, the PHT of $K \subseteq \mathbb{R}^d$ is
  $\mathrm{PHT}(K) : \mathbb{S}^{d-1} \to \mathrm{Dgm}^d$
  $\omega \to (\mathrm{Dgm}_0^{\omega}(K), \mathrm{Dgm}_1^{\omega}(K), \ldots, \mathrm{Dgm}_{d-1}^{\omega}(K))$
- The PHT is **injective**, i.e., if $\mathrm{PHT}(A) = \mathrm{PHT}(B)$, then $A = B$
- The points in PHT are **continuous**

Comparing two Persistence Diagrams

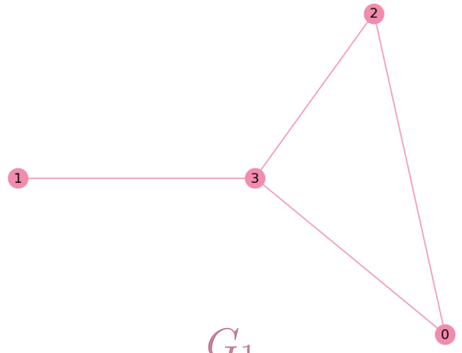
Bottleneck Distance



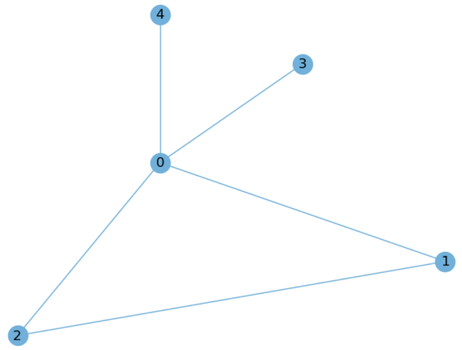
- Given two persistence diagrams X, Y, the **bottleneck distance** is $d_B(X, Y) = \inf_{\eta: X \to Y} \sup_{x \in X} \|x - \eta(x)\|_\infty$
- $\eta : X \to Y$ ranges over all partial bijections

Comparing PHT

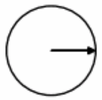
Given two PHTs, can we efficiently maintain the bottleneck distance?

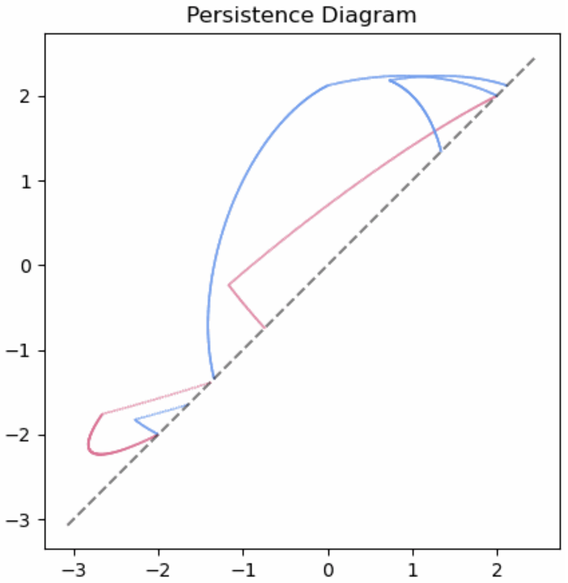


G_1



G_2

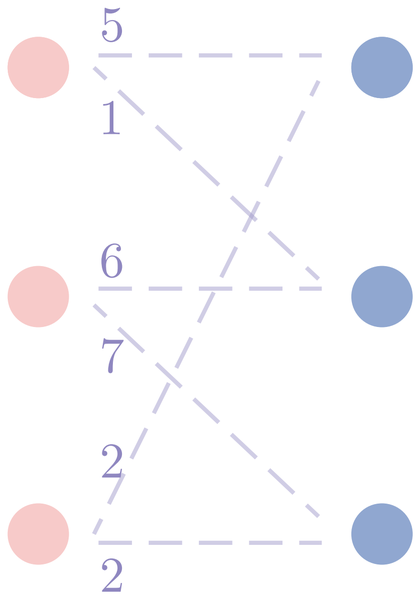


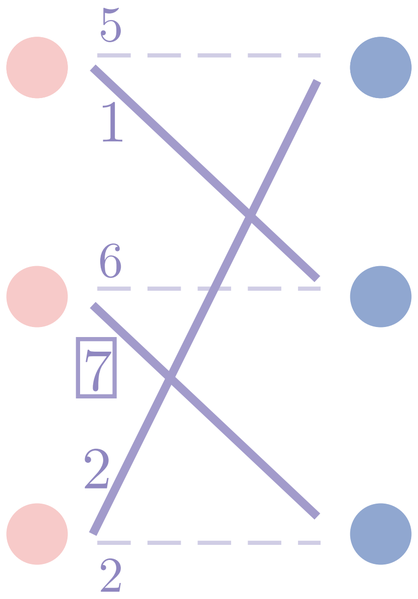


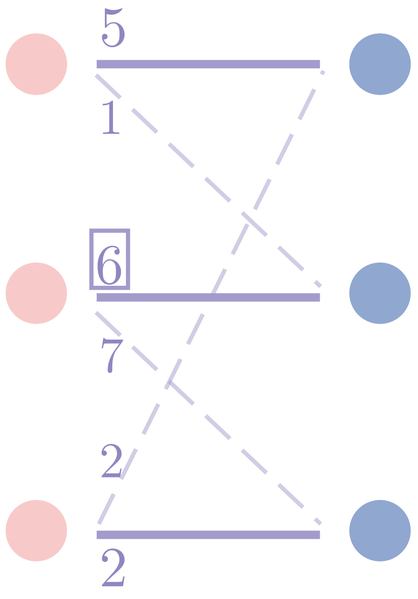
$$d_1(\mathrm{PHT}(G_1), \mathrm{PHT}(G_2)) = \int_0^{2\pi} d_B\left(\mathrm{Dgm}(h_\omega^{G_1}), \mathrm{Dgm}(h_\omega^{G_2})\right) d\omega$$

Graph Matching Problem

- Given a bipartite graph $G = (X \sqcup Y, E)$, where $|X| = |Y| = n$, we can have a **perfect matching** M, where $|M| = n$
- If the edges are weighted with $c : E \mapsto \mathbb{R}_+$, the **bottleneck cost** is $\max\{c(e) \mid e \in M\}$
- A perfect matching is **optimal** if its cost is minimal



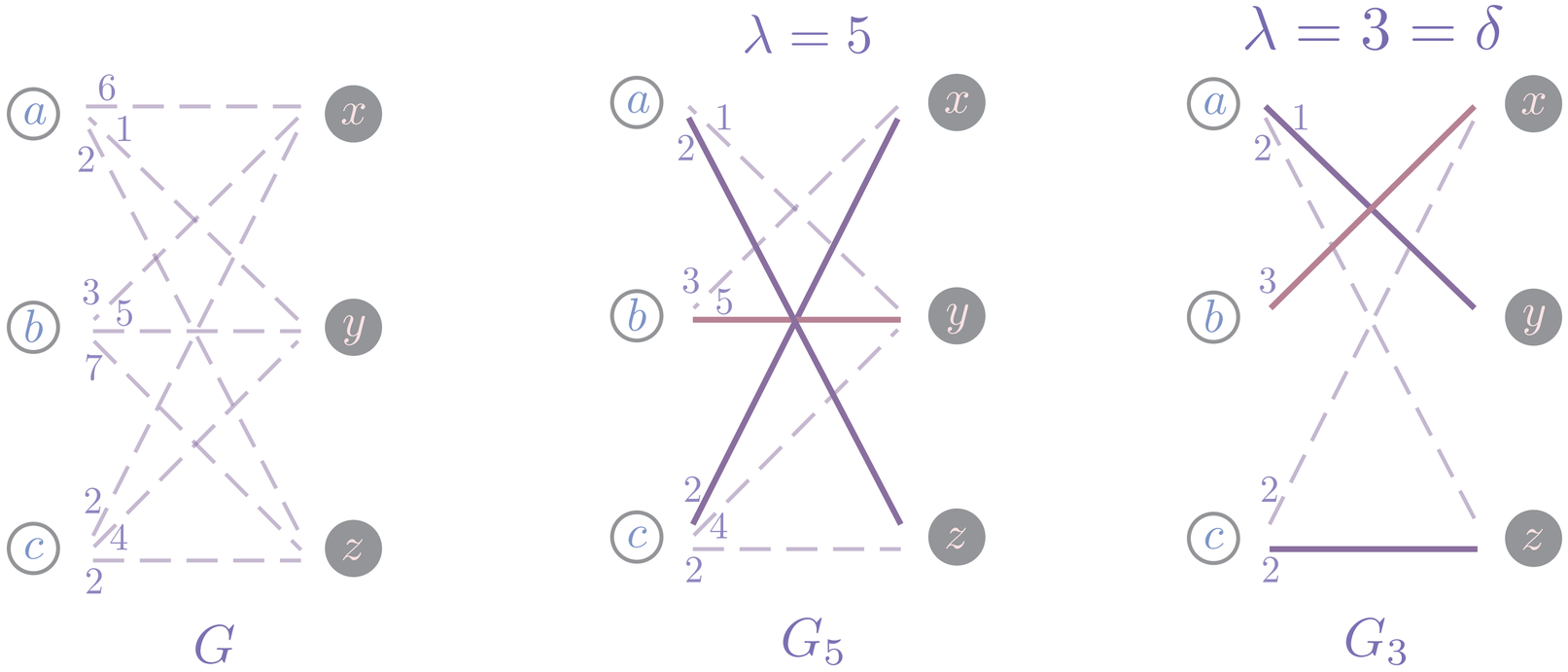




Bottleneck Cost

- Given a weighted complete bipartite graph, we consider its subgraph containing edges with weights at most λ: $G_\lambda = (X \sqcup Y, \{xy \mid c(xy) \leq \lambda\})$

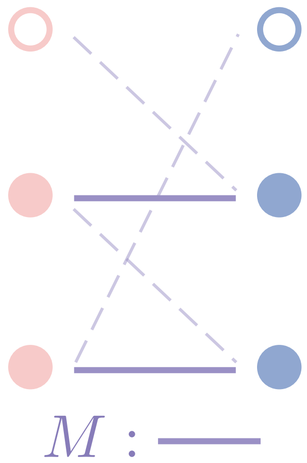
- The **bottleneck cost** $\delta(G_\lambda)$ is the minimum λ such that G_λ has a perfect matching

