HOURGLASS PERSISTENCE FOR GRAPHS, SIMPLICES, AND CELLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Persistent homology (PH) based schemes help encode information, such as cycles, and are thus increasingly being integrated with graph neural networks (GNNs) and higher order message-passing networks. Many PH based schemes in graph learning employ inclusion-based filtration mechanisms that trace a sequence of subgraphs of increasing size, maintaining bookkeeping information about the evolution (e.g., in terms of birth and death of components). We offer a novel perspective that goes beyond this inclusion paradigm. Specifically, we introduce topological descriptors for graphs, simplices, and cells that interleave a sequence of inclusions with a sequence of contractions and related families parametrized by two functions. The resulting descriptors on the extended sequence are provably more expressive than many existing PH methods with suitable stability conditions. Empirical results substantiate the merits of the proposed approach.

1 INTRODUCTION

Graph Neural Networks (GNNs) (Hamilton, 2020) is a powerful paradigm for learning on structured data, yet their expressive power is fundamentally limited by the Weisfeiler–Lehman (WL) hierarchy (Xu et al., 2019; Morris et al., 2019). In particular, message-passing GNNs struggle to capture higher-order topological signals (ex. the presence, interaction, and disappearance of cycles) that often drive downstream performance in molecular learning, physical systems, and network science (Garg et al., 2020; Chen et al., 2020; Tahmasebi et al., 2021). Persistent Homology (PH) (Edelsbrunner et al., 2002) from Topological Data Analysis (TDA), offers a way to extract such signals by tracking the life of topological features in a filtration. Accordingly, PH-based descriptors are increasingly used to augment GNNs and higher-order neural architectures (Papamarkou et al., 2024), and to boost expressivity Ballester & Rieck (2024); Zhang et al. (2025); Li & Leskovec (2022).

However, many PH pipelines in graph learning rely on *inclusion-based filtrations* ("forward persistence") (Immonen et al., 2023; Ballester & Rieck, 2024; Ying et al., 2024): we monotonically *add* vertices/edges and record when features first appear. This one-sided view leaves information on the table. For instance, in forward time on graphs, *cycles can be born but (in dimension 1) they do not die*; and while *new connected components can appear* as vertices arrive. This is also undesirable in the perspective of *metrizability*, as diagram distances can be ill-defined due to the presence of different number of permanent features. Extended persistence (Cohen-Steiner et al., 2009; Zhao et al., 2020; Carriere et al., 2020; Yan et al., 2022) resolves this by considering both the sublevel and superlevel sets of a given filtration. We instead develop geometric, combinatorial descriptors which achieve the same while capturing information missed by inclusion-only PH.

A filtration can be thought of as a time-forwarding process. Conversely, we can also consider a *time-reversing process* that operate by *contracting* substructures ("backward persistence"). Rather than focusing on superlevel sets, we make the substructures more granular by considering quotients of so called "intermediate complexes". For graphs, we recover the complementary picture to inclusion: *contractions cannot create new components*, but they can *create and kill cycles* as the graph collapses. Forward and backward capture *different* and *incomparable* aspects of the same topology.

Our geometric insight motivates a perspective of persistence that *combines* inclusions and contractions. We first propose *Forward–Backward (FB) persistence*, which *concatenates* a forward (inclusion) phase with a backward (contraction) phase. Intuitively, FB links *how features appear* (forward)

Persistence Goes Forward and Backward (Section 3):	
Construction of Backward PH and Forward-Backward PH	Definition 4, Definition 5
Incomparability for Forward vs. Backward PH	Proposition 1
FB-Persistence $\succ$ Forward PH + Backward PH	Theorem 1
Construction of (σ, τ)-FB PH and Hourglass Persistence	Definition 6, Definition 7
Hourglass Persistence $\succ$ FB-Persistence	Proposition 2
Relation to Extended Persistence and Extensions (Section 4):	
Introduce (f, g)-FB Persistence, with (σ, τ)-FB as Example	Definition 8, Proposition 4
FB-Persistence and Extended Persistence Are Different	Proposition 5
FB-Persistence $\neq$ Extended Persistence	Proposition 5
(f, f)-FB persistence $=$ Forward PH in f	Proposition 6
Extensions to Higher Complexes and Stability (Section 5, 6):	Proposition 7; Theorem 2
Algorithmic Design and Experiments (Section 7)	Alg 1, Alg 2; Table 1

Figure 1: Overview of the paper. Each row summarizes a core result and the section where it appears.

with *how they subsequently vanish* (backward), assigning meaningful, finite lifetimes to structures that would otherwise persist to infinity in forward PH. The resulting descriptors are *strictly more expressive* than forward and backward considered in isolation.

Because of the modular nature of how we set-up the intermediate complexes, we can interchange the sequence of intermediate complexes that are included and quotiented at each step. Guided by this combinatorial insight, we introduce *Hourglass persistence*, which *interleaves* inclusions and contractions in *arbitrary order* (subject only to the constraint that a piece must be included before it can be contracted). Hourglass explores a far richer space of "what-if" topological evolutions, enabling it to discriminate structures that FB—constrained to "all forward then all backward"—cannot.

Finally, we propose a general framework of (f, g)-FB persistence with respect to two filtration functions f and g, which extends both extended and FB-persistence. We then discuss how such constructions can be extended to higher order structures and discuss algorithmic designs. We summarize our content in Figure 1 and include detailed proofs in the Appendix.

2 Background and Preliminaries

Let $G = (V, E)$ be a finite undirected graph, possibly with self-loops and multi-edges. A common theme of enhancing graph neural networks (GNNs) is to incorporate persistent topological descriptors derived from a given filtration of G. For our purpose, a filtration is as follows.

Definition 1. *Let $f : V \cup E \to \mathbb{R}$ be a function. We say f is a **filtration function** on G if for every edge $e = (v, w)$, $f(v) \leq f(e)$ and $f(w) \leq f(e)$. Since G is finite, the list $\{f^{-1}((-\infty, t])\}_{t\in\mathbb{R}}$ has only finitely many elements, which gives a list of sequential subsets we call a **filtration**:*

$$G_{-1} = \emptyset \subset G_0 \subset G_1 \subset \ldots \subset G_n = G.$$

*We say f is a **vertex-based filtration** if for every edge $e = (v, w)$, $f(e) = \max(v, w)$. We say that f is an **edge-based filtration** if $f(v) = 0$ for all $v \in V$ and $f(e) > 0$ for all $e \in E$.*

For consistency, the function f should not change after relabeling the vertices and edges, rather f should only depend on the intrinsic features the graph G comes with. This property is called **permutation equivariance** (Ballester & Rieck, 2024), and we assume all filtration functions to be permutation equivariant. Here we discuss two relevant examples.

Example 1. *Let $\deg : V \cup E \to \mathbb{R}$ be the **degree-based vertex filtration function** such that for any $v \in V$, $\deg(v)$ is the degree of the vertex in G, and for any $e = (v, w) \in E$, $\deg(e) = \max(\deg(v), \deg(w))$. This is permutation equivariant as it only depends on the structure of G.*

Example 2. *Suppose G is a **colored graph** (ie. vertices have labels) equipped with an additional structure of a coloring function $c : V \to C$, where C is a collection of colors. A **vertex-color***

filtration *is a vertex-based filtration $f : V \cup E \to \mathbb{R}$ such that $f(v) = f(w)$ for all v, w with $c(v) = c(w)$. Intuitively, f preserves the vertex coloring.*

Example 1 can be viewed as a special case of vertex-color filtration in Example 2 where c is the degree function and $f = c$. A common scenario where colored graphs arise is from molecular graphs (Hoogeboom et al., 2022; Xu et al., 2022; 2023; Song et al., 2024), where multiple instances of different atom types (e.g., oxygen, carbon) are labeled on nodes within the same molecule.

A common descriptor to extract data from a given filtration is **persistent homology** (PH). For the ease of our discussions in Section 3, we will state a more general definition.

Definition 2. *Let $X_\bullet = (X_0 \xrightarrow{f_1} X_1 \dots X_{n-1} \xrightarrow{f_{n-1}} X_n)$ be a sequence of topological spaces (ex. **graphs**) X_i with maps $f_i : X_i \to X_{i+1}$. The **k-th persistent homology** of $X_\bullet$ is the result of applying $H_k(-)$ (in $\mathbb{Z}/2$-coefficients) to $X_\bullet$, that is, it is the sequence of linear maps:*

$$H_k(X_\bullet) = H_k(X_0) \xrightarrow{(f_0)_*} H_k(X_1) \xrightarrow{(f_1)_*} \dots H_k(X_{n-1}) \xrightarrow{(f_{n-1})_*} H_k(X_n).$$

*An element $v \in H_k(X_i)$ is said to **born** at i if $v \notin \operatorname{im}((f_{i-1})_*)$. The same element $v \in H_k(X_i)$ **dies** at $d \geq i$ if d is the first element $\geq i$ such that $(f_{d-1})_* \circ \dots \circ (f_i)_*(v) = 0$. The **persistence pair** associated to v is the pair (b, d). If no such d exists, we mark the pair as (b, ∞). If the incoming complex is of the form $X_\bullet = (X_{-1} = \emptyset \xrightarrow{f_{-1}} X_0 \to \dots \to X_n)$ (e.g. from a filtration), by $H_k(X_\bullet)$ we mean $H_k(-)$ of the sequence with the X_{-1}-term truncated.*

*Suppose the vector spaces are all finite dimensional, the k-th **persistence diagram** (ie. barcodes in Ghrist (2007)) of $X_\bullet$ is the collection of persistent pairs for a specific choice of basis of the $H_k(X_i)$'s obtained from an interval decomposition of $H_k(X_\bullet)$ (see Theorem 2.7-8 of Chazal et al. (2016) or Theorem 4.7 of Lesnick (2025) for more details).*

Appendix B.1 shows its equivalence to the usual definition of graph PH. Here $H_0(-)$ are connected components, $H_1(-)$ are independent cycles, $H_2(-)$ are voids, and $H_k(-)$ are k-dimensional voids.

Remark 1. *Explicitly, the k-th persistence diagram of $X_\bullet$ can be computed by first picking a non-zero vector v of $H_k(X_0)$, and consider the sequence of linear subspace generated by the iterated image of v in $H_k(X_\bullet)$ until it becomes 0. Then we remove this sequence of linear subspaces off of $H_k(X_\bullet)$. If there is still a non-zero vector w in $H_k(X_0)$, we pick w and repeat the process (if the linear map sends w to the complement, we consider it becomes 0). Otherwise, we choose a non-zero vector from what is left in $H_k(X_1)$ (if it exists) and look at its iterated images ahead, and so on.*

3 Persistence Goes Forward and Backward

For now, we will focus on the case of graphs, but many constructions here can be readily generalized to the setting of simplicial and cell complexes (see Section 5). Classically, there are numerous applications in TDA and GNNs that uses an inclusion based persistent homology (Edelsbrunner & Harer, 2008; Edelsbrunner & Morozov, 2012; Immonen et al., 2023; Ballester & Rieck, 2024). Concretely, an inclusion based PH may be viewed as an example of Definition 2 as follows.

Example 3 (Inclusion-based/Forward PH)**.** *Let $f : G \to \mathbb{R}$ be a filtration function, and consider applying $H_0(-)$ and $H_1(-)$ to $G_\bullet : G_{-1} = \emptyset \subset G_0 \subset \dots \subset G_n = G$, where each map is the inclusion map. Then the **persistent diagram** in Definition 2 recovers all birth/death pairs (b, d) such that $b < d$ (i.e., it omits trivial births). See Appendix B.1 for a formal proof. We can recover the trivial births by counting the number of unmarked simplices at each time.*

The interest that initiated this work is - what if we try to reverse the steps of the inclusion using **contractions**? More precisely, we would like to contract the following sub-graphs.

