Contraction and Hourglass Persistence for Learning on Graphs, Simplices, and Cells

Mattie Ji[1], Indradyumna Roy[2], Vikas Garg[3]

[1]mji13@sas.upenn.edu, [2]indradyumna.roy@aalto.fi, [3]vgarg@csail.mit.edu

Inclusion and Contraction-Based Persistent Homology

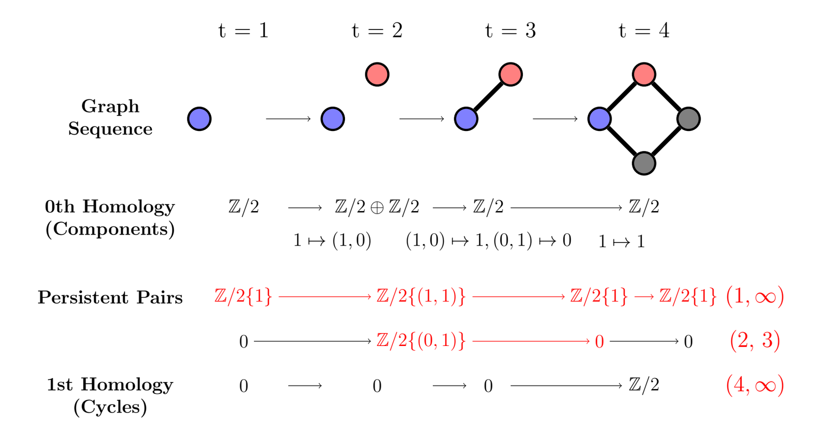
The **persistence homologies (PH)** tracks the **evolution of topological information** over time in a sequence.

$$X_0 \xrightarrow{f_0} X_1 \xrightarrow{f_1} \dots \xrightarrow{f_{n-1}} X_n$$

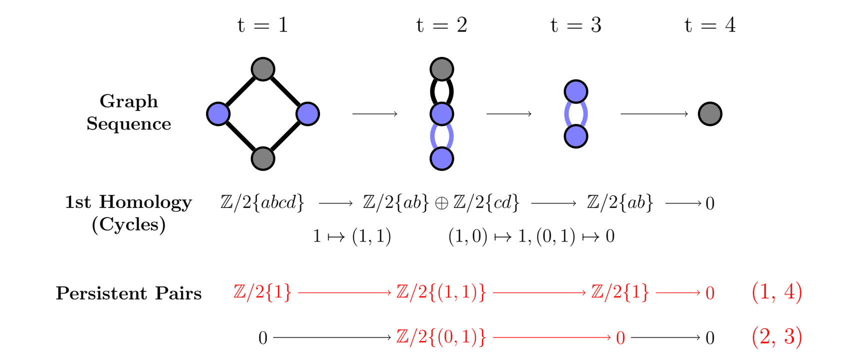
When the X_i's are **graphs**, there are two types of PH signatures: (i) **0th homology which tracks the evolution of components** and (ii) **1st homology which tracks the evolution of cycles**.

Many PH pipelines in graph learning rely on only the case when the maps above are all inclusions. This has limitations in both **expressivity** and **metrizability**. In this work [4], we **study the persistent homologies of contraction sequences** and **how they interact with inclusion sequences**.

Inclusion:



Contraction:



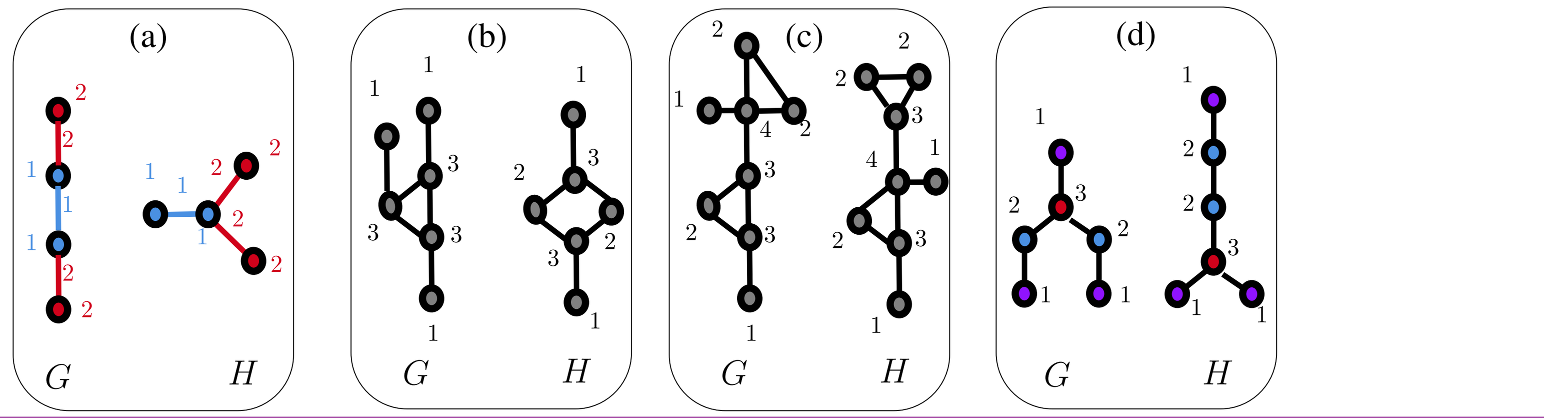
Contraction sequences in general can be quite arbitrary, we first **examine specific descriptors that are produced from a given filtration function f** on X. We consider (* means we saw them in literature before):