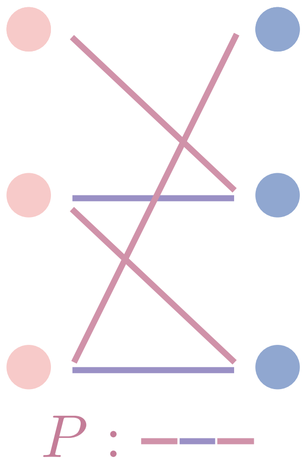


Augmenting Path

For a graph G and a matching M, a path P is called an **augmenting path** for M if:

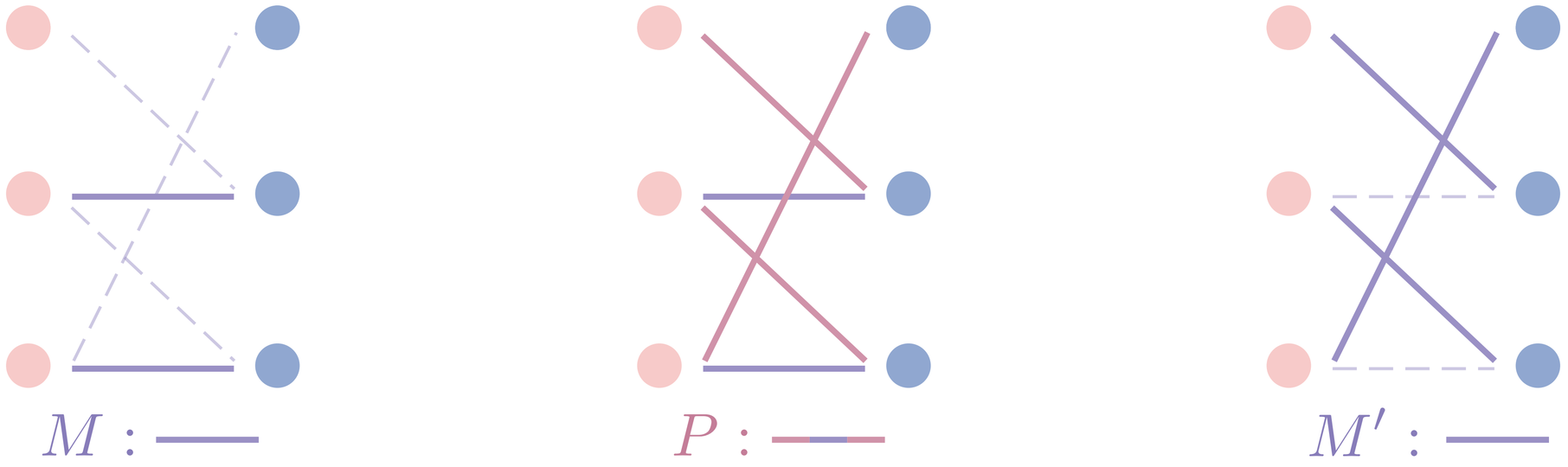
- The two endpoints of P are unmatched in M
- The edges of P alternate between edges $e \in M$ and $e \notin M$





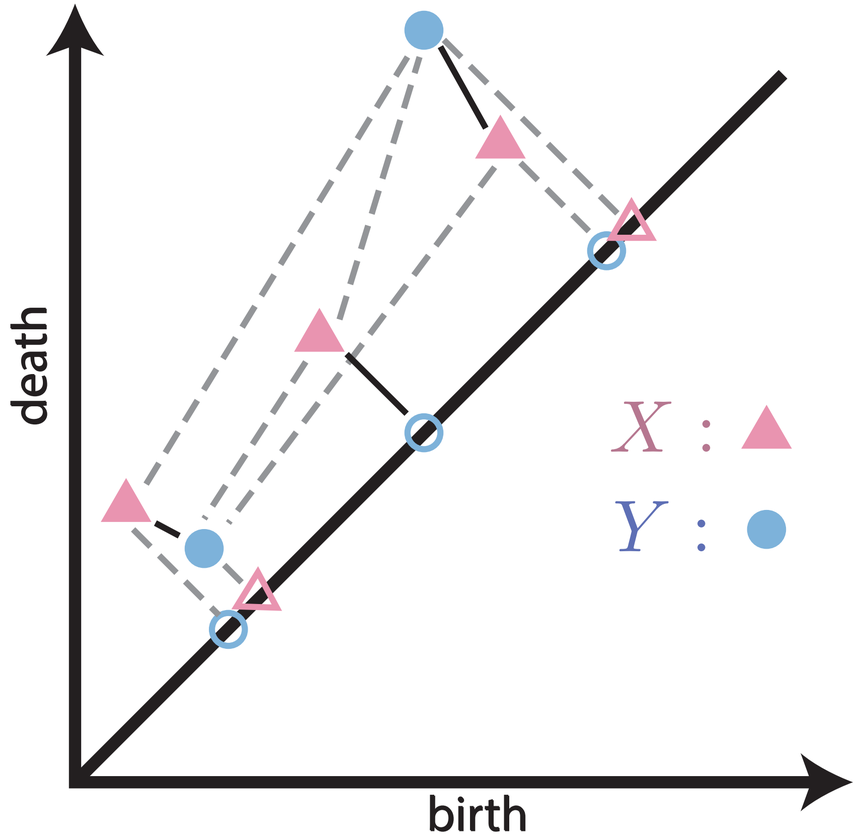
Berge's Theorem

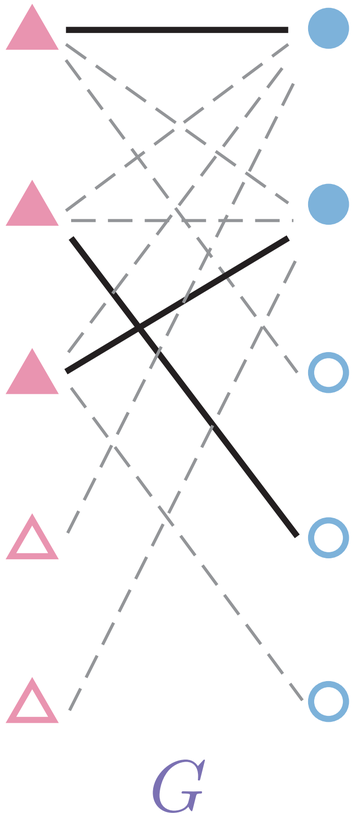
A matching M in a graph G is a maximum matching if and only if G contains no M-augmenting path



In the Context of Persistence

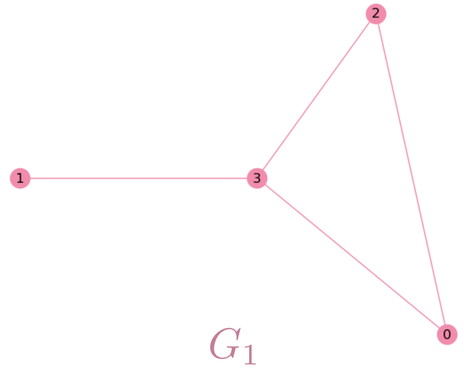
We construct a graph G based on the persistence diagrams X and Y such that the optimal bottleneck cost of G is the bottleneck distance between X and Y



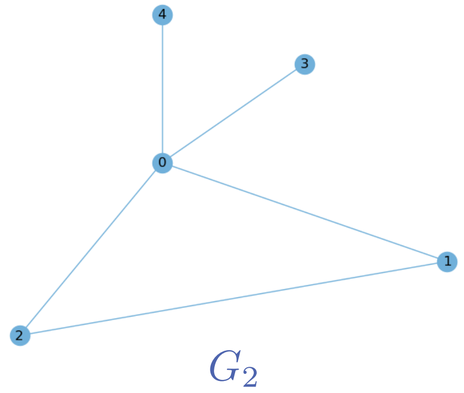


Persistent Homology Transform

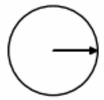
Given two PHTs, can we efficiently maintain the bottleneck distance?