Definition 3. *Let $H \subset G$ be a subset, the closure of H is the union of H and all vertices with incident edges contained in H. Let $\emptyset = G_{-1} \subset G_0 \subset \dots \subset G_n = G$ be a filtration of G induced by f, we define the **intermediate complexes** $\mathrm{IC}_i(G, f)$ as the closure of $G_i - G_{i-1}$ for $i = 0, \dots, n$. We omit the symbol f when the context is clear.*

Definition 4 (Backward-based PH)**.** *Let $G_\bullet : G_{-1} = \emptyset \subset G_0 \subset \dots \subset G_n = G$ be a filtration of G, we consider a **sequence of contractions** with respect to $G_\bullet$ as $(G_\bullet)^v$*

$$G \to G_{n+1} := G/\operatorname{IC}_n(G) \to G_{n+2} := G/(*_{n+1} \cup \operatorname{IC}_{n-1}(G)) \to G/(*_n \cup \operatorname{IC}_{n-2}(G)) \to \dots \to *,$$

where $*_{n+1}$ is the point representing the total contracted subcomplex in the previous step, and so on. The intermediate maps are the natural quotient maps. From here, we define the ***i-th backward-based PH*** *of $G_\bullet$ as $H_i((G_\bullet)^v)$, and the* ***i-th backward persistent diagram*** *as the persistent diagram associated with $H_i((G_\bullet^v))$.*

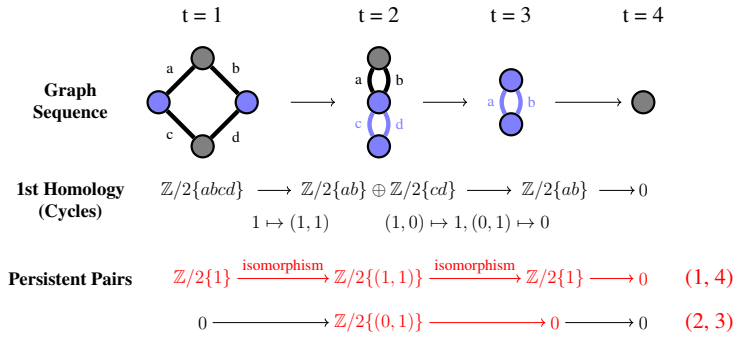


Figure 2: Backward PH of a graph G. The blue simplices indicate what is being contracted.

We observe that the backward-based PH introduces new information that inclusion-based PH may ignore. A high-level reason for this distinction is that a cycle that is born in forward-based PH can never die, but backward-based PH can both create new cycles (apart from the initial time) and kill them. Conversely, backward-based PH can never make a new connected component apart from the initial step, but inclusion-based PH can spawn new components later.

Proposition 1. *There exists graphs G, H with permutation equivariant filtrations such that the forward-based PH of their filtrations cannot tell apart G, H but backward-based PH can. Similarly, there are graphs that backward-based PH cannot tell apart but forward-based PH can. Examples can be found in Figure 3.*

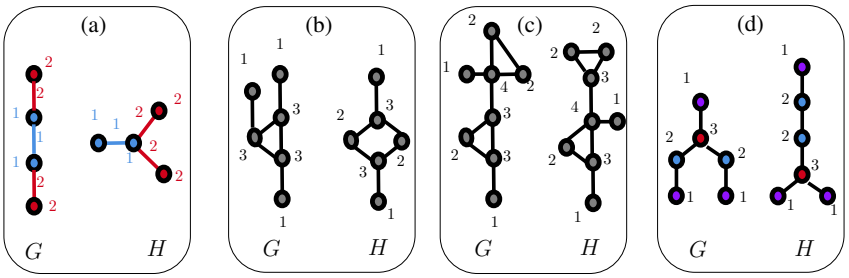


Figure 3: Example graph pairs where (a) color filtration with same **forward-based PH** but different **backward-based PH**, (b) degree filtration with same **backward-based PH** but different **forward-based PH**, (c) degree filtration with same **forward and backward PH** but different **FB-persistence**, (d) degree filtration with same **FB-persistence** but different **hourlgass persistence**.

Another motivation for us to introduce backward-based PH is from the viewpoint of **metrizability** and **stability**. Classically, the **bottleneck distance** Cohen-Steiner et al. (2006) between 2 persistence diagrams $\mathrm{PD}_1, \mathrm{PD}_2$ of the same size is the infimum of the value $\max_{p \in \mathrm{PD}_1} ||p - \pi(p)||_\infty$ where π ranges over all bijections $\pi : \mathrm{PD}_1 \to \mathrm{PD}_2$. The bottleneck distance is finite for persistence diagrams coming from the same graph, but for diagrams, coming from two different graphs, this distance may very well be infinite. The reason why is because the number of tuples of the form $(-, \infty)$ for both graphs may be different, which adds an ∞ to the distance function.

One might argue that there can be workarounds by setting the time to die at a finite time N after both filtrations ended, or make it die at -1. These modifications are however still not that desirable because they can be quite sensitively altered by, for example, having many features die around N or -1. One solution would be to extend the (inclusion-based) persistent diagrams by killing off the permanent features topologically via a sequence of contractions. Formally, this motivates our definition of **Forward-Backward (FB) persistent homology**.

Definition 5 (Forward-Backward Persistence). *Let $G_\bullet = (\emptyset = G_{-1} \subset G_0 \subset ... \subset G_n = G)$ be a filtration of G. The i-**th forward-backward persistent homology** is the diagram $H_i(G_\bullet + (G_\bullet)^v)$, where $G_\bullet^v$ denotes the sequence of contractions in Definition 4 and $+$ denotes the concatenation of the two sequences.*

Concretely, the forward-backward persistence diagrams of a filtration $G_\bullet$ can be obtained in the following way. First compute the inclusion-based PH on $G_\bullet$; however, at the end of the filtration, instead of marking the remaining simplices to die at ∞, we instead mark them with their death times in the backward PH (shifted up by $+n$). We also include the additional pairs that arise from the backward PH. We now establish the expressivity benefits due to FB-persistence.

Theorem 1. *FB-persistence is **strictly more expressive** than forward and backward persistence combined (see Figure 3(c) for an example). More precisely, for any graph G with filtration f, the FB-persistent diagram with respect to f can recover the correspondent forward and backward persistence diagrams. However, there exists a pair of graphs G, H with a permutation equivariant filtration f such that their forward and backward persistence diagrams are the same, but their FB-persistence diagrams differ.*

A high level reason why Theorem 1 holds is that deciding how we concatenate the values from the forward persistent diagrams with the values from the backward persistent diagrams is rather subtle.

Motivated by the perspective of **color-based filtrations** (Ballester & Rieck, 2024; Immonen et al., 2023; Ji et al., 2025), we observe that there is no canonical reason to contract in the order of $\mathrm{IC}_n(G), \mathrm{IC}_{n-1}(G), ...$ in Definition 4 and Definition 5. Since the intermediate complexes satisfy (i) $\bigcup_{i=0}^n \mathrm{IC}_i(G) = G$ and (ii) $\mathrm{IC}_i(G)$ and $\mathrm{IC}_j(G)$ can only intersect at vertices, any sequence of contracting the subgraphs in the list $\mathrm{IC}_0(G), \mathrm{IC}_1(G), ..., \mathrm{IC}_n(G)$ would have terminated at the single point set $*$. Likewise, we can also see the filtration step as a special case of spawning the pieces $\mathrm{IC}_0(X), \mathrm{IC}_i(X), ...$ in ascending order. There is no reason why we should have included them in this order. This motivates the following construction.

Definition 6 ((σ, τ)-FB persistence). *Let σ, τ be two permutations of the list $[n] = \{0, ..., n\}$. The i-th $(\sigma, \tau)-$**FB persistence** is the persistence diagram associated the persistence module obtained by applying $H_i(-)$ to the sequence*

$$\emptyset = Y_{-1} \subset ... \subset Y_n = G = Z_0 \to Z_1 \to ... \to Z_n = *,$$

where for $0 \leq i \leq n$, $Y_i = \bigcup_{j \leq i} \mathrm{IC}_{\sigma(j)}(G)$, and for $1 \leq j \leq n$, $Z_j = Z_{j-1}/(*_{j-1} \cup \mathrm{IC}_{\tau(j)}(G))$ where $*_{j-1}$ is previously defined for $j - 1 \geq 1$, and $*_0$ is any point in $\mathrm{IC}_{\tau(1)}(G)$.

In other words, we apply Definition 2 to the sequence where we filtrate G by spawning the intermediate complexes in the order σ, and then contracting G in the order τ. In the case where id is the identity and re is the reverse list bijection. FB-persistence is $(\mathrm{id}, \mathrm{re})$-FB-persistence.

A core motivation for Definition 6 comes from color-based filtrations (Example 2). Indeed, recall that Example 1 is a special case of Example 2 when the vertices are spawned in ascending order of their degrees. The perspective of coloring filtration is that we can spawn the vertices in any permutation of the set of possible degree values here, which may lead to more information. On the other hand, to faithfully adopt the "coloring-based philosophy to this context", one could also argue why should we wait until the entire graph has been filtrated to start the contraction process?

Indeed, the whole process would still terminate to a point as long as we ensure the intermediate complex being contracted has appeared earlier in time, and this yields further flexibility in the process of switching back and forth between the forward steps and the backward steps (analogous to the imagery of an **hourglass**). This motivates us to the definition of **hourglass persistence**, which may be regarded as a "color-based version" of FB-persistence that also has a well-defined metric for persistence diagrams on different graphs.

Definition 7. *Let $f : X \to \mathbb{R}$ be a filtration function with associated intermediate complexes $\mathrm{IC}(X_i)$. An **hourglass persistence diagram** is the persistence diagram of any sequence of inclusions and contractions, provided that $\mathrm{IC}(X_i)$ is included in the sequence before it is being contracted.*

Figure 4 gives an illustration of an example of sequence of maps that occurs in hourglass persistence. To demonstrate its expressivity, we have that:

Proposition 2. *Hourglass persistence is more expressive than FB-persistence (Figure 2(c)).*

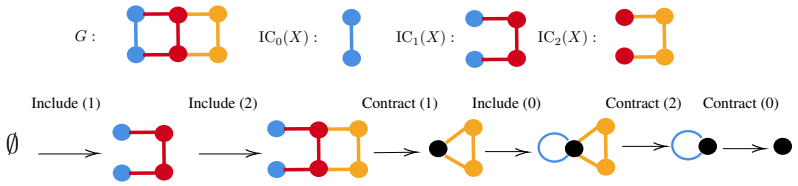


Figure 4: Example of a sequence of maps arising in the hourglass persistence for a filtration of G.

Hourglass persistence has promising potential to tackle a problem that PH-based methods sometimes struggle to scale on large graphs/simplicial complexes/cell complexes, see Appendix E.4.

4 Relationship to Extended Persistence and Extensions

Extended and FB-persistence are different as follows: For a filtration function f, extended persistence is obtained by considering a concatenation of PDs with filtrating by f first and by $-f$ back. By Proposition 2.22 of Hatcher (2002), this is essentially akin to shrinking the superlevel sets $G^a := \{x \in V \cup E \mid f(x) \geq a\}$ as a goes down, but the superlevel sets and the intermediate complexes coming from f are in general quite different. We make this precise in Proposition 5.

We will now introduce a unifying perspective of both methods:

Definition 8 ((f,g)-FB Persistence)**.** *Let f, g be two filtration functions on G with $G^f_\bullet, G^g_\bullet$ being their induced filtrations respectively. The i-**th** (f,g)-**FB Persistence** of G is the persistence diagram associated with the sequence:*

$$\emptyset = G^f_{-1} \subset G^f_0 \subset \ldots \subset G^f_n = G = G^g_0 \to G^g_1 \to \ldots \to G^g_m = *.$$

where $\{G^f_i\}_{i\in\{-1,\ldots,n\}}$ is the filtration of G induced by f, and the sequence $G^g_\bullet := G^g_0 \to G^g_1 \to \ldots \to G^g_m = *$ is a sequence of contractions following the subgraphs that appear in the filtration induced by g. More precisely, $G^g_1 := G/\,\mathrm{IC}_0(G,g)$, $G^g_2 := G/(*_1 \cup \mathrm{IC}_1(G,g))$, $G^g_3 := G/(*_2 \cup \mathrm{IC}_2(G,g))$, etc., where $*_i$ is the point representing the total contracted subcomplex from before.

The idea of combining different filtration directions have appeared in zigzag filtration (Carlsson & de Silva, 2010), bipath filtration (Aoki et al., 2025), and extended persistence (for specific pairs). Our considerations here is different from zigzag and bipath filtrations as we are doing PH on the uni-directional quiver on the path graph P_n but with maps coming from both inclusions and contractions, whereas zigzag and bipath filtrations are usually used on different quivers with only inclusions. Our setup is more general than extended persistence as we can interpret extended persistence in the framework of (f,g)-FB Persistence. Note that the following result is not diffcult to prove; we only include it here for completeness.

Proposition 3. *The extended persistence of a filtration function f (as defined in Section 2 of Yan et al. (2022)) has the same expressive power as $(f,-f)$-FB persistence.*

Note that the expressivity analysis of extended persistence had been carried out in Yan et al. (2025).