1. **Forward PH*** as the PH of the inclusion sequence $\emptyset = X_{-1} \subsetneq \dots \subsetneq X_n = X$ of sublevel sets of f.
2. **Backward PH** as the PH of a sequence of contracting intermediate complexes $\mathrm{IC}_i(X) =$ closure of $X_i - X_{i+1}$ from $\mathrm{IC}_n(X)$ to $\mathrm{IC}_0(X)$.
3. **Forward-Backward (FB) Persistence** as the PH of the sequence of including sublevel sets of f first and contracting in the order of backward PH.
4. **Extended Persistence*** as the PH of the sequence of including sublevel sets of f first and contracting the superlevel sets of f.
5. **Hourglass Persistence** as the PH of **any interleaving** of inclusion steps in f and contraction steps in $\mathrm{IC}_i(X)$, as long as $X_i - X_{i+1}$ is included before contracting $\mathrm{IC}_i(X)$.

Theorem. The following figure illustrates examples where:

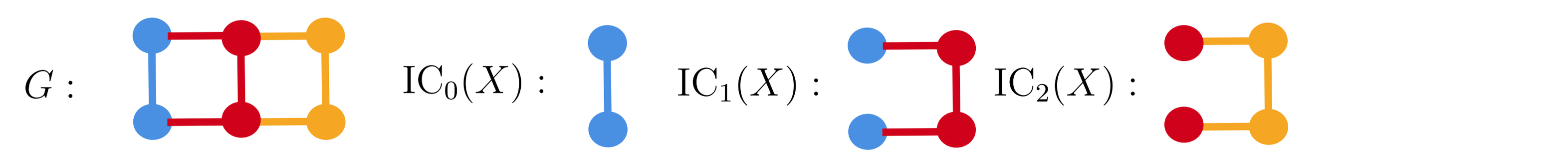
- (a) graph pairs with the **same forward PH** but different **backward PH**.
- (b) graph pairs with the **same backward PH** but different **forward PH**.
- (c) degree filtration with same **forward and backward PH** but different **FB-persistence**.
- (d) degree filtration with same **FB-persistence** but different **hourlgass persistence**.

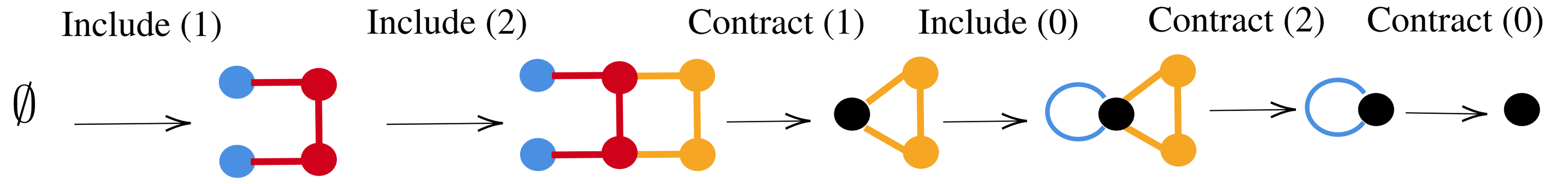
Here (a) is a color-based filtration, and $(b), (c), (d)$ use vertex-based filtrations using the degree. Furthermore, **FB-persistence is strictly more expressive than the union of forward and backward PH**.