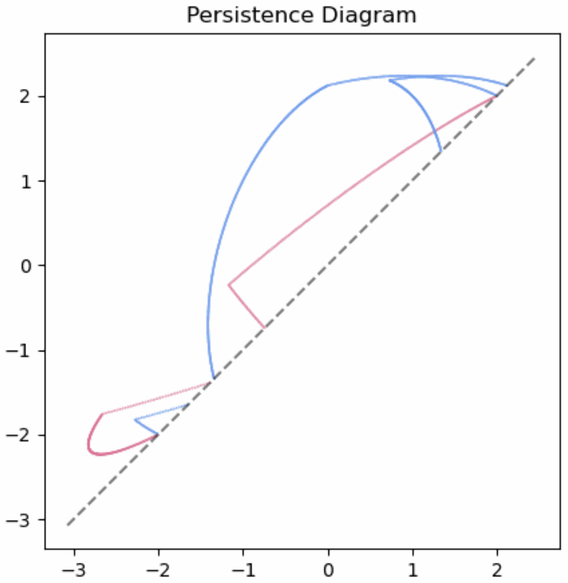


G_1



G_2

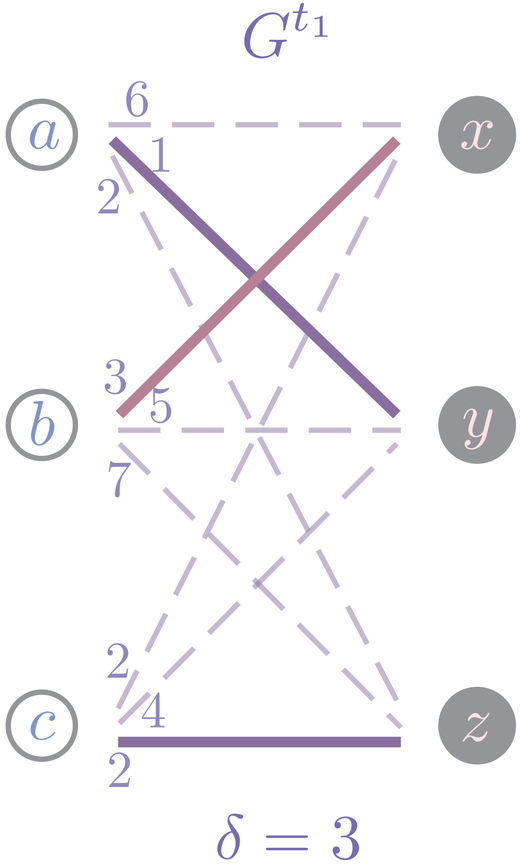


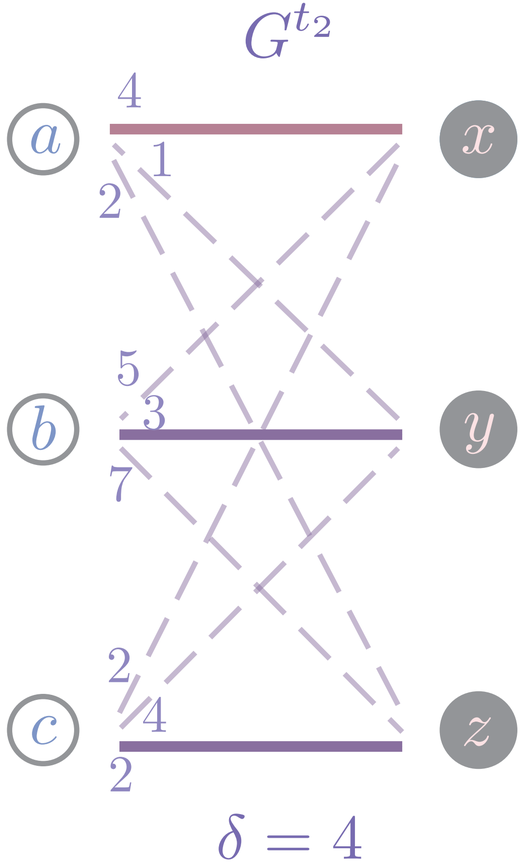


$$d_1(\mathrm{PHT}(G_1), \mathrm{PHT}(G_2)) = \int_0^{2\pi} d_B\left(\mathrm{Dgm}(h_\omega^{G_1}), \mathrm{Dgm}(h_\omega^{G_2})\right) d\omega$$

Bottleneck Distance of Time-Varying Data

How can we maintain the optimal bottleneck cost of G if the weights $c(t)$ change continuously over time t?





The Kinetic Hourglass for Computing the Bottleneck Distance

Proceedings of the 37th Canadian Conference on Computational Geometry (CCCG), 2025. *Best Paper Award*

arXiv:2505.04048



Carola Wenk



Liz Munch

Kinetic Data Structure

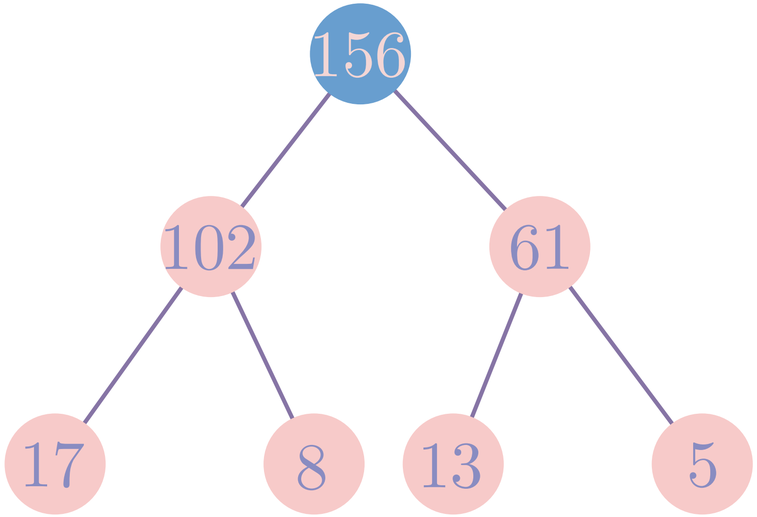
- **Kinetic Data Structure (KDS)** maintains a system of objects v that move continuously
- **Flight Plans:** motions of the objects as functions of time, $v = f(t)$
- **Certificates:** conditions under which the data structure is accurate
- **Events:** certificate failure times and what updates need to be made



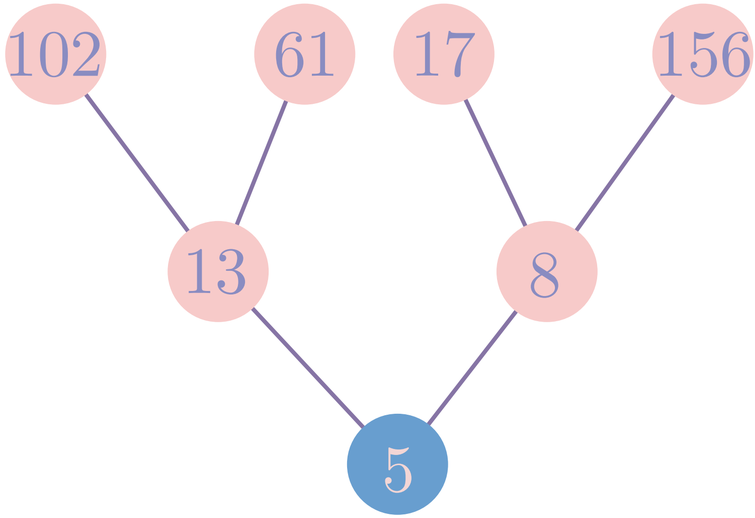
Example: maintaining the maximum of the above five moving points.

Static Heap

- A balanced binary tree
- Value of parent $\geq$ (or $\leq$) value of child
- Root of the tree has the maximum/minimum value



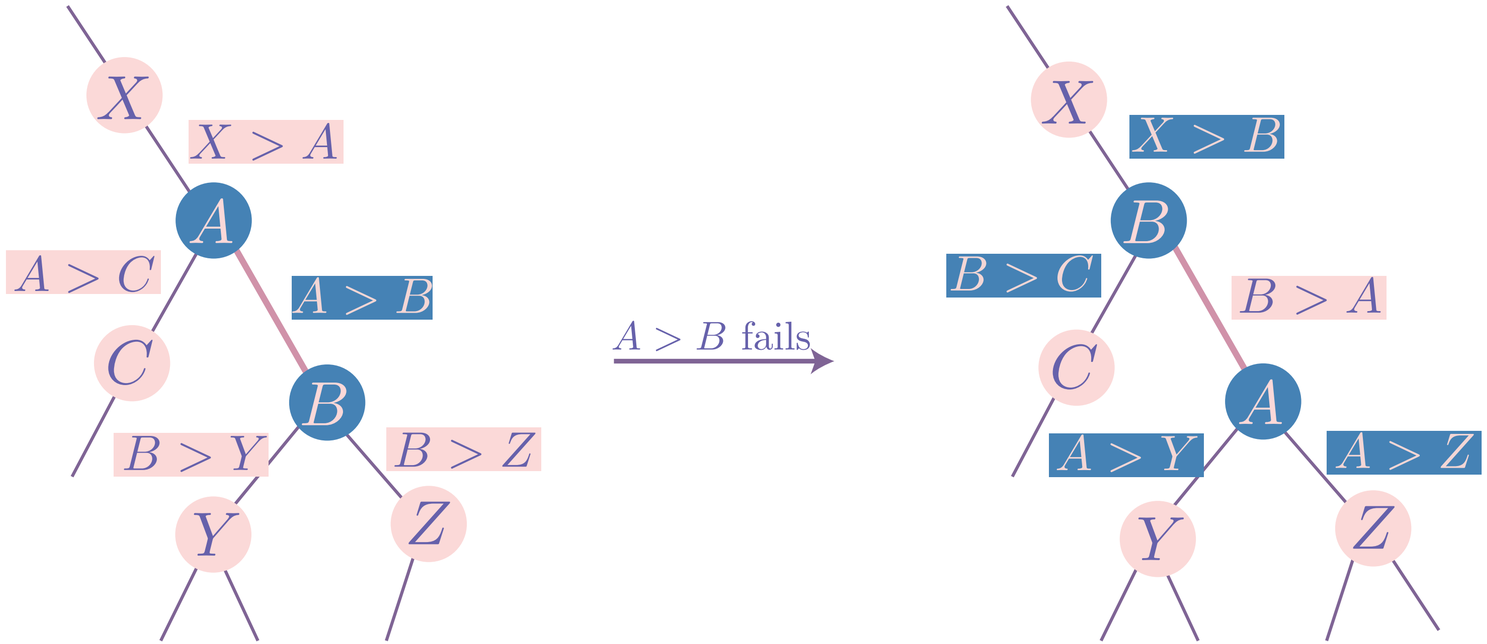
Max heap



Min heap

Kinetic Max Heap

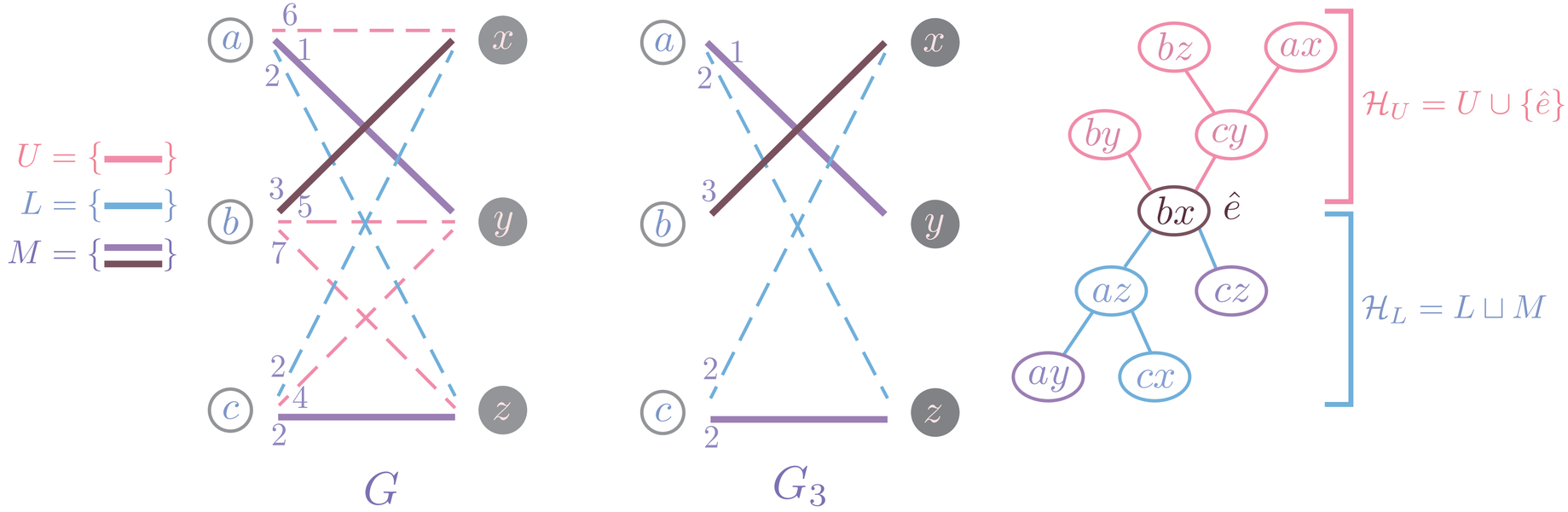
- **Certificate**: every parent-child relationship
- **Event**: priority swap between parent-child
  - Internal event: does not affect the maximum
  - External event: changes the maximum
- **Update**: one event calls for 5 certificate updates