Proposition 4. *Let f be a filtration function. There exist filtration functions f_1, f_2 such that the sequence of topological maps in (f_1,f_2)-FB persistence is exactly the sequence of topological maps in (σ,τ)-FB persistence. If f is vertex-based, we provide an explicit $O(n\log n)$ algorithm to compute (f_1,f_2) that becomes linear time for FB-persistence. (see Appendix B.3).*

Now we establish a clear separation between FB persistence and extended persistence.

Proposition 5. *There exists graphs G, H with permutation equivariant filtrations such that FB persistence can tell them apart but extended persistence cannot, see Figure 5.*

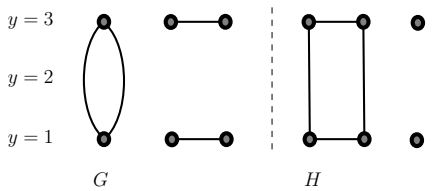


Figure 5: Example of graphs G and H with **height filtration** such that FB-persistence can differ but not $(f,-f)$-FB persistence.

Besides the choice of $-f$ or f^b, another candidate can be f itself. Here we show that (f, f)-hourglass persistence does not bestow further information.

Proposition 6. *(f, f)-FB persistence has the same expressive power as forward-PH for f. For graphs, (f, f)-FB persistence can be computed by:*

- *Suppose there are n-steps in the filtration, a cycle born in filtration at $t = i$ dies at $t = n + i$.*
- *A cycle born in the contraction at $t = n + i$ corresponds exactly to the occurrence of a connected component at $t = i$ that will be merged into a pre-existing connected component in the filtration steps. The 1st time $j > i$ in filtration when the cycle does the merge above gives the death time $n + j$ in contraction.*
- *The component death times are recorded as usual by keeping track of vertex representatives.*

5 Extension to Simplicial and Cellular Complexes

We now extend the framework to higher dimensions, viewing graphs as 1-dimensional simplicial complexes (for simple graphs) and 1-dimensional cell complexes.

5.1 Extension to Simplicial Complexes

Definition 2 works for any sequence of topological spaces, so our constructions may be defined directly. The issue with simplicial complexes is, that they are not closed under quotients, so the spaces showing up in the contraction stage may not be simplicial.

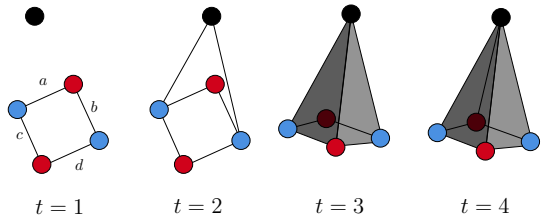


Figure 6: The contraction steps in Figure 2 interpreted as adding simplices instead.

We can however adapt a trick to conduct "simplicial quotients" (see Cohen-Steiner et al. (2006); Dey & Wang (2022)) as follows. Add a disjoint vertex v_+ to the simplicial complex at the beginning, and whenever a simplex σ is asked to be contracted, one instead adds a simplex $[v_+, \sigma]$ to keep this as a filtration. On graphs, this means that one adds an edge from v to v_+ for every vertex v being contracted and a triangle (v_1, v_2, v_+) for every edge (v_1, v_2) being contracted (see Figure 6). One then computes the persistent diagram of this filtration instead.

Proposition 7. *For all methods defined in Section 3, 4, the simplicial quotient method will recover the k-th dimensional persistence diagrams for $k > 0$ but may differ on the level of $k = 0$. For $k = 0$, the pairs can be directly computed by keeping track of one vertex representative per component.*

5.2 Extension to Cellular Complexes

The simplicial version adds more simplices to the set-up, which may make the computations more costly. We should try to exploit the contraction steps geometrically as they would reduce the data. To do this, we would like to relax the collection of objects we are working with.

A **(regular) cell complex** (Hansen & Ghrist, 2019) is a generalization of simplicial complexes that adds more flexibility by allowing X be built off of k-dimensional disks (called k-cells) as opposed to k-dimensional triangles. Intuitively, a cell complex is built out inductively by starting with the 0-cells (ie. discrete points), attaching 1-cells to 0-cells, then attaching 2-cells to the complex, and so on. This process yields a poset structure on the cells where $\tau \leq \sigma$ indicates part of σ is attached onto τ. We refer to Section 2 of Bodnar et al. (2021) for a thorough explanation.

Now we may extend a filtration function on a cell complex X as follows.

Definition 9. *Let $\mathrm{Cell}(X)$ denote the collection of cells on X. A function $f : \mathrm{Cell}(X) \to \mathbb{R}$ is a filtration function if $f(\tau) \leq f(\sigma)$ for all $\tau \leq \sigma$ in $\mathrm{Cell}(X)$, which induces a filtration on X.*

An important feature of cell complexes is that they are closed under quotients of subcomplexes. Thus, every construction we discussed in Section 3 and 4 extends *mutatis mutandis* - that is, one can just replace the word "graph" with the word "cell complex".

6 STABILITY

Recall one motivation for "concatenating" inclusions and contractions was to compare metrics for diagrams on different spaces. We would still like to ensure the diagrams are stable if they are on the same space. This is a desired property as we would like perturbations in the input filtration to not affect the output drastically. To properly discuss (bottleneck) stability, however, we need to make a distinction between **combinatorial time** and **function time**.

So far, the persistence diagrams in our construction have all been combinatorial time - we use some function(s) to make a sequence of maps, and the pairs (i, j) record the birth and death steps in the sequence. However, if we want a notion of bottleneck stability, we need to work with a notion of function time - that is we change the pair (i, j) to the (a_i, a_j). Take forward PH for example, if we use combinatorial time and perturb the filtration function, the output would be locally constant.

When extending function time to include contractions, we observe that there are some subtleties in defining function time for (f, g)-FB persistence, because the functions f and g can be independent. It can be the case that the values of g are mixed with the values of f on the real line, but to define a function time we would like the values of g to appear after the values of f. To get a well-defined notion of function time, we require f and g to both be positive, and that the corresponding function time of (a) an inclusion step at i in (f, g)-FB persistence to be the i-th value of f and (b) a contraction step at $n + i$ in (f, g)-FB persistence to $\max(f)+$ the i-th value of g.

Theorem 2 (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 $i \geq 0$:*

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

We remark that establishing an appropriate **function time** for hourglass persistence is challenging, because the function values for each step is not as canonical since we are interleaving the steps of contractions and inclusions. Hourglass persistence does satisfy the condition of what Chazal et al. (2009) called "tame", which has a **stability** in combinatorial time (see Theorem 4.4 therein).

7 ALGORITHM DESIGN AND EXPERIMENTS

We now present a practical algorithmic framework that supports any filtration scheme composed of a sequence of inclusions and contractions, assuming that all contraction intermediates have already appeared previously in the forward filtration.

7.1 FORWARD INCLUSION WITH AUXILIARY BOOKKEEPING.

In addition to the standard union–find structure, which incrementally tracks connected-component memberships during the forward filtration, we maintain two further structures: (i) neighborhood information for the spanning forest being built, and (ii) a fundamental cycle basis over $\mathbb{F}_2$. Each new edge $e = (u, v)$ is then handled in two cases:

Algorithm 1 FORWARDINCLUSION

Input: Filtration f; Graph G
Output: PD_0, PD_1, cycle basis $\mathcal{B}$, union–find UF
Initialize UF on V; $\mathsf{PD}_0, \mathsf{PD}_1, \mathcal{B} \leftarrow \emptyset$
Sort edges $e_1, \ldots, e_m$ by $f(e_j)$
for $j = 1..m$ **do**
  $(u, v) \leftarrow e_j$
  if $\mathsf{UF}.\mathrm{find}(u) = \mathsf{UF}.\mathrm{find}(v)$ **then**
    // Cycle-creating edge
    Build $\gamma \in \{0, 1\}^m$ from e_j and path $u \rightsquigarrow v$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\gamma\}$; $\mathsf{PD}_1[\gamma] \leftarrow (f(e_j), \infty)$
  else
    $r_u \leftarrow \mathsf{UF}.\mathrm{find}(u)$; $r_v \leftarrow \mathsf{UF}.\mathrm{find}(v)$
    $y \leftarrow \arg\max_{r \in \{r_u, r_v\}} f(r)$
    $\mathsf{PD}_0 \leftarrow \mathsf{PD}_0 \cup \{(f(y), f(e_j))\}$
    $\mathsf{UF}.\mathrm{merge}(u, v)$
    // record mutual neighbors in spanning forest
    $\mathsf{UF}.\mathrm{nbrs} \cup \{u \leftrightarrow v\}$
for each root r of UF **do** add $(f(r), \infty)$ to PD_0
return $\mathsf{PD}_0, \mathsf{PD}_1, \mathcal{B}, \mathsf{UF}$

(1) If u and v are in different components, e is a *spanning-tree edge*. We update the union–find and record u and v as neighbors in the spanning forest, thereby incrementally extending the spanning structure maintained during the filtration.

(2) If u and v are already connected, e is a *cycle-creating edge*. Together with the unique forest path $u \rightsquigarrow v$, this defines a new cycle C_{k+1}, given an existing basis $\{C_1, \ldots, C_k\}$. Since e does not appear in any earlier cycle, C_{k+1} is linearly independent of $\{C_1, \ldots, C_k\}$ in the cycle space over $\mathbb{F}_2$. The basis is thus extended to $\{C_1, \ldots, C_k, C_{k+1}\}$, and a new interval $(f(e), \infty)$ is added.

We denote by PD_0 and PD_1 the persistence diagrams of H_0 and H_1, respectively. Algorithm 1 describes the forward stage of our framework. Its input is a filtration function $f : V \cup E \to \mathbb{R}$ on a graph $G = (V, E)$, and its output consists of the persistence diagrams PD_0 and PD_1, a fundamental cycle basis $\mathcal{B}$ represented by indicator vectors in $\{0, 1\}^m$, and a union–find structure UF that maintains connected-component information and parent pointers for the spanning forest.

7.2 Backward contraction with supernode bookkeeping.

The backward stage uses a contraction function g to order contractions. We maintain a *supernode* that accumulates contracted vertices; a vertex merges when scheduled by g, and an edge contracts once both endpoints reside in the supernode. Algorithm 2 summarizes the contraction process.

Vertex contractions. The contraction of a vertex into the supernode has two effects: **(1)** If the vertex belongs to a different connected component, the younger component is killed and its H_0 interval is closed. **(2)** If the vertex lies in the same component, a new *supernode cycle* is created. Unlike forward cycles, these are not tied to any edge but arise from merging disconnected subgraphs of the same component.

Edge contractions. An edge is contracted once it becomes a self-loop on the supernode. Each such contraction kills one cycle: **(1)** If the edge participates in some forward cycle, we remove it from all cycle indicators in $\mathcal{B}$ and reduce the basis. If a younger cycle becomes dependent on older ones, it is killed and its H_1 interval is closed. **(2)** If no forward cycle is removed, the contraction kills the most recent supernode cycle. In either case, the contraction assigns a finite death time to an H_1 interval, completing the bookkeeping of backward updates.

Algorithm 2 BackwardContraction

Input: contraction function g; persistence data $(\mathsf{PD}_0, \mathsf{PD}_1, \mathcal{B})$ and union–find UF from Alg. 1
Output: updated $\mathsf{PD}_0, \mathsf{PD}_1$ with finite deaths
Initialize supernode $S \leftarrow \emptyset$, stack B, list L
for elements $x \in V \cup E$ in order of $g(x)$ **do**
  if $x \in V$ **then** ▷ Node contraction
    $S \leftarrow S \cup \{x\}$
    if $\mathsf{UF}.\mathrm{find}(x) \neq \mathsf{UF}.\mathrm{find}(S)$ **then**
      // kill younger component y
      $\mathsf{PD}_0 \leftarrow \mathsf{PD}_0 \cup \{(f(y), g(x))\}$
    else
      $B.\mathrm{push}(g(x))$ // Birth supernode cycle
    $\mathsf{UF}.\mathrm{merge}(x, S)$
  else ▷ Edge contraction
    // $x = e = (u, v) \in E$ with $u, v \in S$
    remove e from all $\gamma \in \mathcal{B}$; reduce basis
    if some γ becomes dependent **then**
      // Close forward cycle
      $\mathsf{PD}_1[\gamma] \leftarrow (\mathrm{birth}(\gamma), g(x))$
      $\mathcal{B} \leftarrow \mathcal{B} \setminus \{\gamma\}$
    else
      // Close supernode cycle
      $\tau \leftarrow B.\mathrm{pop}(); \quad L \leftarrow L \cup \{(\tau, g(x))\}$
$\mathsf{PD}_1 \leftarrow \mathsf{PD}_1 \cup L$
return $\mathsf{PD}_0, \mathsf{PD}_1$

7.3 Experimental setup and results

Datasets. We evaluate on four standard graph classification datasets (Morris et al., 2020): NCI109, PROTEINS, IMDB-BINARY, and NCI1 (Accuracy); a graph-regression dataset ZINC (Dwivedi et al., 2023) (MAE) and the OGBG-MOLHIV (Hu et al., 2020) molecular property dataset (AUC). All methods use the same GIN (Xu et al., 2018) or GCN (Kipf, 2016) backbone and the same DeepSets (Zaheer et al., 2017) pooling for encoding persistence diagrams.