Hourglass Persistence: Expressivity and Computations

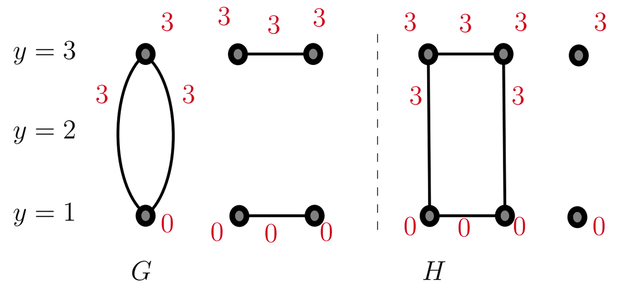
Here is an example of the type of sequence to show up in **hourglass persistence**.





In particular, **FB-persistence** is an example of **hourglass persistence**, and **is not the same as extended persistence**.

Proposition. There are embedded graphs G, H with height filtrations (see Figure on the right) such that **FB-persistence can tell them apart**, but **extended persistence cannot**.



Motivation for Hourglass. Interleaving the filtration and contraction steps avoids the need to filtrate the entire space before starting contractions, which allows the total size of the space that appears in its entire lifetime to be bounded. This can potentially **improve over the runtime on general simplicial complexes**, as the computation for their PH is $O(n^3)$ where n is the number of simplicies. Putting a restriction on the number of simplicies by contraction **scales the computation down**. There is a tradeoff between runtime and expressivity by when and how to contract.

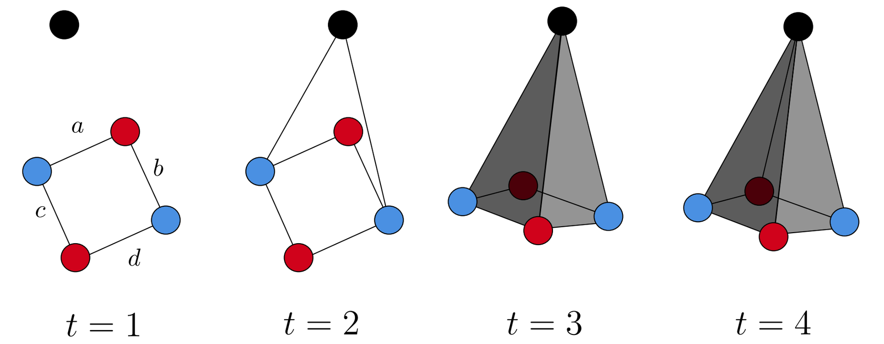
Extension to Simplicial and Cellular Domain

FB-persistence, **Extended Persistence**, and any example of **Hourglass persistence** that filtrates the entire graph before performing contractions, are **all examples of a more general scheme** we know consider.

Definition. Let $f, g : X \to \mathbb{R}$ be two filtrations on X. The (f, g)-FB persistence is the PH of the sequence of including the sublevel sets of f first and contracting the sublevel sets of g.

Our discussions are very generalizable to the setting when X is a **regular cell complex** or a **simplicial complex**.

1. For regular cell complexes, the extension is straightforward as they are closed under quotients considered here.
2. For simplicial complexes, we can adopt a trick of **simplicial quotient** ([2, 3]):



Theorem (Stability of (f, g)-FB persistence)**.** *Let X be a graph, simplicial or cellular complex. For 2 pairs of filtrations $(f, g), (f', g')$ on X, we have the following for all $i \geq 0$. Here the persistence pairs are in **functional time**.*

$$d_B(\mathrm{PH}_i^{FB}(X, f, g), \mathrm{PH}_i^{FB}(X, f', g')) \leq 2||f - f'||_\infty + ||g - g'||_\infty + |\max(f) - \max(f')|.$$

Algorithms

Algorithm 1 FORWARDINCLUSION

```
Input: Filtration f; Graph G
Output: PD_0, PD_1, cycle basis B, union-find UF
Initialize UF on V; PD_0, PD_1, B ← ∅
Sort edges e_1, ..., e_m by f(e_j)
for j = 1..m do
    (u, v) ← e_j
    if UF.find(u) = UF.find(v) then
                                            ▷ Cycle-creating edge
        Build γ ∈ {0,1}^m from e_j and path u ⇝ v
                                            ▷ Update Cycle Basis
        B ← B ∪ {γ}
        PD_1[γ] ← (f(e_j), ∞)
    else                                    ▷ Spanning-tree edge
        r_u ← UF.find(u); r_v ← UF.find(v)
        y ← arg max_{r ∈ {r_u, r_v}} f(r)
        PD_0 ← PD_0 ∪ {(f(y), f(e_j))}
        UF.merge(u, v)
        UF.nbrs ∪ {u ↔ v}                   ▷ Record in spanning forest
for each root r of UF do
    add (f(r), ∞) to PD_0
return PD_0, PD_1, B, UF
```