- Improved version: Kinetic Hanger (randomized kinetic heap with better expected runtime)

Kinetic Hourglass

- Given $G = (E, V)$, at time $t = 0$, we have an optimal perfect matching M, and edge $\hat{e} \in M$ that gives the bottleneck distance δ
- $U = \{c(e) > \delta \,|\, e \in E\}$, $L = \{c(e) < \delta \,|\, e \in E \backslash M\}$
- $E = U \sqcup L \sqcup M$
- The **kinetic hourglass** consists of a kinetic max heap, $\mathscr{H}_L$, containing $L \sqcup M$, and a kinetic min heap, $\mathscr{H}_U$, containing $U \cup \{\hat{e}\}$

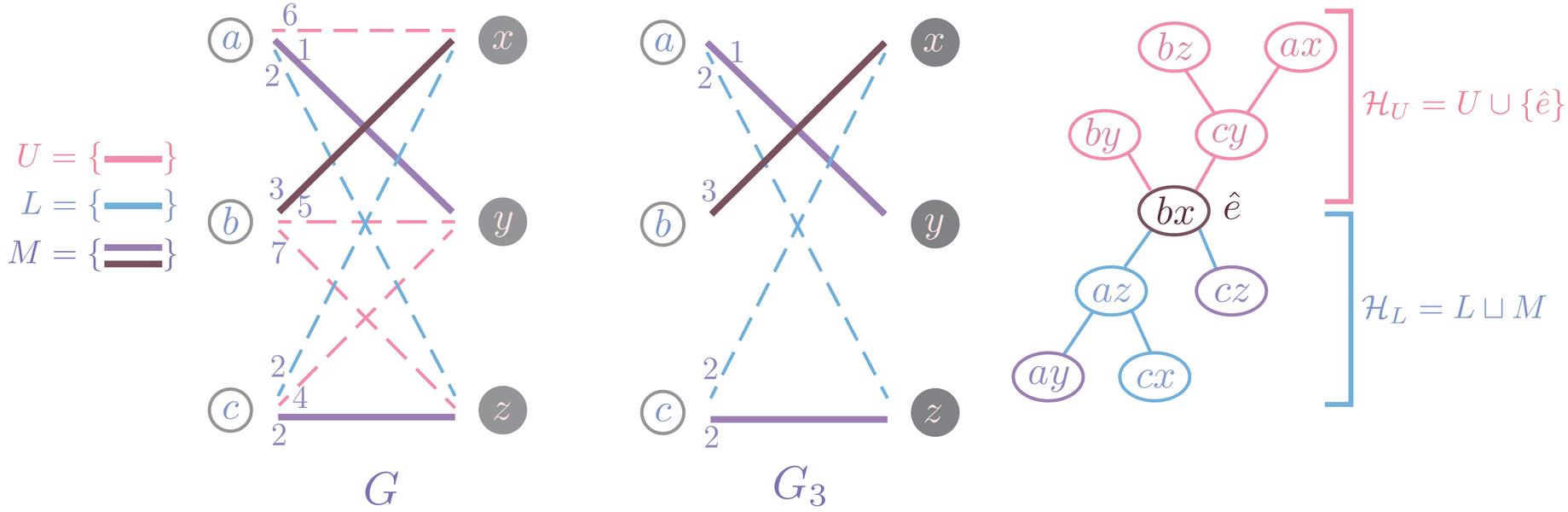


Certificates and Events

The certificates for the kinetic hourglass are the certificates for $\mathscr{H}_L$ and $\mathscr{H}_U$; both heaps have the same root r, and r is $\hat{e}$

External events: change of r and $\hat{e}$

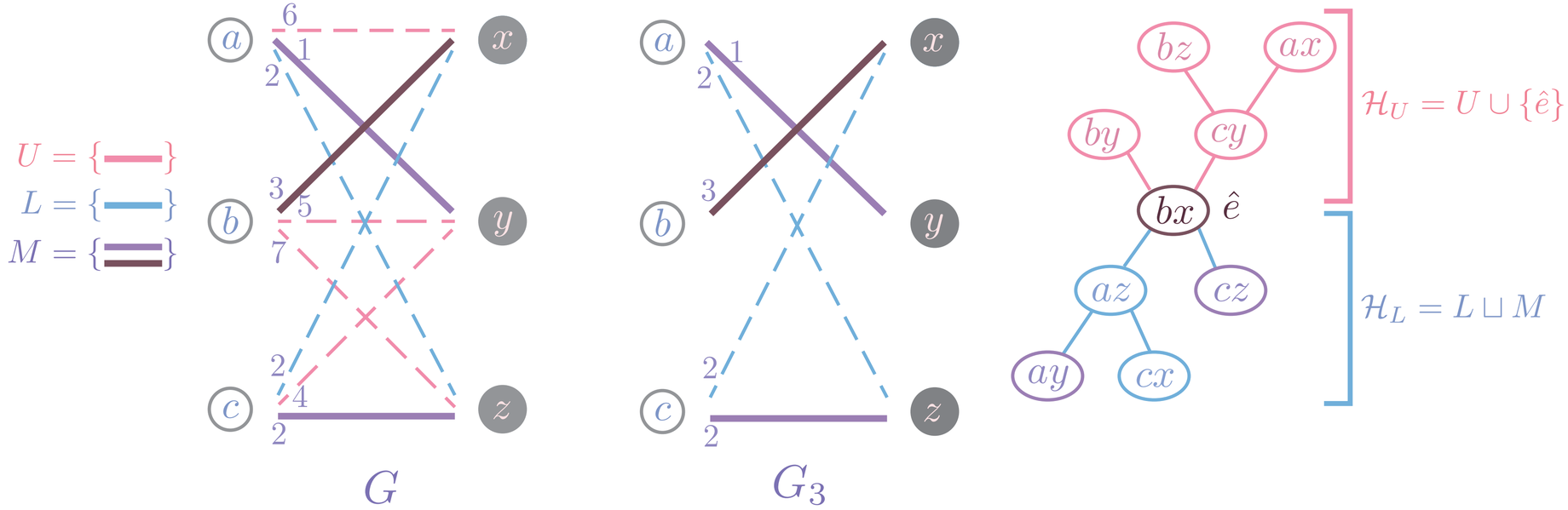
Lemma: Internal event of $\mathscr{H}_L$ or $\mathscr{H}_U$ don't result in external events of kinetic hourglass



External Events

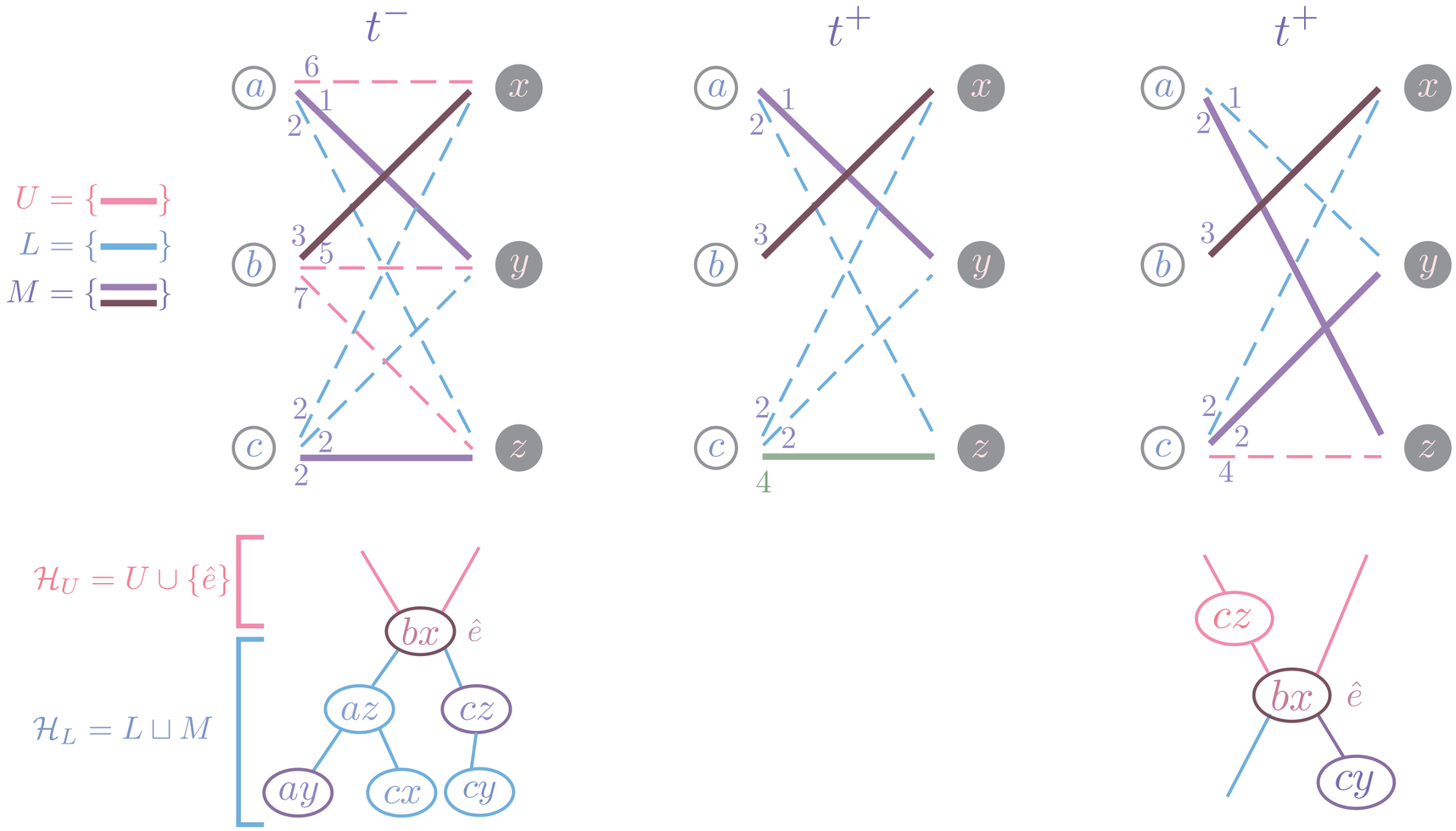
We consider the following three external events of the two heaps