Baselines and ablations. We compare five variants of persistence-based representations. (i) **PH** (Horn et al., 2021) uses a fixed vertex filtration and assigns each edge the maximum filtration of its endpoints, producing standard inclusion-based persistence. (ii) **RePHINE** (Immonen et al., 2023) learns both vertex and edge filtrations and, importantly, augments every 0D persistence point with the filtration value of the earliest incident edge, providing additional early-connectivity information. (iii) **Fwd-only** learns a general vertex–edge filtration but does *not* include RePHINE's augmentation, and disables all contractions; this isolates the effect of learning the forward filtration alone. (iv) **Bwd-only** learns only the contraction schedule, isolating the backward component of our design. (v) **Ours** implements the full learnable inclusion–contraction (forward–backward) framework, jointly learning both vertex/edge filtrations and contraction order.

Observations. Table 1 reports performance across all six benchmarks. We observe: **(1)** Our method achieves the best or second-best performance in 9 of 12 settings, underscoring the value

Table 1: Comparison of PH variants across six datasets using GIN and GCN backbones. Classification accuracy and AUC scores are reported in percentage (%, ↑) and ZINC regression evaluation in MAE (↓). Best and second-best results per row are shown in **bold** and underline, respectively.

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

of jointly learning both inclusion and contraction schedules. **(2) RePHINE** remains a strong baseline and consistently outperforms standard PH in nearly all cases, reflecting the benefit of learning both vertex and edge filtrations. **(3)** The ablations further clarify the role of each component: **Fwd-only**, which learns vertex–edge filtrations but disables contractions, typically improves over standard PH; **Bwd-only**, which learns only contraction order, also improves over PH on several datasets and performs roughly on par with **Fwd-only** overall, with no clear winner between them. **(4) RePHINE** significantly outperforms both **Fwd-only** and **Bwd-only** on ZINC and MOLHIV, highlighting the effectiveness of its additional 0D augmentation in capturing early connectivity structure. **(5)** Our full model almost always surpasses both ablations, demonstrating that inclusion and contraction encode complementary structural information, and that jointly learning them is essential for achieving the strongest performance across classification, regression, and molecular prediction tasks.

7.4 Comparison with extended persistence

We further compare our framework to methods based on extended persistence, which incorporate both sublevel and superlevel information. PersLay (Carriere et al., 2020) implements this by first computing extended-persistence diagrams and then applying a learnable layer on top. In contrast, motivated by Proposition 3, which shows that classical extended persistence has the same expressive power as $(f, -f)$–FB persistence, we integrate extended persistence directly into our forward–backward framework by using $-f$ as a backward schedule. In Table 2, we observe: **(1)** Our $(f, -f)$ adaptation of extended persistence consistently outperforms PersLay under both GIN and GCN backbones, showing that embedding extended-persistence structure directly into the forward–backward framework is more effective than processing extended-persistence diagrams through a post-hoc learnable layer. **(2)** Our full (f, g)-forward–backward model further exceeds both PersLay and the extended variant, indicating that jointly learning inclusion and contraction provides expressive topological features beyond what extended persistence alone can capture.

Table 2: Accuracy (%) with GIN and GCN backbones. Best score per row is in **bold**.

	GIN			GCN		
Dataset	ExtP	PersLay	Ours	ExtP	PersLay	Ours
NCI109	**78.21**	68.28	**78.21**	**77.48**	68.28	**77.48**
PROTEINS	**74.11**	66.07	73.21	**72.32**	66.07	**72.32**
IMDB-B	63.00	70.00	**73.00**	66.00	**70.00**	**70.00**
NCI1	78.59	68.86	**81.51**	80.05	68.86	**80.78**

8 Conclusion

We formalize *backward*, *(f,g)-forward–backward*, and *hourglass* persistence and develop a systematic expressivity theory, certified by minimal witness graphs and constructive proofs, accompanied by algorithms that realize the general (f, g) framework. These constructions extend to simplicial and cellular complexes and admit a functional stability guarantee, offering principled tools for inclusion–contraction schedules beyond the classical PH. While hourglass persistence satisfies combinatorial stability, establishing a functional stability result is a compelling direction for future work.

Ethics Statement

This paper presents work whose goal is to advance the interaction of topological descriptors with neural networks. Topological neural networks have many usage in the applied science, including medicine, finance, recommendation algorithms. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Reproducibility Statement

We provide an anonymized artifact at https://anonymous.4open.science/r/Hourglass-4AC2 containing everything needed to reproduce our results: complete source code for all methods in PyTorch and C++; scripts to download datasets and generate deterministic splits with fixed seeds; exact configuration files and a pinned `environment.yml` with a README giving step-by-step commands to run all experiments; config files specifying hyperparameters and training protocols (optimizer, learning rate, batch size, epochs, scheduler, early stopping, and seeds) and a Jupyter notebook to generate results table. The appendix contains the full statements and proofs of all theorems, and the paper includes extensive diagrams that illustrate the separation examples and clarify our theoretical claims.

References

Toshitaka Aoki, Emerson G. Escolar, and Shunsuke Tada. Bipath persistence. *Japan Journal of Industrial and Applied Mathematics*, 42(1):453–486, Jan 2025. ISSN 1868-937X. doi: 10.1007/s13160-024-00681-3. URL https://doi.org/10.1007/s13160-024-00681-3.

Rubén Ballester and Bastian Rieck. On the expressivity of persistent homology in graph learning, 2024. URL https://arxiv.org/abs/2302.09826.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montufar, and Michael M. Bronstein. Weisfeiler and lehman go cellular: CW networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=uVPZCMVtsSG.

Gunnar Carlsson and Vin de Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10(4):367–405, Aug 2010. ISSN 1615-3383. doi: 10.1007/s10208-010-9066-0. URL https://doi.org/10.1007/s10208-010-9066-0.

Mathieu Carriere, Frederic Chazal, Yuichi Ike, Theo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2786–2796. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/carriere20a.html.

Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Y. Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, SCG '09, pp. 237–246, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585017. doi: 10.1145/1542362.1542407. URL https://doi.org/10.1145/1542362.1542407.

Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. SpringerBriefs in Mathematics. Springer, 2016. ISBN 978-3-319-42544-3. doi: 10.1007/978-3-319-42545-0. URL https://doi.org/10.1007/978-3-319-42545-0.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *NeurIPS*, 2020. URL http://dblp.uni-trier.de/db/conf/nips/neurips2020.html#Chen0VB20.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, Dec 2006. doi: 10.1007/s00454-006-1276-5.

David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using poincaré and lefschetz duality. *Found. Comput. Math.*, 9(1):79–103, February 2009. ISSN 1615-3375.

Michel Coste. An Introduction to O-minimal Geometry. 2002.

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

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, Nov 2002. ISSN 1432-0444. doi: 10.1007/s00454-002-2885-2. URL https://doi.org/10.1007/s00454-002-2885-2.

Herbert Edelsbrunner and John Harer. Persistent homology—a survey. *Contemporary Mathematics*, pp. 257–282, 2008. doi: 10.1090/conm/453/08802.

Herbert Edelsbrunner and Dmitriy Morozov. Persistent homology: Theory and practice. In *Proceedings of the European Congress of Mathematics*, 2012.

V. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning (ICML)*, 2020.

Robert Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2007.

William L. Hamilton. *The Graph Neural Network Model*, pp. 51–70. Springer International Publishing, Cham, 2020. ISBN 978-3-031-01588-5. doi: 10.1007/978-3-031-01588-5_5. URL https://doi.org/10.1007/978-3-031-01588-5_5.

Jakob Hansen and Robert Ghrist. Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology*, 3(4):315–358, August 2019. ISSN 2367-1734. doi: 10.1007/s41468-019-00038-7. URL http://dx.doi.org/10.1007/s41468-019-00038-7.

Allen Hatcher. *Algebraic Topology*. New York : Cambridge University Press, 2002.

Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. In *ICML*, 2022.

Max Horn, Edward De Brouwer, Michael Moor, Yves Moreau, Bastian Rieck, and Karsten Borgwardt. Topological graph neural networks. *arXiv preprint arXiv:2102.07835*, 2021.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Johanna Immonen, Amauri Souza, and Vikas Garg. Going beyond persistent homology using persistent homology. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 63150–63173. Curran Associates, Inc., 2023.

Mattie Ji, Amauri H. Souza, and Vikas Garg. Graph persistence goes spectral, 2025. URL https://arxiv.org/abs/2506.06571.

TN Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Michael Lesnick. Notes on multiparameter persistence (for amat 840). Course notes, University at Albany, SUNY, 2025. URL https://www.albany.edu/~ML644186/840_2022/Math840_Notes_22.pdf. Last updated April 12, 2025.

Pan Li and Jure Leskovec. *The Expressive Power of Graph Neural Networks*, pp. 63–98. Springer Nature Singapore, Singapore, 2022. ISBN 978-981-16-6054-2. doi: 10.1007/978-981-16-6054-2_5. URL https://doi.org/10.1007/978-981-16-6054-2_5.

Cordelia Laura Elizabeth Henderson Moggach. *A Homotopical Categorification of the Euler Calculus*. Ph.d. thesis, University of Liverpool, 2020. URL https://docslib.org/doc/8977190/classifying-types-topics-in-synthetic-homotopy-theory. Minor modifications: Grenoble, 28 July 2020.

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1), August 2017. ISSN 2193-1127. doi: 10.1140/epjds/s13688-017-0109-5. URL http://dx.doi.org/10.1140/epjds/s13688-017-0109-5.

Theodore Papamarkou, Tolga Birdal, Michael M. Bronstein, Gunnar E. Carlsson, Justin Curry, Yue Gao, Mustafa Hajij, Roland Kwitt, Pietro Lio, Paolo Di Lorenzo, Vasileios Maroulas, Nina Miolane, Farzana Nasrin, Karthikeyan Natesan Ramamurthy, Bastian Rieck, Simone Scardapane, Michael T Schaub, Petar Veličković, Bei Wang, Yusu Wang, Guowei Wei, and Ghada Zamzmi. Position: Topological deep learning is the new frontier for relational learning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39529–39555. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/papamarkou24a.html.

The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.11.0 edition, 2025. URL https://gudhi.inria.fr/doc/3.11.0/.

Patrick Schnider. Chapter 6: Distances and stability. Lecture notes for the course "Introduction to Topological Data Analysis (TDA)", ETH Zürich, 2024. URL https://ti.inf.ethz.ch/ew/courses/TDA24/Chapter6.pdf.

Primoz Skraba and Katharine Turner. Notes on an elementary proof for the stability of persistence diagrams, 2021. URL https://arxiv.org/abs/2103.10723.

Yuxuan Song, Jingjing Gong, Hao Zhou, Mingyue Zheng, Jingjing Liu, and Wei-Ying Ma. Unified generative modeling of 3d molecules with bayesian flow networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NSVtmmzeRB.

Behrooz Tahmasebi, Derek Lim, and Stefanie Jegelka. Counting substructures with higher-order graph neural networks: Possibility and impossibility results, 2021. URL https://arxiv.org/abs/2012.03174.

Lou van den Dries. *Tame Topology and O-minimal Structures*, volume 248. Cambridge university press, 1998.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *ICLR*, 2022.

Minkai Xu, Alexander S. Powers, Ron O. Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, Yusu Wang, and Chao Chen. Neural approximation of graph topological features. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=qwjrO7Rewqy.