Algorithm 2 BACKWARDCONTRACTION

```
Input: contraction function g; (PD_0, PD_1, B, UF) from Alg. 1
Output: updated PD_0, PD_1 with finite deaths
Initialize supernode S ← ∅, stack B, list L
for elements x ∈ V ∪ E in order of g(x) do
    if x ∈ V then                           ▷ Node contraction
        S ← S ∪ {x}
        if UF.find(x) ≠ UF.find(S) then
            PD_0 ← PD_0 ∪ {(f(y), g(x))}    ▷ Kill younger
        else
            B.push(g(x))                    ▷ Birth supernode cycle
        UF.merge(x, S)
    else                                    ▷ Edge contraction
        remove e from all γ ∈ B; reduce basis
        if some γ becomes dependent then
            PD_1[γ] ← (birth(γ), g(x))      ▷ Close forward cycle
            B ← B \ {γ}
        else                                ▷ Close supernode cycle
            τ ← B.pop()
            L ← L ∪ {(τ, g(x))}
PD_1 ← PD_1 ∪ L
return PD_0, PD_1
```

Experiments

Graph Expressivity Tests and **Ablations**.

Backbone: GIN					
Dataset	PH	RePHINE	Fwd-only	Bwd-only	Ours
NCI109 (Acc.%, ↑)	76.76 ± 0.40	<u>77.89 ± 1.19</u>	77.00 ± 1.03	76.35 ± 0.50	**77.89 ± 1.87**
PROTEINS (Acc.%, ↑)	69.35 ± 1.83	69.94 ± 2.76	<u>70.24 ± 2.95</u>	70.54 ± 2.19	**73.51 ± 1.11**
IMDB-B (Acc.%, ↑)	68.67 ± 1.25	70.67 ± 0.94	**74.67 ± 0.47**	<u>74.33 ± 0.94</u>	72.00 ± 2.16
NCI1 (Acc.%, ↑)	79.24 ± 1.74	78.75 ± 2.55	<u>76.72 ± 1.13</u>	75.75 ± 0.94	**81.27 ± 0.00**
ZINC (MAE, ↓)	0.43 ± 0.01	<u>0.41 ± 0.01</u>	0.62 ± 0.01	0.61 ± 0.00	**0.40 ± 0.01**
MOLHIV (AUC%, ↑)	**74.34 ± 4.57**	<u>72.88 ± 2.15</u>	70.00 ± 3.11	70.59 ± 1.83	72.34 ± 0.74
Backbone: GCN					
Dataset	PH	RePHINE	Fwd-only	Bwd-only	Ours
NCI109 (Acc.%, ↑)	76.59 ± 1.32	**79.50 ± 0.11**	71.91 ± 0.52	<u>74.58 ± 0.71</u>	75.87 ± 0.89
PROTEINS (Acc.%, ↑)	70.54 ± 0.73	68.75 ± 2.53	69.35 ± 1.83	<u>70.54 ± 1.46</u>	**72.32 ± 1.46**
IMDB-B (Acc.%, ↑)	65.00 ± 1.63	<u>70.00 ± 0.82</u>	62.67 ± 3.30	64.33 ± 3.30	**68.00 ± 2.16**
NCI1 (Acc.%, ↑)	78.43 ± 0.98	**78.91 ± 0.80**	75.59 ± 1.00	76.24 ± 1.74	<u>78.67 ± 1.69</u>
ZINC (MAE, ↓)	0.49 ± 0.02	<u>0.46 ± 0.01</u>	0.86 ± 0.01	0.87 ± 0.01	**0.44 ± 0.01**
MOLHIV (AUC%, ↑)	75.12 ± 0.68	<u>75.40 ± 0.53</u>	71.02 ± 2.18	71.55 ± 1.21	**76.37 ± 1.45**

What You Will Find in the Paper

- More theoretical results on expressivity and algorithms. Comparison to bipath and zigzag filtrations.
- Experiments comparing PersLay [1] and running Algorithm 1 + 2 on $(f, -f)$.

[1] Mathieu Carriere et al. "PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures". In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Vol. 108. PMLR, 2020.

[2] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. "Stability of persistence diagrams". In: *Discrete & Computational Geometry* 37.1 (2006), 103–120.

[3] Tamal Krishna Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022.

[4] Mattie Ji, Indradyumna Roy, and Vikas K Garg. "Contraction and Hourglass Persistence for Learning on Graphs, Simplices, and Cells". In: *The Fourteenth International Conference on Learning Representations*. 2026.