- ($\boldsymbol{L}$-Event) Swap priority of $\boldsymbol{r}$ and an $\boldsymbol{e} \in \boldsymbol{L}^t$ in $\mathscr{H}_L$
- ($\boldsymbol{M}$-Event) Swap priority of $\boldsymbol{r}$ and an $\boldsymbol{e} \in \boldsymbol{M}^t$ in $\mathscr{H}_L$
- ($\boldsymbol{U}$-Event) Swap priority of $\boldsymbol{r}$ and an $\boldsymbol{e} \in \boldsymbol{U}^t$ in $\mathscr{H}_U$



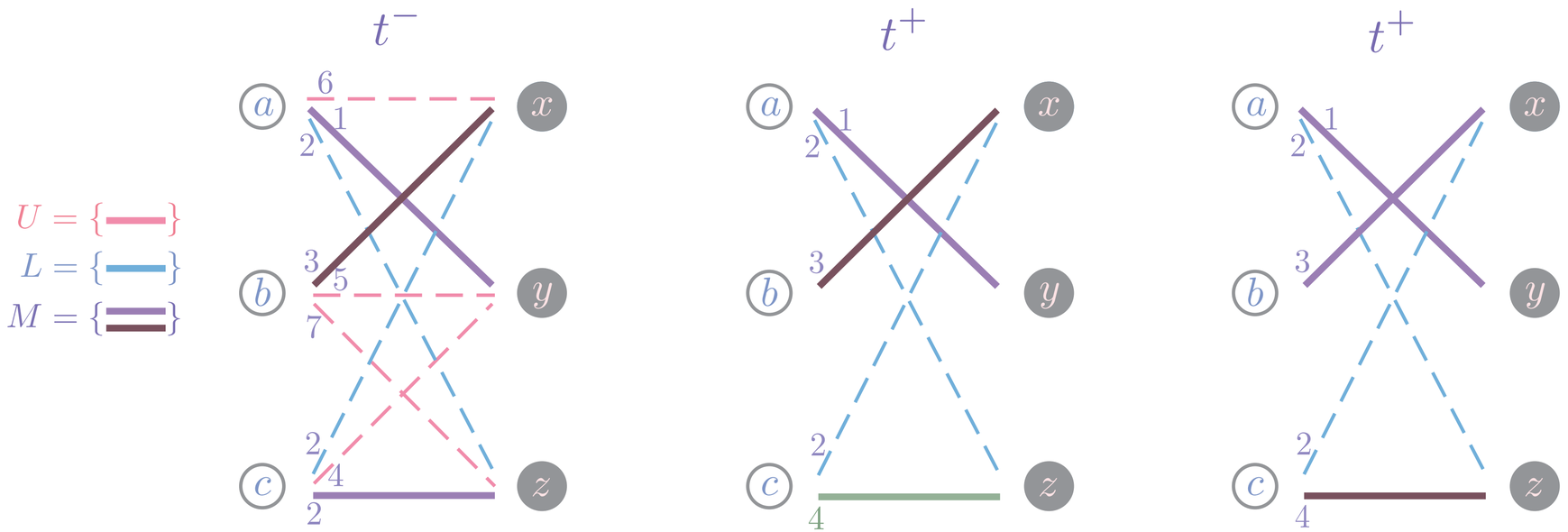
M-Event - Scenario 1

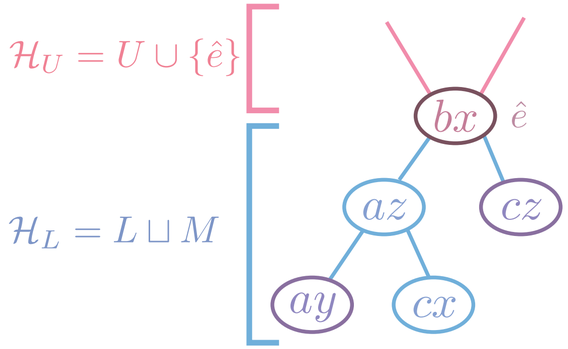
Swap priority of r and an $e \in M^t$ in $\mathscr{H}_L$; i.e., $e \preccurlyeq_{t^-} r$ and $r \preccurlyeq_{t^+} e$

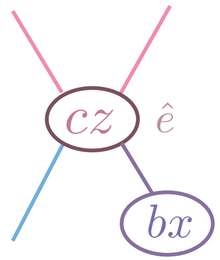


M-Event - Scenario 2

Swap priority of r and an $e \in M^t$ in $\mathscr{H}_L$; i.e., $e \preccurlyeq_{t^-} r$ and $r \preccurlyeq_{t^+} e$





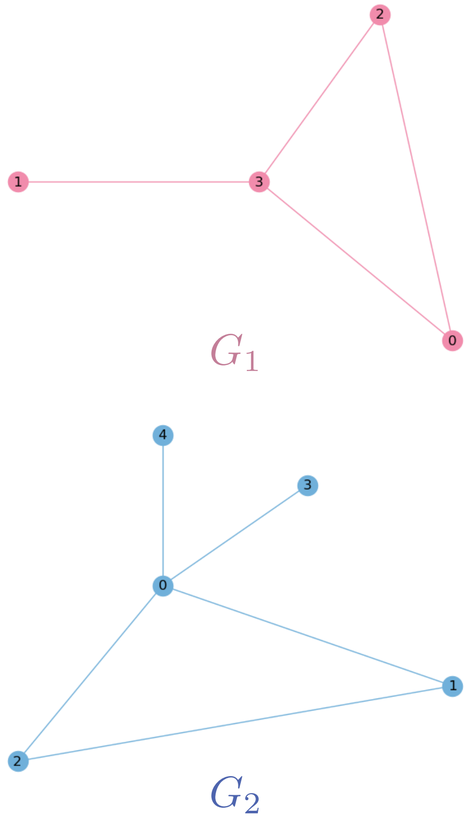


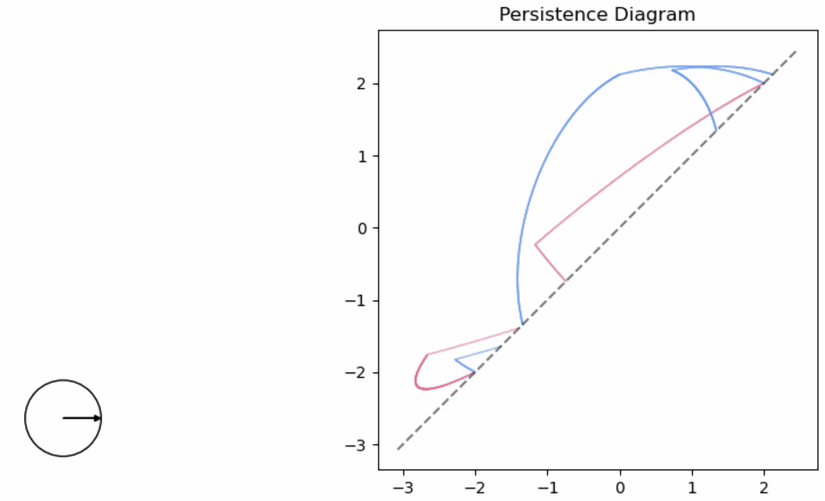
Complexity Analysis

- n is the total number of elements
- $\alpha(\cdot)$ denotes the inverse Ackerman function
- $\lambda_s(n)$ is the Davenport-Schinzle sequence for s-intersecting curves
- External event update (augmenting path search): $O(n \log n)$ [Efrat, Itai, Katz, 2001]

Scenario	Heap Hourglass	Hanger Hourglass
Line segments	$O(n^2\sqrt{n}\log^{1/2} n)$	$O(n^2\alpha(n)\log n)$
s-int. curve segments	$O(n^3 \log n)$	$O(n\lambda_{s+2}(n)\log n)$

Persistent Homology Transform





$$d_1(\mathrm{PHT}(G_1), \mathrm{PHT}(G_2)) = \int_0^{2\pi} d_B\left(\mathrm{Dgm}(h_\omega^{G_1}), \mathrm{Dgm}(h_\omega^{G_2})\right) d\omega$$

The expected time complexity to compute the **integral** bottleneck distance between two PHTs of graphs of n points is $O(n^6 \log n)$ when using the kinetic hanger hourglass

Can we do better?