Zuoyu Yan, Qi Zhao, Ze Ye, Tengfei Ma, Liangcai Gao, Zhi Tang, Yusu Wang, and Chao Chen. Enhancing graph representation learning with localized topological features. *Journal of Machine Learning Research*, 26(5):1–36, 2025. URL http://jmlr.org/papers/v26/23-1424.html.

Chaolong Ying, Xinjian Zhao, and Tianshu Yu. Boosting graph pooling with persistent homology, 2024. URL https://arxiv.org/abs/2402.16346.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Bingxu Zhang, Changjun Fan, Shixuan Liu, Kuihua Huang, Xiang Zhao, Jincai Huang, and Zhong Liu. The expressive power of graph neural networks: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 37(3):1455–1474, March 2025. ISSN 2326-3865. doi: 10.1109/tkde.2024.3523700. URL http://dx.doi.org/10.1109/TKDE.2024.3523700.

Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2896–2906. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/zhao20d.html.

Hourglass Persistence for Graphs, Simplices, and Cells (Appendix)

A Proofs for Section 3

In this section, we will provide the proofs for the results in Section 3.

Proof of Proposition 1. Let G be a path graph on 4 vertices colored in the order R, B, B, R. Let H be a star graph of 4 vertices such that the unique vertex with degree 3 has color B, and the other 3 vertices have color B, R, R (see Figure 3(a)). Consider a filtration of G and H by first spawning the induced subgraph on blue vertices, and then the induced subgraph on blue and red vertices. In other words, we have filtrations of the form

$$\emptyset \subset P_2 \subset G \text{ and } \emptyset \subset P_2 \subset H,$$

where P_2 is the path graph on two vertices both colored blue. The inclusion-based PH of both filtrations are the same (see Figure 4 in Immonen et al. (2023)). However for backward-based PH, we observe that $G/\operatorname{IC}_1(G)$ creates a cycle but $H/\operatorname{IC}_1(H)$ does not create a cycle. Thus, they can be told apart in the contraction stage.

For the second part, consider the following pair of graphs in Figure 3(b), equipped with a vertex-based filtration using degree. In this case, we observe that the filtration of G (left) and H (right) respectively is a sequence of discrete vertices until all the edges are spawned at the last time. This is because every edge is connected to a vertex of maximal degree equal to 3 for both G and H. Thus, the backward persistence for both G and H would be to quotient out the entire graph, which cannot distinguish G and H since they have the same number of components and independent cycles. On the other hand, FB-persistence can tell G and H apart since in the filtration step spawning degree 1 vertices, 3 vertices are spawned for G while 2 vertices are spawned for H. □

Proof of Theorem 1. We first see how the FB-persistence diagram can recover the forward persistence diagrams and the backward persistence diagrams. Indeed, at n be the time the filtration completes and we are about to start contraction. The tuples that are born before or on time n (which appears at the last step of the filtration) are of the form (b, d) where d is possibly greater than n. From here, we can recover the persistence diagram in the forward filtration as by sending each pair

$$(b, d), \text{ with } b \leq n \mapsto \begin{cases} (b, d) \text{ if } d \leq n \\ (b, \infty) \text{ if } d > n \end{cases}$$

where we note that if the pair (b, d) has death after time n, than it must have died in the contraction step, which means that the feature lived to ∞ in the filtration step.

Similarly, we may obtain the backward persistence diagram by focusing on the persistent pairs (b, d) such that $d > n$. We can recover them by the function

$$(b, d) \text{ with } d > n \mapsto \begin{cases} (0, d), \text{ if } b \leq n \\ (b, d), \text{ if } b > n \end{cases}.$$

The reason why is because all the features that have not died yet at the start of contraction in FB-PH are the same as the features that are born at the initial step of the contraction scheme in backward PH. Thus, the elder rule applies to the same features when we decide what pairs to kill off. The only difference is that in backward PH, the features from the filtration steps all appeared at the same time, so it does not matter which one to kill if we do have to kill them, but for FB persistence we would need to be more careful. Evidently, though, this gives a straightforward reduction of backward PH from FB PH.

To see that FB PH is strictly more expressive than forward and backward PH, we consider the graphs G and H in Figure 3(c) using degree as a vertex-based filtration. Using the gudhi library (Project, 2025)'s persistence pairs computation, we can directly apply it to the following code in `Python`:

```python
import gudhi

VG = [0, 1, 2, 3, 4, 5, 6, 7]
EG = [(0, 1), (1, 2), (1, 3), (2, 3), (3, 4), (4, 5), (4, 6), (4, 7), (6, 7)]
G_values = [1, 3, 2, 3, 4, 1, 2, 2]

VH = [0, 1, 2, 3, 4, 5, 6, 7]
EH = [(0, 1), (1, 2), (1, 3), (2, 3), (3, 4), (3, 5), (5, 6), (5, 7), (6, 7)]
H_values = [1, 3, 2, 4, 1, 3, 2, 2]

# Compute PDs for G
stG = gudhi.SimplexTree()
for i in range(0, len(VG)):
  current_v = VG[i]
  v_val = G_values[i]
  stG.insert([current_v], filtration=v_val)

# Adding edges
for i in range(0, len(EG)):
  current_e = EG[i]
  e_val = max(G_values[current_e[0]],G_values[current_e[1]])
  stG.insert(current_e, filtration=e_val)

stG.make_filtration_non_decreasing()
G_dgms = stG.persistence(min_persistence=-1, persistence_dim_max=True)
print(G_dgms)

# Compute PDs for H
stH = gudhi.SimplexTree()
for i in range(0, len(VH)):
  current_v = VH[i]
  v_val = H_values[i]
  stH.insert([current_v], filtration=v_val)

# Adding edges
for i in range(0, len(EH)):
  current_e = EH[i]
  e_val = max(H_values[current_e[0]],H_values[current_e[1]])
  stH.insert(current_e, filtration=e_val)

stH.make_filtration_non_decreasing()
H_dgms = stH.persistence(min_persistence=-1, persistence_dim_max=True)
print(H_dgms)

print(G_dgms == H_dgms)
```

The output would say that for graphs have the persistence diagrams of the form [(1, (3.0, inf)), (1, (4.0, inf)), (0, (1.0, inf)), (0, (1.0, 4.0)), (0, (2.0, 4.0)), (0, (2.0, 3.0)), (0, (2.0, 2.0)), (0, (3.0, 3.0)), (0, (3.0, 3.0)), (0, (4.0, 4.0))].

Now we show that they have the same backward persistence diagrams. At the initial time, they have the same number of connected components and cycles, so there are no distinctions for G and H. In the first step, we contract the $\mathrm{IC}_3(G)$ or $\mathrm{IC}_3(H)$ (which is the closed star containing the unique vertex labeled with the value 4). This is a connected subtree for both G and H, so the contraction does not produce non-trivial persistence pairs. Then we contract $\mathrm{IC}_2(G)$ (or $\mathrm{IC}_2(H)$), which would kill a cycle on both sides. Since the birth time of the cycles are the same, we just mark one of the $(0, -)$ to die at the time. But by the time we contract $\mathrm{IC}_1(G)$ (or $\mathrm{IC}_1(H)$), we will kill the remaining cycle on both sides, which marks another one, and we are done after the last step. Thus, we see that they will have the same backward persistence diagram.

Now to see that G and H have different FB-persistence, we note that the birth time of the two cycles when contracting $\mathrm{IC}_2(G)$ (or $\mathrm{IC}_1(H)$) are different. For G, the cycle being contracted is born when the degree 3 vertices are spawned. For H, the cycle being contracted is born when the degree 4 vertices are spawned. This would create a different persistent pair and hence FB-persistence can tell them apart. □

Proof of Proposition 2. Clearly FB-persistence is an example of hourglass persistence, so hourglass has at least as much expressivity as FB. Let G, H be graphs constructed in the NetworkX library, with filtration function f being degree-based vertex-level filtrations, as:

```
G = nx.from_edgelist([(0, 1), (1, 2), (2, 3), (1, 4), (4, 5)])
H = nx.from_edgelist([(0, 1), (1, 2), (1, 3), (3, 4), (4, 5)])
```

See Figure 3(d) for a visualization of the two graphs above. We wish to show they have the same FB-persistence but different hourglass persistence.

In this case, we note that the two graphs would have the same FB-persistence diagram with respect to f. Indeed, at $t = 1$, all the vertices labeled 1 are spawned for both G and H, which gives three copies of the form $(1, -)$. At $t = 2$, the induced subgraph on vertices labeled 1 and 2 appears. We note here that on both G and H, by the elder's rule, the two new vertices are both killed, thus both diagrams have the form

$$(1, -), (1, -), (1, -), (2, 2), \text{ and } (2, 2).$$

Finally, at $t = 3$, the vertex labeled 3 appears which makes both G and H connected at the step. The vertex labeled 3 dies at the same time, and two of the tuples labeled $(1, -)$ also dies. This gives

$$(1, -), (1, 1), (1, 1), (2, 2), (2, 2), \text{ and } (3.3).$$

Now we enter the contraction steps. Indeed, we first contract $\mathrm{IC}_2(G)$ (or the corresponding version for H), evidently we are contracting isomorphic connected subtrees on G and H, so the process would neither create non-trivial cycles or non-trivial steps. Then we contract $\mathrm{IC}_1(G)$ (or the corresponding version for H), for H this is a subtree, so the contraction does not create new non-trivial tuples. For G, one might be tempted to think that this is contracting a disconnected graph. However, since we contracted $\mathrm{IC}_2(G)$ already, the two edges are actually connected here and forms a tree, so this also does not incur a change.

Finally, at the end, we remark the unique remaining tuple to die at ∞.

$$(1, \infty), (1, 1), (1, 1), (2, 2), (2, 2), \text{ and } (3.3).$$

Remark: Note also that in the proof here we computed the persistence tuples in function time as opposed to combinatorial time (see Section 6 for a discussion), but this does not matter in terms of expressivity.

Finally, to see why hourglass persistence can tell them apart, we see that if we spawn $\mathrm{IC}_1(G)$ (resp. $\mathrm{IC}_1(H)$) first in the sequence, then they would incur two connected components for G but only one for H, so hourglass persistence can tell them apart. □

B PROOFS FOR SECTION 4

We split the discussion for the proofs for Section 4 into three parts:

1. In Appendix B.1, we establish a lemma that will be helpful in streamlining the proof of Proposition 6 in Appendix B.2. For completeness, we will also use a special case of this lemma to show how persistence modules can derive the usual interpretation of persistence pairs for a graph filtration. We will also introduce some concepts helpful in understanding the proof of Proposition 6.
2. In Appendix B.2, we prove Proposition 3, Proposition 5, and Proposition 6 (ie. the expressivity parts of the section).
3. In Appendix B.3, we will explain the two algorithms for Proposition 4 and prove it.

B.1 KEY LEMMA AND THE USUAL PERSISTENT PAIRS INTERPRETATION FOR GRAPH FILTRATIONS

Before we give a proof of Proposition 6, we first note that the description of death times for connected components holds on an elevated generality that both the proofs of Proposition 6 and (later) Proposition 7 would use, so we might as well extract it out as a formal lemma.

For completeness, we will also give a self-contained proof of how Definition 2 recovers the usual way to compute persistent pairs for graph filtrations (for instance, see Section 4 of Horn et al. (2021) or Algorithm 1 of Immonen et al. (2023)).

The lemma is as follows.

Lemma 1. *Let $X_\bullet = (\emptyset = X_{-1} \to X_0 \to X_1 \to ... \to X_N)$ be a sequence of inclusions and contractions of graphs (ie. the set-up of hourglass persistence). Note that by necessity the first step $X_{-1} \to X_0$ has to include in a graph. The persistence pairs[1] for $H_0(X_\bullet)$ can be computed algorithmically as follows - the method of which we called "keeping track of one vertex representative per component":*

1. *In the filtration step $X_{-1} \to X_0$, we pick 1 vertex per connected component in X_0 and mark a tuple $(0, -)$ corresponding to that. We fix one of these vertices to be called the **supernode** (denoted $*$), in the sense that any time we compare a vertex v with $*$ to decide which one to kill off, we always kill off v.*
2. *For $i \geq 0$, if the step $X_i \to X_{i+1}$ is a filtration, then one treats this in algorithmic time as a procedure to spawn every new vertex that appears first and mark them as tuples $(i, -)$, and then spawn edges between them. For each edge between spawned, if an edge is joined between 2 different components represented by vertices v and w, we mark the vertex born later to die at time i and pick v as the representative of the new component. (Since we do not focus on trivial deaths, we can discard them after this step). If the two vertices are born at the same time, we randomly pick 1 to kill off unless one of them is the supernode $*$, in which case we always kill off the other vertex.*
3. *For $i \geq 0$, if the step $X_i \to X_{i+1}$ is a contraction, say contracting a subgraph G. We loop through each connected component X_j of G, we find the vertex v_j representing the component X_j belongs in and kill v_j unless (1) it got killed already in some index $j' < j$ when looping through the X_j's or (2) it is the supernode $*$.*
4. *After $i = N$, for every tuple not dead yet, we mark it to die at time ∞.*

Remark 2. *Note that the statement of Lemma 1 does not require X_N to terminate with only the point $*$ left, so the case of hourglass persistence in Definition 7 is a special case in this lemma. A more specific case of Lemma 1 is when the maps are all inclusions. This will recover the usual procedure to compute the 0-dimensional PH of a graph filtration (as we just ignore Step 3).*

Proof of Lemma 1. Indeed let us recall from Remark 1 how one way to compute persistence diagrams are done. Let us interpret how the remark tells us how to compute the quantities here explicitly:

[1] Recall here we are not accounting for trivial deaths

1. We pick a non-zero vector v of $H_0(X_0)$, and consider the sequence of linear subspaces generated by the iterated image of v in $H_0(X_\bullet)$ until it becomes 0.