Computing the Bottleneck Distance between Persistence Homology Transforms



Michael Kerber

International Symposium on Computational Geometry (SoCG), 2026. arXiv: 2512.00821

More Bottleneck Distance for PHT

- Previously: integral version:

$$d_1(\mathrm{PHT}(K), \mathrm{PHT}(K')) = \int_0^{2\pi} d_B\left(\mathrm{Dgm}(h_\omega^K), \mathrm{Dgm}(h_\omega^{K'})\right) d\omega$$

- $\tilde{O}(n^6)$ for $2D$

- This work: max version $d_\infty(\mathrm{PHT}(K), \mathrm{PHT}(K')) = \max_{\omega \in \mathbb{S}^1} d_B\left(\mathrm{Dgm}(h_\omega^K), \mathrm{Dgm}(h_\omega^{K'})\right)$

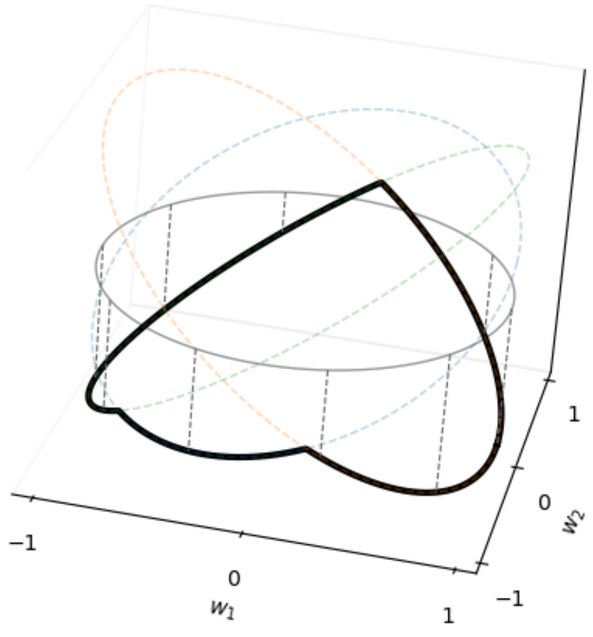
Insertion Curves

- For each simplex $\alpha \in K$ with vertices $\{v_0, \ldots, v_k\} \subseteq V(K)$, we define an *insertion curve*

$$I_\alpha(\omega) := \max_{0 \le i \le k} h_\omega(v_i), \qquad h_\omega(x) = \langle x, \omega \rangle$$

- $I_\alpha(\omega)$ records the height at which each simplex enters the sublevel-set filtration

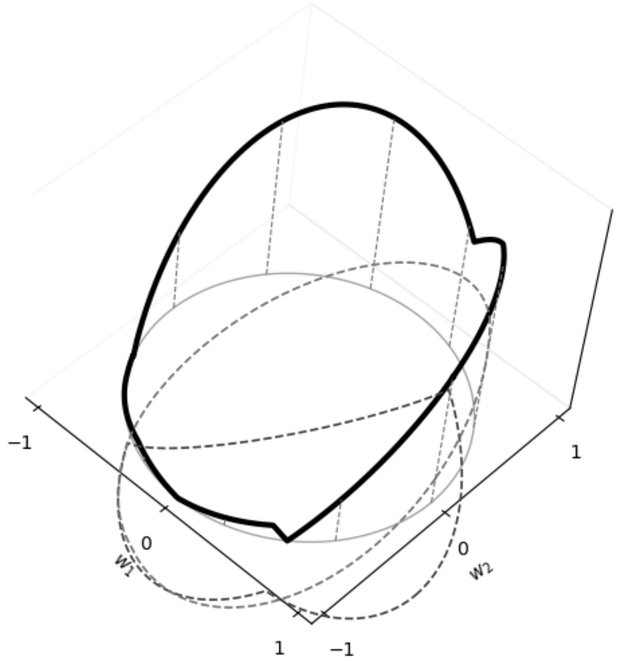
- Given K and K' in $\mathbb{R}^2$ with at most n simplices, there are n insertion curves

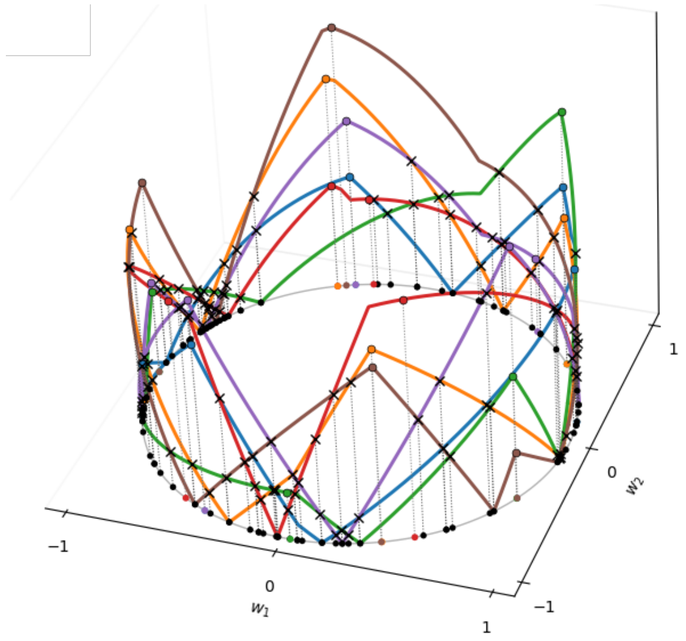


Difference Curves

- For $\alpha \in K$ and $\sigma \in K'$, the *difference curve* is $\Delta_{\alpha,\sigma}(\omega) := \left| I_\alpha(\omega) - I_\sigma(\omega) \right|$

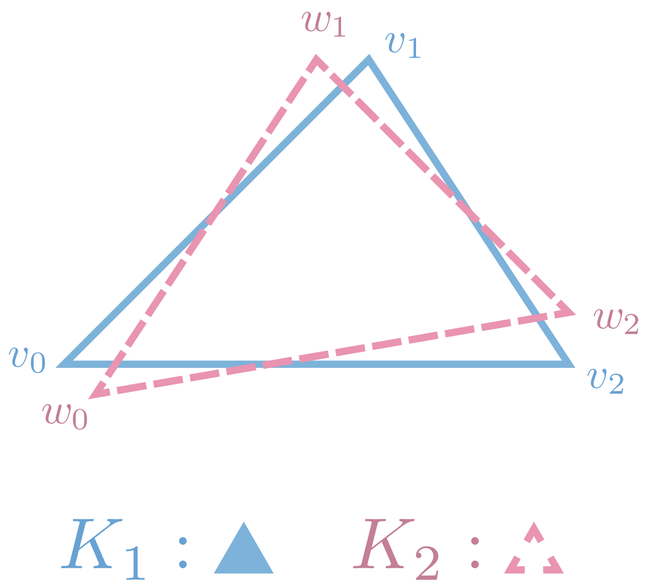
- $O(n^2)$ difference curves

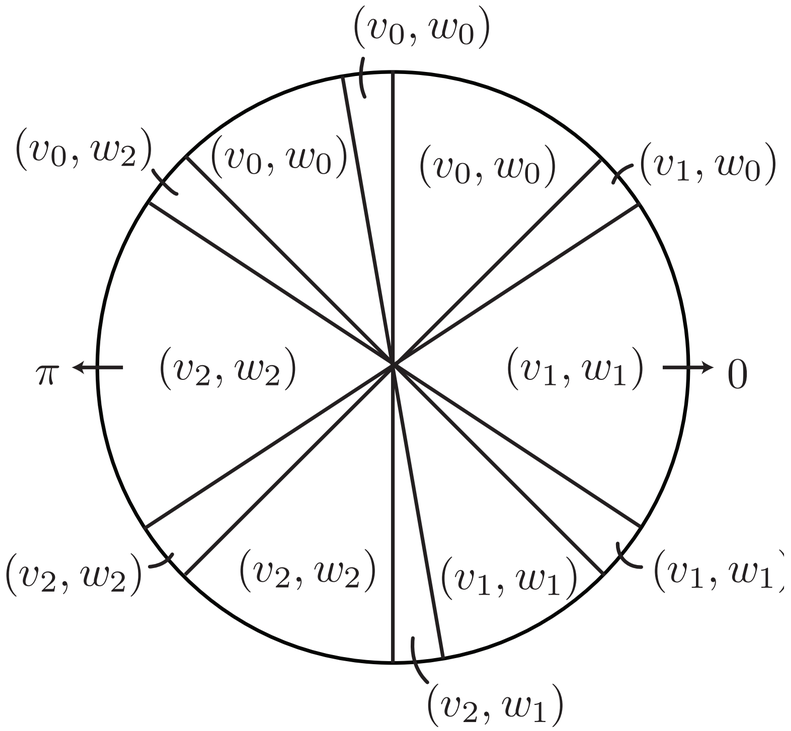




Difference Curve Sectors

Each difference curve $\Delta_{\alpha,\sigma}$ can be broken up into sectors based on the contributing 0-simplex





Events

Theorem: The direction ω^* attaining d_∞ occurs at either

- A local maximum of a difference curve, or
- An intersection of two difference curves

Corollary: There are $O(n^4)$ *candidate values* where d_∞ may be attained

- $O(n^2)$ difference curves $\Delta_{\alpha,\sigma}$ with $O(1)$ local maxima each
- Pairwise intersections of $\Delta_{\alpha,\sigma}$: $O(n^4)$

Improvements in Computing d_1

- Events: $O(n^4)$ difference curve intersections
- Update: $O(n \log n)$ for one augmenting path search [Efrat, Itai, Katz, 2001]
- Total: $O(n \log n) \times O(n^4) = \tilde{O}(n^5)$

Computing d_∞ in $\mathbb{R}^2$ — Overview

Inspiration: Matching distance for 2-parameter persistence [Bjerkevik and Kerber, 2023]

- **Decision procedure**: given λ, is $d_\infty \leq \lambda$?
  - Sweep algorithm: sweeping ω around $\mathbb{S}^1$ and maintaining a bottleneck matching
  - Persistence diagram updates at $O(n^2)$ insertion-curve intersections
  - Augmenting-path search when a difference curve crosses the threshold ($O(1)$ per curve)
  - Total: $O(n \cdot n^2 + n^3 \mathrm{polylog}(n))$
- **Basic Algorithm**
  - $O(n^4)$ candidate values
  - Enumerated and sorted in $O(n^4 \log n)$ time
  - Binary search + decision procedure: $O(n^4 \log n + n^3 \mathrm{polylog}(n)) = \tilde{O}(n^4)$

Computing d_∞ in $\mathbb{R}^2$ — Overview

Band Refinement

- Denote $O(n^4)$ candidates by $\lambda_1, \lambda_2, \ldots$ in increasing order
- Maintain a height band $(\lambda_1, \lambda_2]$ satisfying $d_\infty > \lambda_1$ and $d_\infty \leq \lambda_2$
- α-candidate: a candidate that arises from $\Delta_{\alpha,\sigma}$ for some α. $O(n^3)$ for each α

Predicates

- Existence predicate: Tests if any α-candidate lies in $(\lambda_1, \lambda_2]$ in $O(n^2 \mathrm{polylog}(n))$ time
- Band-refinement predicate: If existence test positive, enumerate all α-candidates in the band and refine via binary search in $O(n^3 \mathrm{polylog}(n))$ time.
- **Total running time:** $O(n \cdot n^2 \mathrm{polylog}(n) + C \cdot n^3 \mathrm{polylog}(n))$, where C is the number of band-refinement calls.
- Random processing: $\mathbb{E}[C] = O(\log n)$, expected time: $\tilde{O}(n^3)$

The rest gets a bit out of hand...