2. Then we remove this sequence of linear subspaces off of $H_0(X_\bullet)$. If there is still a non-zero vector w in $H_k(X_0)$, we pick w and repeat the process (if the linear map sends w outside. Otherwise, we choose a non-zero vector from what is left in $H_k(X_1)$ (if it exists) and look at its iterated images ahead, and so on.

Let us also recall the following standard fact in algebraic topology - suppose $f : A \to B$ is a map of (say) cellular complexes, then the induced map $f_* : H_0(A; \mathbb{Z}/2) \to H_0(B; \mathbb{Z}/2)$ can be described exactly as follows - recall $H_0(A; \mathbb{Z}/2)$ and $H_0(B; \mathbb{Z}/2)$ are respectively isomorphic to the direct sum of copies over $\mathbb{Z}/2$ indexed over their path-connected components. For each path component A_i, let 1_{A_i} be the unique non-zero vector representing that component. The continuous image of a path-connected component is path-connected, so $f(A_i)$ lies in a path-connected component of B, say B_j, then the map described above sends $f_*(1_{A_i}) = 1_{B_j}$.

Remark 3. *This description might deceptively suggest that f_* is not the zero map on all of $H_0(A; \mathbb{Z}/2)$, but this is clearly not the case, it just happens that we chose a convenient basis on both sides such that f_* is not zero on the chosen basis vectors. The subtlety in the persistence calculation is that we want to choose the basis in the $H_0(-)$ of the next space consistently with the basis of the previous one to nicely compute the persistence pairs.*

By the remark in the line above, though, we see that it follows that there exists a non-zero vector $v_1 \in H_0(X_0)$ (as long as X_0 is not empty, by requirement) such that the successive images of v are never 0 when passing through $H_0(X_\bullet)$, thanks to the convenient choice of basis, and v can in fact come from representing some connected component. We then choose v here to represent the **supernode** $*$. In the next step of the remark, we then remove the subspace generated by successive images of $*$ off $H_0(X_0)$, and if the next vector we pick lands in this subspace, we consider it dead. Note that on the level of H_0, a vector $1_C \in H_0(X_i)$ representing a component could land in this subspace at $H_0(X_{i+1})$ if and only if $f_i(C)$ belongs in the component of the supernode at time $i + 1$. Thus, we see that asking a vector representing 1_C to die if it enters this subspace corresponds **exactly** to the property of the supernode.

Now we proceed by induction and suppose the i-th vector v_i we pick following the remark (1) represents a component, (2) gives the correct persistence pair according to the outline of this lemma, and (3) has been picked according to the rules of Remark 1. Now we wish to show we can pick the $i + 1$-th vector v_{i+1} such that they still satisfy (1) and (2). Now we would stop if we have ran out of vectors to pick, so there is at least some non-zero vector in the complement of the subspace generated by vectors $v_1, ..., v_i$ and their iterated images (call this W_i for convenience). So there exists some vector v_{i+1} that represents a component C and say $v_{i+1} \in H_0(X_j)$.

By assumption, v_{i+1} cannot be in the image of any previous vector, so the component C must be a new component that appears from $X_{j-1} \to X_j$ (by the description of what the correspondent map in H_0 is like). Now we have two descriptions of how C gets killed:

1. C is killed in the sense of this lemma if and only if when (a) it merges with an earlier component during filtration or (b) a subgraph of C was asked to be contracted in the future.
2. Remark 1 says that C is killed if and only if the iterated image of 1_C gets landed in W_i.

We will go from the remark and see why it is equivalent to the first itme. Based on the description of how the induced map on $H_0(-)$ behaves, we see that 1_C can land into W_i at time $j' > j$ if and only if there is some $0 \leq k \leq i$ such that $f_{j'-1} \circ ... \circ f_j(C)$ and $f_{j'-1} \circ ... \circ f_j(C_k)$ represents the same connected component, where C_k is the component component v_k represents at the time it was born.

Without loss, we can choose j' to be the first time step $> j$ such that $(f_{j'-1})_* \circ ... \circ (f_j)_*(1_C)$ is in W_i and k to be the first v_k whose component C merges with. Then it follows that in the step $f_{j'-1} : X_{j'-1} \to X_{j'}$, C dies by mergining into $f_{j'-1} \circ ... \circ f_j(C_k)$. If $f_{j'-1}$ is a filtration, this can

only happen if some edge is spawned between them. If $f_{j'-1}$ is a contraction, this can only happen if $k = 0$ (ie. it goes into the supernode). Thus, we see that these are exactly the two conditions proposed by the scenario in the lemma. Furthermore, the birth and death times proposed by the lemma and Remark 1 agrees.

We then induct repeatedly ahead and conclude the proof. □

As noted earlier, Lemma 1 recovers the usual algorithm to compute persistence pairs for graph filtrations on the 0th dimensions. We now explain how to obtain the one for the 1st dimension from Definition 2. This will follow from the following more general fact.

Lemma 2. *Let X be a (finite) cell complex of dimension D and $X_\bullet := \emptyset = X_{-1} \to X_0 \to ... \to X_n = X$ be a filtration of X by cellular sub-complexes, then the persistence pairs of $H_D(X_\bullet)$ may be computed as follows:*

1. *At $t = 0$, we instantiate $\dim H_D(X_0)$ many tuples of the form $(0, \infty)$.*
2. *For $t > 0$, we instantie $\dim H_D(X_t) - \dim H_D(X_{t-1})$ many tuples of the form (t, ∞).*
3. *We end at $t = n$.*

We first look at how Lemma 2 specializes for graphs, before proving it. When $D = 1$, a 1-dimensional cell complex is the same as a graph, and the lemma recovers the usual way to calculate 1-dimensional PH's.

Indeed, $H_1(-)$ of a graph corresponds exactly to its list of independent cycles. For completeness, by "independent cycles", we mean the following interpretation:

Definition 10. *Let C, C' be two cycles of the graph G. The $C \boxtimes C'$ as the XOR of $E(C)$ and $E(C')$ (ie. their symmetric difference). For a list of cycles $D_1, ..., D_n$, the XOR span of the list is the following set:*

$$\{D_{i_1} \boxtimes D_{i_2} \boxtimes ... \boxtimes D_{i_k} \mid i_1, i_2, ..., i_k \subseteq \{1, ..., n\}\}$$

Definition 11. *Let $C_1, C_2, ..., C_k$ be a list of cycles of G. The list is a list of **independent cycles** if for any C_i, C_i is not contained in the XOR span of $C_1, ..., C_{i-1}, C_{i+1}, ..., C_k$*

The following is a well-known interpretation of $H_1(-)$ of a graph and can be interpreted as the definition.

Fact: The maximal list of independent cycles on G forms a basis for $H_1(G)$.

The usual algorithm for 1st dimensional persistence diagrams of a graph filtration keeps track of vertex representatives for the components, and they mark the birth-time of cycles of the form (t, ∞) at filtration time t for each edge drawn from a component to itself at time t.

Proposition 8. *Lemma 2 recovers the usual algorithm described above.*

Proof. Let us start from the algorithm side and work to Definition 2. Indeed, each (t, ∞) corresponds to an edge e that is born in G_t (the subgraph at time t). Choose C_e to be a cycle in G_t containing the edge e, we choose this for every such edge arising above.

By Lemma 2, it suffices for us to check that for each time step T, $H_1(G_t)$ has a basis being $\{C_e\}_{e \in E}$, where E is the collection of edge-creating cycle that is born in time $\leq T$. We first check that this is linearly independent. Indeed, we choose a total ordering $\leq$ on E such that $e_1 < e_2$ if e_1 appeared earlier than e_2 in the algorithm, (even if they are born at the same filtration time, there is an ordering of which one is born first in algorithmic time). Under this total ordering, we rewrite the elements of E into $e^1, e^2, ..., e^N$.

Clearly the list $\{C_{e^1}\}$ is independent, since C_{e^1} is a non-trivial cycle. Suppose by induction $\{C_{e^1}, ..., C_{e^k}\}$ is independent, we wish to show adding $C_{e^{k+1}}$ into the list remains independent.

Indeed, the XOR span of any sublist of $\{C_{e^1}, ..., C_{e^k}\}$ is always contained in the union of edges of $C_{e^1}, ..., C_{e^k}$'s, which does not contain the edge e^{k+1} because of the total ordering we picked. Thus, $C_{e^{k+1}}$, which contains e^{k+1} by construction, is not in the XOR span. Thus, we have verified this is a **list of independent cycles**.

Now, to show that this is a maximal list. We observe that removing the set E from G_t will turn G_t into a forest. This implies that there are no additional cycles left, which concludes the proof. □

Now we will prove Lemma 2.

Proof of Lemma 2. This lemma follows from the fact that the induced map $H_D(X_t) \to H_D(X_{t+1})$ by inclusion has to be injective for all t. Indeed, this comes from the long exact sequence in homology for the pair (X_{t+1}, X_t) (see Theorem 2.16 of Hatcher (2002)), since $H_{D+1}(X_{t+1}, X_t)$ is evidently zero as X_{t+1} and X_t are both at most D-dimensional. Since this is injective, Remark 1 tells us that a chosen component according to the steps of the remark can never merge into a pre-existing component, so all the vectors picked survive to ∞. Once we move from X_t to X_{t+1}, the new pairs that are created are exactly $\dim H_D(X_{t+1}) - \dim H_D(X_t)$ many pairs of the form $(t+1, \infty)$. This concludes the proof. □

B.2 Proofs for the Expressivity Parts of Section 4

Proof of Proposition 3. As defined in Section 2 of Yan et al. (2022), the extended persistence of a filtration function f is equivalent to the persistent pairs associated to the persistence module:

$$0 = H(G_{-\infty}) \to ...H(G_a) \to H(G) = H(G, G^{\infty}) \to ...H(G, G^a) \to H(G, G^{-\infty}).$$

Here, G_a means $\{x \in G \mid f(x) \leq a\}$ and G^a means $\{x \in G \mid f(x) \geq a\}$, and H denotes either H_0 or H_1. Here we note that the paper Yan et al. (2022) wrote $\emptyset = H(G_{-\infty}) = H(\emptyset)$, but some prefer the convention that $H(-)$ of the empty set is 0, as opposed to the empty set. One motivating reason is that the empty set is not a vector space. This does not affect the persistence pairs produced since they start at the non-zero parts.

Although a is indexed over the entire extended real numbers $[-\infty, +\infty]$, since f takes only finitely many values (or, if f is a real valued function taken on a graph G, viewed as a non-discrete topological space, the topological changes only occur when a vertex is spawned), the persistence module reduces to a finite length persistence module of the form

$$0 \to H(G_{a_0}) \to ... \to H(G_{a_n}) = H(G) \to H(G, G^{a_n}) \to ... \to H(G, G^{a_0}),$$

where $a_0 < ... < a_n$ is the sequence of filtration values of f. By Proposition 2.22 of Hatcher (2002), there is a morphism of persistence modules of the form:

$$\begin{array}{ccccccccccc}
0 & \longrightarrow & H(G_{a_0}) & \longrightarrow & ... & \longrightarrow & H(G_{a_n}) & \longrightarrow & H(G, G^{a_n}) & \longrightarrow & ... \longrightarrow H(G, G^{a_0}) \\
 & & = \downarrow & & = \downarrow & & = \downarrow & & \downarrow & & \downarrow \\
0 & \longrightarrow & H(G_{a_0}) & \longrightarrow & ... & \longrightarrow & H(G_{a_n}) & \longrightarrow & H(G/G^{a_n}) & \longrightarrow & ... \longrightarrow H(G/G^{a_0})
\end{array}$$

where the second row is analogous to the construction of (f, g)-FB persistence we did. Observe we can rewrite G^{a_i} as $\{x \in G \mid -f(x) \leq -a_i\}$. By and the successive union of the intermediate complexes arising in the definition of $(f, -f)$-FB persistence up to step i is exactly the same as G^{a_i}, so the persistence module of the second row arises exactly from the $(f, -f)$-FB persistence.

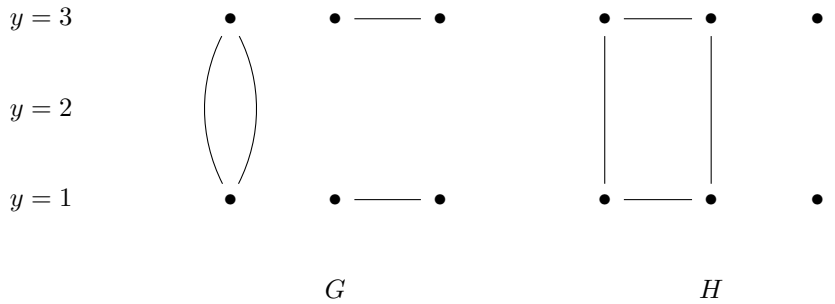
Proposition 2.22 of Hatcher (2002) also implies that the morphism above is an isomorphism of persistence modules if H is $H_1(-)$, so they will have the same 1-dimensional persistent diagrams.

If H is $H_0(-)$, then the maps $H(G, G^{a_i}) \to H(G/G^{a_i})$ injects onto a direct summand $\tilde{H}(G/G^{a_i})$ of $H(G/G^{a_i})$ such that $H(G/G^{a_i}) = \tilde{H}(G/G^{a_i}) \oplus \mathbb{Z}/2$. On the other hand, there is a very explicit interpretation on what the generator of the $\mathbb{Z}/2$ summand is - it is exactly the image of the chosen **supernode** (see Lemma 1). The difference here is that the supernode in the second row becomes the

only vertex feature that does not die, of the form $(0, \infty)$, and the supernode in the first row becomes the vertex feature of the form $(0, d)$ where d achieves the maximum death time among all vertices of the graph (concretely, it is the first i (from n to 0) such that G/G^{a_i} is a connected graph). By the decision rule we imposed on the supernode, we see that the first and second row agree on all 0-dimensional persistence pairs except for the supernode, but clearly we can invert from one to another.

This concludes the discussion that they have the same expressivity. □

Proof of Proposition 5. Indeed, consider the pair of graphs G and H from Figure 5 with the filtration function f being the vertex-based filtration function induced by the height function on the vertices. For convenience, we also redraw the two graphs below as:



In the forward time, we observe that the respective filtrations for G and H only change twice as $\emptyset = G_{-1} \subset G_0 \subset G_1 = G$ and $\emptyset = H_{-1} \subset H_0 \subset H_1 = H$, where G_0 and H_0 are identical. The same number of vertex births/deaths and cycle births occur at $y = 3$, and hence G and H cannot be told apart in forward time. If we apply the backward contraction with respect to $-f$ (ie. extended persistence), then at $y = 3$, the subgraph being contracted is the same, and they both only kill 1 connected component. No changes happen until we get back to $y = 0$, but by then the entire remaining graphs are contracted, and no differences are detected. Thus, we conclude that forward with respect to f + backward with respect to $-f$ cannot tell apart G and H.

On the other hand, we observe that FB-persistence can clearly tell them apart. This is because $\mathrm{IC}_1(G)$ has a cycle and $\mathrm{IC}_1(H)$ does not, so contracting G by $\mathrm{IC}_1(G)$ kills a cycle but contracting H by $\mathrm{IC}_1(H)$ does not. □

Proof of Proposition 6. From a similar proof to that of Theorem 1, we know that (f, f)-FB has at least the same expressivity as forward PH with f. Note that the explicit description of how to compute (f, f)-FB persistence means that it is actually a function of the filtraiton function f, so if the explicit description holds then forward PH with f is at least as expressive as (f, f)-FB persistence, which will show they have the same expressive power.

Thus, it suffices for us to verify this explicit description. The case for connected components is resolved by Lemma 1 already, so we will examine how to compute the case of cycles.

By a similar procedure to the explaination in Theorem 1 (essentially due to elder's rule), the cycles that appear in the contraction steps for (f, f)-FB persistence are exactly the cycles (that are not born at the beginning) in the backward persistence diagram that contracts the intermediate complexes in the order $\mathrm{IC}_0(G, f), \mathrm{IC}_1(G, f), ..., \mathrm{IC}_n(G, f)$. Note that a cycle can be born in the contraction stage if and only if we are being asked to contract two components that belong to the same connected component of the entire graph G (note that, without loss in algorithmic time, we can always contract two components at a time).

Since the intermediate complexes being contracted are in the same order they appear in the filtration, we see that the birth of cycles in the contraction steps corresponds exactly to the appearance of connected components in filtraiton that will be merged to connected components that are born earlier. The death time of such cycle also corresponds to when the two components actually merge. Perhaps one way to see this is to note that in the simplicial quotient interpretation (see Section 5),

this fills in a list of triangles to the disjoint base point v_+ on top of a path connects the two vertex representatives. Prior to the filling, the two vertex representatives would each have an edge to v_+ (which is a cycle since they are in the same component for the entire graph G). The filling of the triangles here would kill the cycle. This proves the case for cycles born in contraction.

Finally, for the cycles born in filtration, we would like to predict their death time in contraction. Recall from the proof of Proposition 8 that there is an explicit description of the compatible basis of $H_1(G^{\text{fil}}_\bullet)$, where $G^{\text{fil}}_\bullet$ denotes the sequence of inclusions of subgraphs induced by the filtration function f. The basis of $H_1(G)$ correspond to edge-creating cycles, and an explicit ordered basis of independent cycles $C_1 < ... < C_N$ can be found by choosing cycles C_i that contain the edge-creating cycles e_i in an inductive way (see the proof of Proposition 8 for more details). The upshot is that this would give a linearly independent list of cycles because the edge-creating cycle C_i contains is not contained in C_j for $j < i$.

In the contraction step, we observe that a necessary condition for the list $\{C_1, ..., C_N\}$ to degenerate (ie. become linearly dependent) is when some cycle creating edge e_i is contracted. Furthermore, if e_i is the last edge in the cycle C_i to be contracted, the cycle C_i would die right there. The diffcult with general (f, g)-FB persistence is that, due to the arbitrariness of g, often times the cycle creating edge e_i is not the last edge being contracted in C_i. In the case of (f, f)-FB persistence, however, we observe that we can always choose e_i to die last in algorithmic time. Thus, this shows that each cycle C_i born at time k dies exactly at time $n + k$ (when the same cycle is being asked to be contracted with respect to f). □

B.3 Algorithm to Compute (σ, τ)-Forward-Backward Filtrations

In this section, we will prove Proposition 4 and explain the algorithms behind in the case when f is vertex-based. We also remark that a similar algorithm exists in the arbitrary case when f is not vertex-based, just one also has to carefully permute the edge based values. We chose to present the case when f is vertex-based for simplicity.

Before we start explaining the algorithm, we first explain why (f_1, f_2) exists in general. Indeed, this will follow immediately from the following more general fact.

Lemma 3. *Let G be a graph and $G_\bullet = (\emptyset = G_{-1} \subset G_0 \subset G_2 \subset ... \subset G_n)$ be a strict inclusion of subgraphs on G, then there exists a filtration function $f : G \to \mathbb{R}$ with filtration values $a_0 < ... < a_n$ such that $f^{-1}((-\infty, a_i]) = G_{a_i}$.*

Proof. For each vertex or edge $x \in G$, we define $f(x)$ to be $\min_{i \in \{0,1,...,n\}} x \in G_i$. Since $G_\bullet$ is an inclusion of subgraphs, an edge e cannot appear earlier than the vertices that support it, so it follows that f is a filtration function. This concludes the proof. □

From now on in this subsection, we assume that f is vertex-based for simplicity. Let us first look at the algorithm for FB-persistence specifically and see why it is only linear time.

Proposition 9. *Let f be a filtration function, then there exists a filtration function f^b such that the sequence of topological maps in (f, f^b)-persistence is the same as the ones in FB-persistence with respect to f. Furthermore, Algorithm 3 computes f^b in with a runtime of $O(|V| + |G|)$.*

Proof. To show that Algorithm 3 correctly computes f^b, it suffices for us to show that $\text{IC}_i(G; f^b)$ is exactly $\text{IC}_{n-i}(G; f)$. Indeed, the intermediate complexes produced for the pair (G, f) are "upward closed" in the sense that the edges of $\text{IC}_i(G; f)$ all have the same value a_i, but the value of vertices are in general only $\leq a_i$.

If we want $\text{IC}_n(G; f)$ to be the first intermediate complex that appears in the filtration f^b, we would want to relabel all of its vertices to the maximal value a_n, which is precisely what the algorithm does. Since f is vertex-based, we only need to do this on vertices and we can modify the edges later.

Algorithm 3 Computing the Backward Filtration Function

Input $\to$ **Output:** The graph G and filtration function $f \to$ The function f^b

$f^b_{\text{vertex}} \leftarrow \{v : -\infty \mid \text{for each } v \in V(G)\}$ ▷ Initialize f^b on vertices with values in $-\infty$.
for edge $e = (v, w)$ in $E(G)$ **do**
  $f^b_{\text{vertex}}[v] \leftarrow \max(f(e), f^b_{\text{vertex}}[v])$.
for vertex v in $V(G)$ **do**
  if $f^b_{\text{vertex}}[v] = -\infty$ **then**
    $f^b_{\text{vertex}}[v] \leftarrow f(v)$ ▷ Mark isolated vertices to their value under f.
$(f^b_{\text{vertex}}, f^b_{\text{edge}}) \leftarrow (\text{map}(x \mapsto -x; f^b_{\text{vertex}}), \text{map}(x \mapsto -x; f_{\text{edge}}))$.
return $(f^b_{\text{vertex}}, f^b_{\text{edge}})$.

If we want $\text{IC}_{n-1}(G; f)$ to be the second intermediate complex that appears in the filtration f^b, we would want all vertices that have not been marked a_n already to be marked a_{n-1}. This amounts to a maximality comparison in the for loop in the algorithm.

If we want $\text{IC}_{n-2}(G; f)$ to be the third that appears, we similarly want to label all vertices not marked a_n, a_{n-1} yet to be marked as such, so repeating this process yields the correctness of the algorithm.

The algorithm has runtime $O(|V| + |G|)$ since it only requires looping through the vertex set and the edge set linearly. □

Now we move on to Proposition 4. In fact, Algorithm 3 before is a special case of the algorithms for this. In order to create a backward-filtration function that contracts in a permutation of the intended order specified by a permutation σ, we observe that Algorithm 3 would actually hold if the $\max$ function is operated with respect to an "ordering given by σ" as opposed to the natural ordering of the reals.

Instead of changing the ordering system of the reals in our algorithm, we will instead change the filtration function f inside the parameter of the $\max$ function to this ordering. To do this, we define the following variant of f:

Definition 12. *Let $f : G \to \mathbb{R}$ be a filtration function with filtration values $a_1 < a_2 < ... < a_n$. Let σ be a permutation of the list $\{1, ..., n\}$, we define $\sigma \cdot f$ as the function*

$$\sigma \cdot f : G \to \mathbb{R}, \quad \sigma \cdot f(x) = a_{\sigma(i)} \text{ if } f(x) = a_i,$$

where x is either a vertex or an edge.