Summary

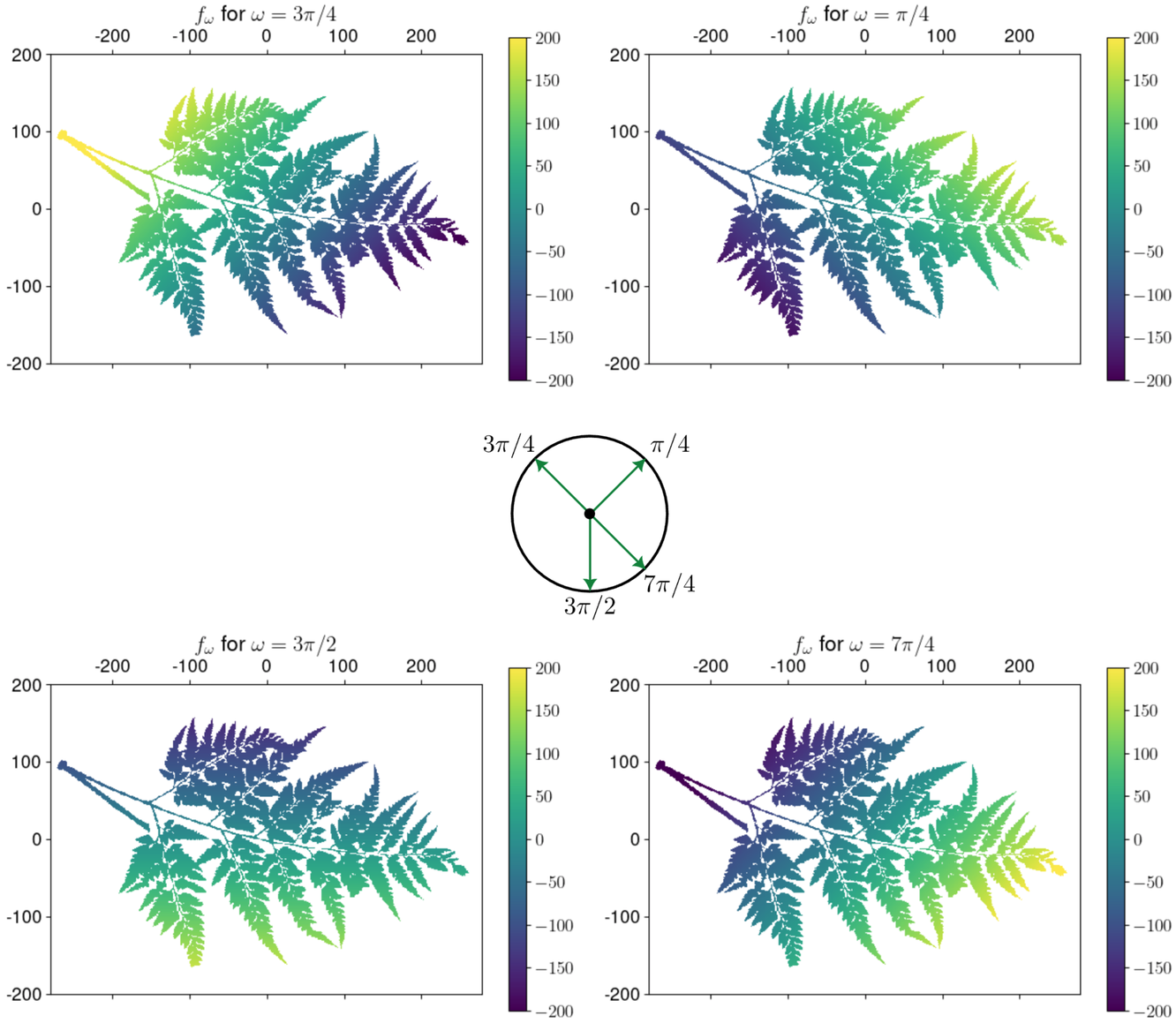
- Adapts the framework of Bjerkevik and Kerber for the matching distance to the PHT setting
- Decision procedure + candidate pruning + randomized incremental construction
- $\tilde{O}(n^3)$ for PHTs in 2D ($\tilde{O}(n^5)$ for PHTs in 3D)
- Bonus: $\tilde{O}(n^5)$ for the PHT integral distance in 2D ($\tilde{O}(n^7)$ for 3D)

What about the ECT?

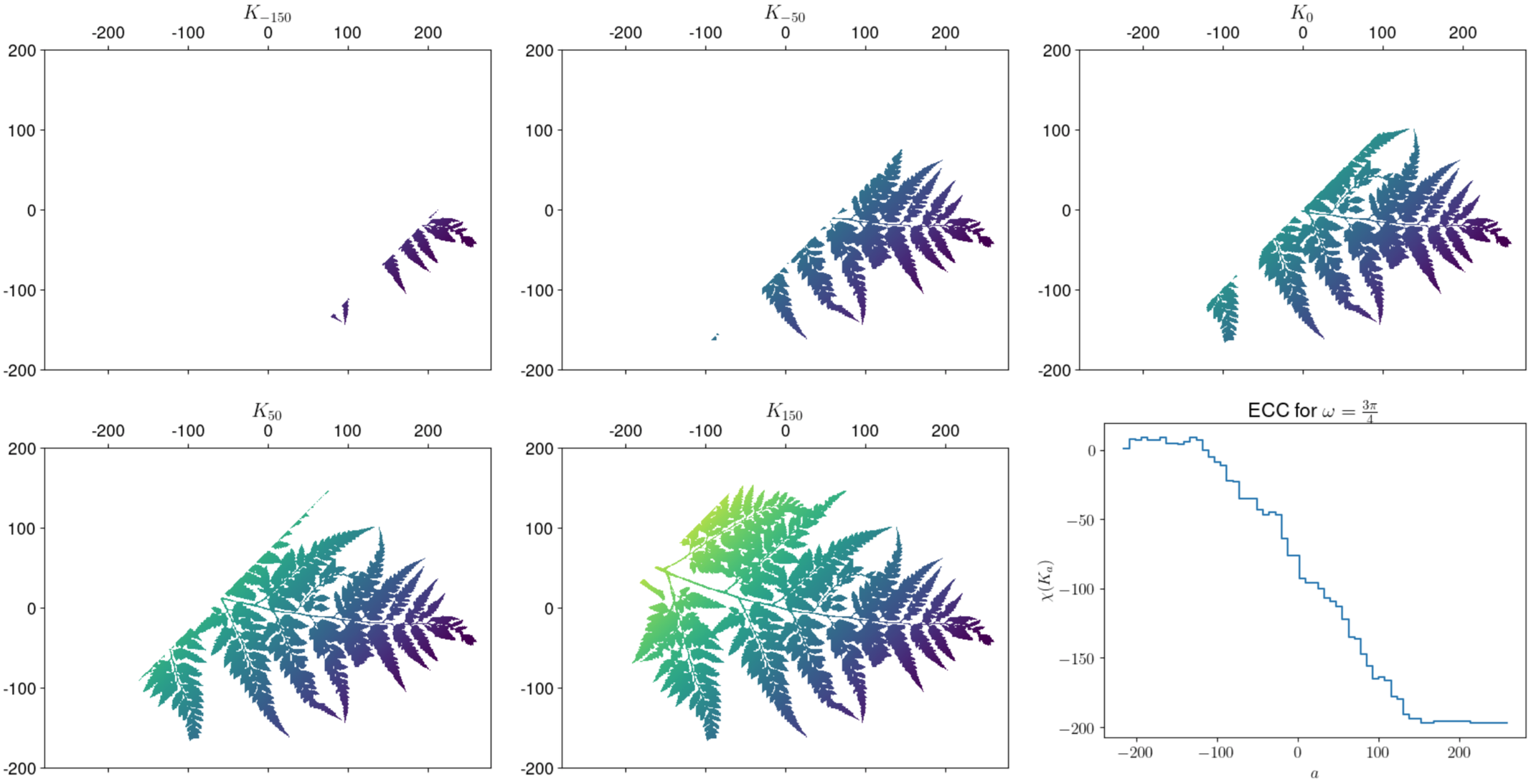
Fern example



Fern - Directional Transform

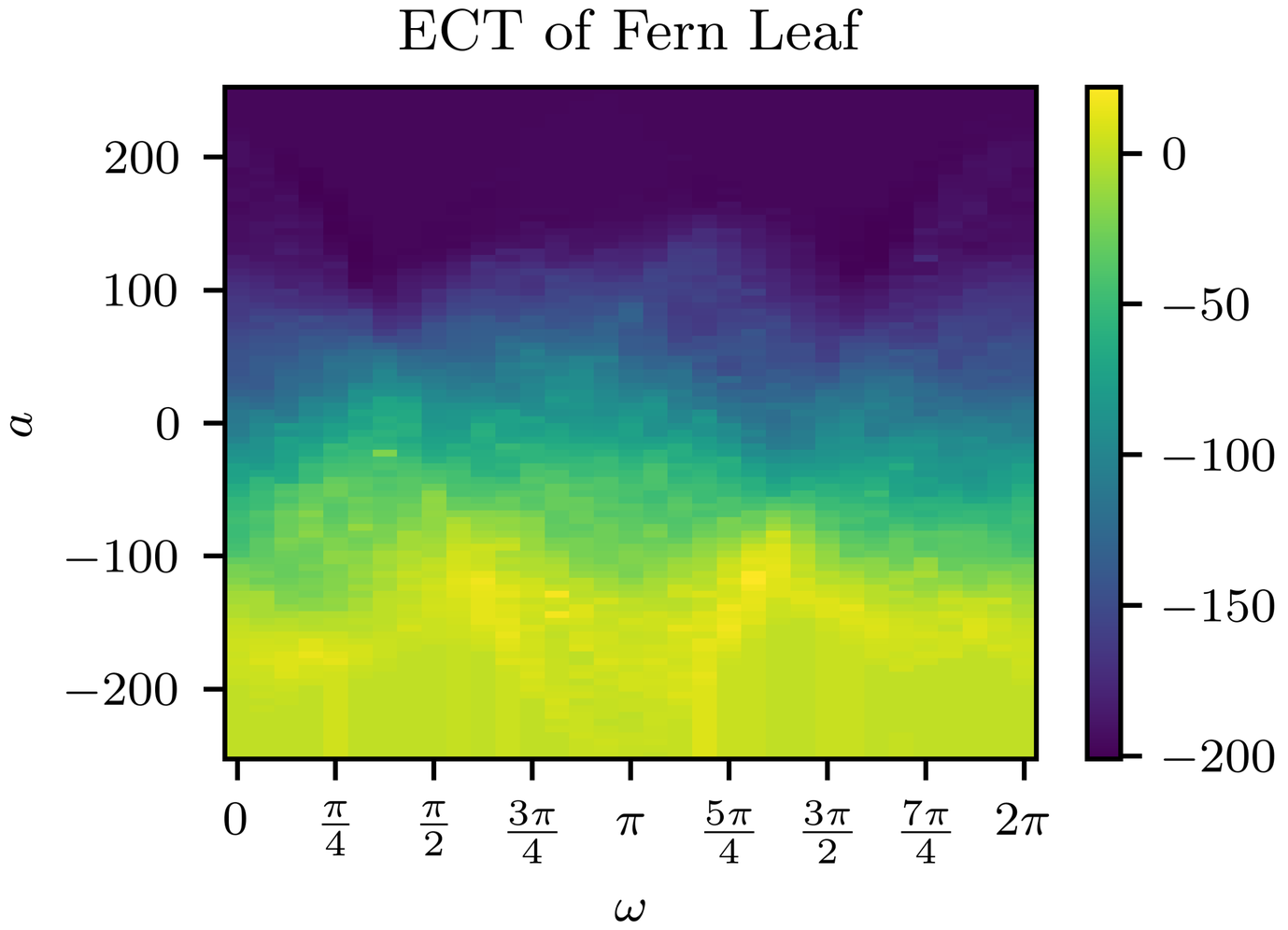


Fern - Thresholds



Fern - ECT




ECT of Fern Leaf
200
100
0
−100
−200
a
0
π/4
π/2
3π/4
π
5π/4
3π/2
7π/4
2π
ω
0
−50
−100
−150
−200

Encoding the ECT

Nello Blaser, Odin Hoff Gardaa, Lars Salbu, EXW, Bastian Rieck

Topology, Algebra, and Geometry in Data Science (TAG-DS) 2026, *Spotlight*

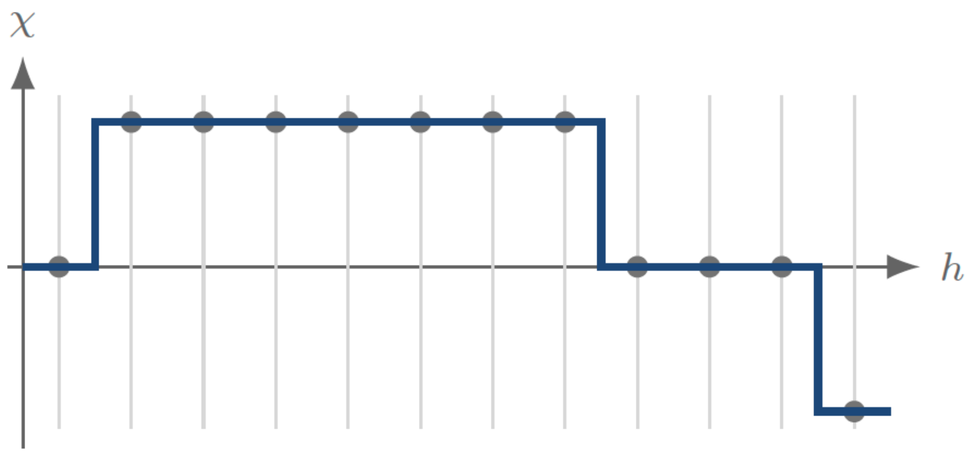
Neural Network Pipeline



- The ECT encodes our shape (data)
- The learner only sees the chosen numerical representation
- Standard choice: evaluate every curve on a fixed height grid
- **Question**: Can we represent each ECC directly, without choosing a grid?

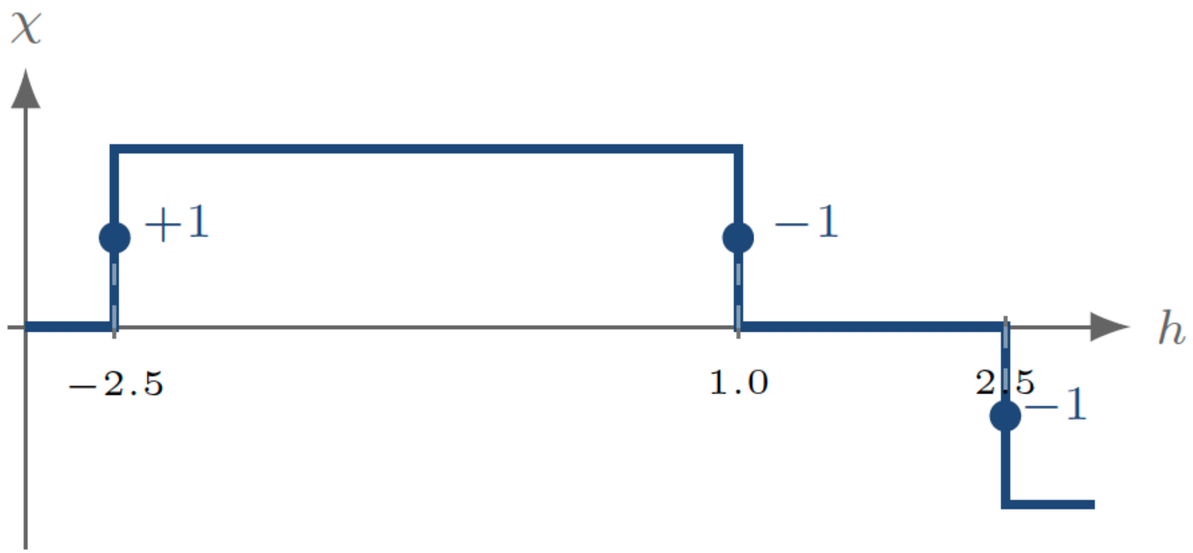
Problem with Usual Encoding

- Resolution becomes a hyperparameter
- Coarse grid: important changes may be missed
- Fine grid: more memory and computations
- “Flat” regions receive the same sampling budget as “active” regions
- Suitable resolution depends on the scale of the dataset



Solution: continuous encoding

Store the events

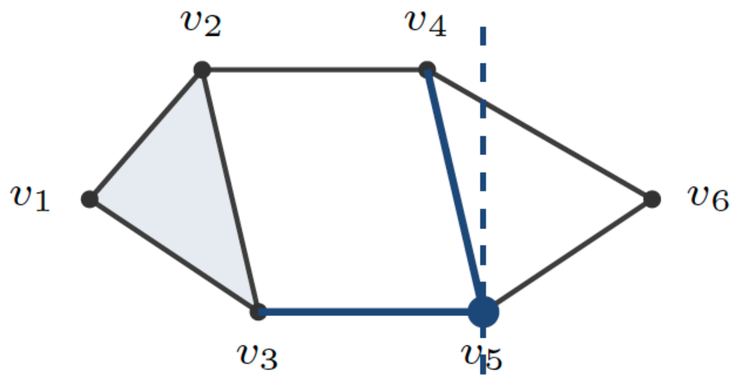


height	$\Delta\chi$
-2.5	$+1$
1.0	-1
2.5	-1

Three jumps → Three tokens

- Token: **where** the curve changes by **how much**
- Exact reconstruction of the staircase
- Good: No height-resolution parameter
- Bad: variable-length sequence of active events

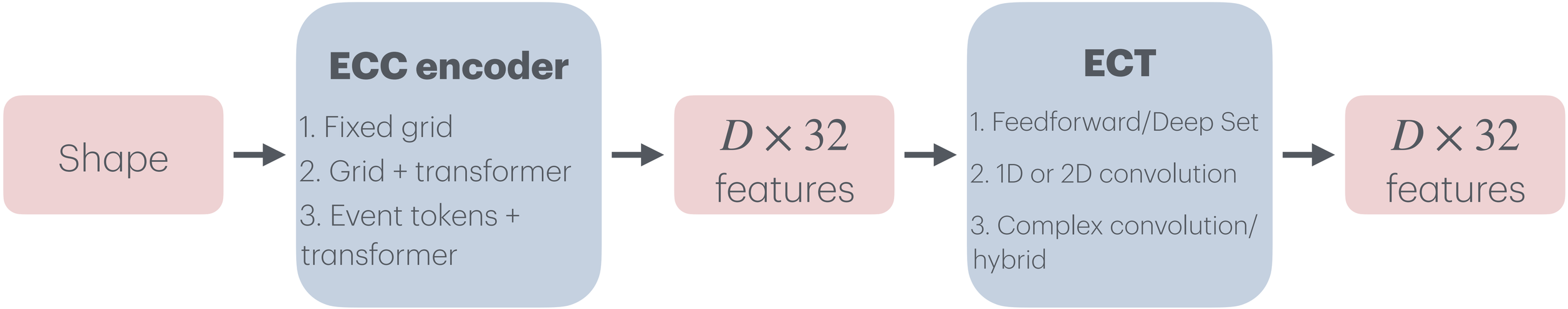
Example



At vertex v_5: $\underbrace{+1}_{\text{vertex}} \quad \underbrace{-2}_{\text{two edges}} = \Delta\chi(v_5) = -1.$

Modeling Choices

New question: how should the directional features be combined?



Architectural Assumptions

Assumption	**Models**	**Structure used**
None	Feedforward	Generic feature vector; no directional structure
Unordered directions	Deep Set	Permutation-invariant aggregation
Circular ordering	1D conv., 2D conv., complex conv., hybrid	Neighboring directions and circular shifts

Benchmarks

Data type	**Dataset**
Point clouds	FashionMNISTPointCloud
Graphs	Letter-low, Letter-med, Letter-high
Cubical complexes	FashionMNISTCubical
Triangle meshes	SwissBuildings

- 64 sampling directions
- 32 features per direction
- Same lightweight classifier

Results

Dataset	**Discrete**	**Continuous**
FashionMNISTPointCloud	0.702	**0.778**
Letter-low	0.936	0.959
Letter-med	0.828	**0.843**
Letter-high	0.751	**0.810**
SwissBuildings	0.717	**0.812**
FashionMNISTCubical	0.778	**0.837**

Control

Dataset	**Continuous tokens**	**Discrete + transformer**
FashionMNISTPointCloud	+7.7 pp	+1.6 pp
Letter-low	+2.3 pp	+3.2 pp
Letter-med	+1.5 pp	−0.8 pp
Letter-high	+5.9 pp	−7.6 pp
SwissBuildings	+9.5 pp	−3.7 pp
FashionMNISTCubical	+5.9 pp	−1.0 pp