Algorithm 4 Computing the τ-Backward Filtration Function

Input $\to$ **Output:** The graph G, function f, permutation $\tau^{-1} \to$ The function f^τ

$g \leftarrow (\text{re} \circ \tau^{-1}) \cdot f$
$f^\tau_{\text{vertex}} \leftarrow \{v : -\infty \mid \text{for each } v \in V(G)\}$ ▷ Initialize f^τ on vertices with values in $-\infty$.
for edge $e = (v, w)$ in $E(G)$ **do**
  $f^\tau_{\text{vertex}}[v] \leftarrow \max(g(e), f^\tau_{\text{vertex}}[v])$.
for vertex v in $V(G)$ **do**
  if $f^\tau_{\text{vertex}}[v] = -\infty$ **then**
    $f^\tau_{\text{vertex}}[v] \leftarrow g(v)$ ▷ Mark isolated vertices to their value under f.
$f^\tau_{\text{vertex}} \leftarrow \text{map}(x \mapsto -x; f^\tau_{\text{vertex}})$.
$f^\tau_{\text{edge}} \leftarrow \text{map}(x \mapsto -x; g_{\text{edge}})$. ▷ g_{edge} is the function g on $E(G)$.
return $(f^\tau_{\text{vertex}}, f^\tau_{\text{edge}})$.

In practice, a permutation σ can be realized as a dictionary with entry being i and output being $\sigma(i)$. We can use this to make an associated dictionary whose entry is a_i and output is $a_{\sigma(i)}$. Depending on how the filtration function f is represented as a data structure, this may require a sorting of the

list of filtration values if the values are not sorted already. We can then produce $\sigma \cdot f$ by a linear scan through the entries of f and swap out its output using the associated dictionary.

A slight modification of Algorithm 3 now produces Algorithm 4. In the special case when $\tau = \mathrm{re}$ (the reverse list permutation), this will recover Algorithm 3. This gives f_2 as requested in Proposition 4, but note that plugging σ into τ also gives the desired f_1 in the proposition. A slight modification of the proof for Algorithm 3 will ensure the correctness of the algorithm here. Because we possibly may need to sort a list, this would incur a worst-case runtime of $O(n \log n)$, where $n = |V(G)| + |E(G)|$.

C PROOFS FOR SECTION 5

Proof of Proposition 7. Let Y be a simplicial complex, and let $Y_\bullet$ be a sequence of inclusions and contractions in some arbitrary order, of the form:

$$\emptyset = Y_{-1} \to Y_0 \to Y_1 \to ... \to Y_m \to Y_{m+1} = *.$$

Using the simplicial quotient method, we adjoin a disjoint base point v_+ to Y and also form a sequence of inclusions $Z_\bullet$ of the form

$$v_+ \to Z_0 \to Z_1 \to ... \to Z_m \to Z_{m+1}.$$

Taking $H_k(-)$ gives the sequence

$$H_k(Z_\bullet) : 0 \to H_k(Z_0) \to H_k(Z_1) \to ... \to H_k(Z_m) \to H_k(Z_{m+1}).$$

Fix $k > 0$, we also let $H_k(Y_\bullet)$ be the k-th persistent module for this from Definition 2.

Now for each Z_i, let C_i be the closed star of the vertex v_+ (ie. the union of all simplices containing v_+ and their faces). Observe that Y_i is isomorphic to the quotient Z_i/C_i. We also note that C_i is actually contractible - indeed, in the standard geometric realization of Z_i, there is an explicit straight line homotopy based on line segments from the simplicial link of v_+ to v_+ from construction, which shows that C_i's are contractible.

It is a general fact that for a simplicial complex K and contractible subcomplex K', the quotient map $K \to K/K'$ is a homotopy equivalence (see Proposition 0.17 of Hatcher (2002)).

$$\begin{array}{ccccccccccc}
0 & \longrightarrow & H_k(Z_0) & \longrightarrow & H_k(Z_1) & \longrightarrow & ... & \longrightarrow & H_k(Z_m) & \longrightarrow & H_k(Z_{m+1}) = 0 \\
 & & \cong\downarrow & & \cong\downarrow & & \cong\downarrow & & \cong\downarrow & & =\downarrow \\
0 & \longrightarrow & H_k(Z_0/C_0) & \longrightarrow & H_k(Z_1/C_1) & \longrightarrow & ... & \longrightarrow & H_k(Z_m/C_m) & \longrightarrow & H_k(\{*\}) = 0 \\
 & & \cong\downarrow & & \cong\downarrow & & \cong\downarrow & & \cong\downarrow & & =\downarrow \\
0 & \longrightarrow & H_k(Y_0) & \longrightarrow & H_k(Y_1) & \longrightarrow & ... & \longrightarrow & H_k(Y_m) & \longrightarrow & H_k(\{*\}) = 0
\end{array}$$

Here $H_k(Z_{m+1}) = 0$ because $C_{m+1} = Z_{m+1}$ in this case. Thus, we have an isomorphism of persistence modules between $H_k(Z_\bullet)$ and $H_k(Y_\bullet)$, so they have the same persistence diagrams.

To see what goes wrong in the case $k = 0$, we observe that the placement of the node v_+ can affect the birth / death time of vertices since v_+ is now the oldest node, so the birth times of the vertices would shift. This can be remedied, for FB-persistence or (σ, τ)-persistence, where we instead place v_+ to spawn after filtration finishes and before contraction begins. However, this does not work for hourglass persistence, since we can contract before all the filtration finishes.

Thus, we would like to consider a different method, as described in the proposition. For a simplicial complex K, we write $(K)^1$ to denote the 1-skeleton of the simplicial complex K (ie. the union of all vertices and edges). On the level of $k = 0$, observe that the inclusion of the 1-skeleton of X induces a map of persistence modules of the form:

$$\begin{array}{ccccccccc}
0 & \longrightarrow & H_0((Y_0)^1) & \longrightarrow & H_0((Y_1)^1) & \longrightarrow & ... & \longrightarrow & H_0((Y_m)^1) & \longrightarrow & H_0(\{*\}) \\
 & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & =\downarrow \\
0 & \longrightarrow & H_0(Y_0) & \longrightarrow & H_0(Y_1) & \longrightarrow & ... & \longrightarrow & H_0(Y_m) & \longrightarrow & H_0(\{*\})
\end{array}.$$

Here each verticial arrow is an isomorphism because the inclusion map $(Y_i)^1 \to Y_i$ is a bijection on connected components. Thus, we have an isomorphism of persistence modules and hence the top and bottom row have the same persistence diagrams. What this means is that 0-th dimensional persistent homology for the simplicial complex is equivalent to the 0-th dimensional persistent homology for its restriction to the 1-skeleton. Thus, this reduces to the scenario in Lemma 1, and this concludes the proof. □

D PROOFS FOR SECTION 6

Proof of Theorem 2. We split the proof of stability into two parts - the case where $k > 0$ and the case where $k = 0$.

Let X be a simplicial complex and $H_k(X_\bullet, f, g)$ and $H_k(X_\bullet, f', g')$ be the two persistence modules in (f, g)-FB persistence and (f', g')-FB persistence respectively.

For $k > 0$, and when X is a simplicial complex, we use the simplicial quotient interpretation of the persistence modules here (in the sense of Section 5). By Proposition 7, we see that the persistence pairs associated to $H_k(X_\bullet, f, g)$ is the same as the persistence pairs associated to the persistence homology of a simplicial filtrations of a simplicial complex Z, which we will write the filtration function as h. Similarly, for $H_k(X_\bullet, f', g')$, we will get a (possibly) different simplicial filtration of the same simplicial complex Z, which we will write the filtration function as h'.

Thus, we see that $d_B(H_k(X_\bullet, f, g), H_K(X_\bullet, f', g') = d_B(H_k(Z_\bullet, h), H_k(Z_\bullet, h'))$ is the bottleneck distance of the persistence diagrams coming from two filtration functions h and h'. By the classic bottleneck stability of Cohen-Steiner et al. (2006), we have that

$$d_B(H_k(Z_\bullet, h), H_k(Z_\bullet, h')) \leq ||h - h'||_\infty.$$

Let us unwrap the construction of h and h' here. Indeed, recall Z is the simplicial cone of X. For (f, g), the function h agrees with f when restricted to the base of the simplicial cone (which is a copy of X). Every other simplex must contain the disjoint basepoint v_+, and is obtained by joining v_+ to a base simplex σ in X. We write all such simplices as (σ, v_+). In this case, $h(\sigma, v_+)$ is defined to be $\max(f)+g(\sigma)$. Finally, one also specifies that $h(v_+)$ to be born before all the other persistence values, say 0 for uniformity (this does not affect the persistence pairs of dimension higher than 0). There is a similar description for h', and hence we see that

$$\begin{aligned} ||h - h'||_\infty &= \max(||f - f'||_\infty, ||g - g' + (\max(f) - \max(f'))||_\infty) \\ &\leq ||f - f'||_\infty + ||g - g' + (\max(f) - \max(f'))||_\infty \\ &\leq ||f - f'||_\infty + ||g - g'||_\infty + |\max(f) - \max(f')|. \end{aligned}$$

Remark: Note that there is no coefficient 2 here. For brevity, in the main paper, we added a 2 because that is the bound we will get for $k = 0$.

When X is a (regular) cell complex, we evidently can still take the cone of a cell complex, which has cell decomposition according to the subcomplex of the base. The arguments above would still hold in this case. This is because the original Bottleneck stability Cohen-Steiner et al. (2006) was proven for all triangulable spaces and tame functions on them. Because of its finiteness and regularity constraint, the geometric realization for the regular cell complex can be taken to be from what is called an **o-minimal structure** (see Coste (2002) or van den Dries (1998) - note this is called an "o-minimal expansion of the real field" in the second reference). See also Section 1 of Moggach (2020) for a detailed description between cell complexes and o-minimal structures. Elements in such o-minimal structures are known to be triangulable by the so-called **Hardt's triangulation theorem** and the filtration is clearly still tame, and the bound above applies through.

The case of $k = 0$ is different because the persistence tuples differ. First of all, by the same argument in the proof of Proposition 7, we can reduce this to the case where X is a graph. We will still work in the simplicial quotient perspective. The reader might wonder - there are some graphs that are technically not simplicial complexes (ie. has self-loops or multiple edges), does the simplicial quotient also work for them? The answer is yes because for such graphs, one can always discretize further by labeling the mid-point of exceptional edges (ie. self-loops and multi-edges) as a vertex of the graph. This then becomes a simplicial complex and makes no difference in filtration since we will be spawning the midpoints at the same time as the edges, so we can ignore these trivial deaths.

Thus, we without loss reduce to the case where X is a simplicial graph, and we can try to apply a variant of the simplicial quotient method. Indeed, we observe that, - if we move the adjoined formal basepoint v_+ to be spawned after we have filtrated the entire graph X first using f and before we start the contraction using g, then the 0-th dimensional persistence diagram with the pair representing v_+ removed is the 0-th dimensional persistence diagram from the 0-th dimensional (f, g)-persistence diagram. Indeed, this is because, v_+, being the last vertex spawned, will die immediately when the first vertex in g appears and asks to be contracted (which will not change the

other outputs).

To go further, we would need to work explicitly with a proof strategy of bottleneck stability (ex. Skraba & Turner (2021), Schnider (2024), or Ji et al. (2025)) and adapt it to our scenario. Indeed, let Z be the simplicial cone of X, and h and h' be the filtrations corresponding to the pairs (f, g) and (f', g'). But also, because we are working with $k = 0$, we can again restrict to the 1-skeleton Z' of Z, which is now a graph.

For each simplex (either an edge or a vertex) $x \in Z'$, we consider a linear interpolation function $h_t(y) = (1-t)h(x) + th'(x)$ with t values in $[0, 1]$. We can divide the interval $[0, 1]$ into a finite collection of intervals $[t_0, t_1], [t_1, t_2], ..., [t_n, t_{n+1}]$ with $t_0 = 0$ and $t_{n+1} = 1$ such that, for every pair of simplices x and y and $t \in [t_i, t_{i+1}]$, we either have that $h_t(x) \geq h_t(y)$ for all t or $h_t(x) \leq h_t(y)$ for all t.

Since the relative ordering of simplices are not changed in this interval, the 0-dimensional persistence diagrams for h_{t_i} and $h_{t_{i+1}}$ are the same in **combinatorial time**. Now choose any bijection π of the **function time** persistence diagrams for $\mathrm{PH}_0^F(Z'; h_{t_i})$ and $\mathrm{PH}_0^F(Z'; h_{t_{i+1}})$ (by which we denote their forward time 0-diemnsional persistence diagrams) such that $\pi(b_t, d_t) = (b_{t+1}, d_{t+1})$ if and only if both tuples represent the same tuple in **combinatorial time**. Note that due to the placement of v_+, this will necessarily pair the two pairs associated to v_+ together (call them (b_+, d_+) and (b'_+, d'_+). Notably the restriction π_1 of π to the tuples excluding the one corresponding to v_+ is still a bijection.

For any two filtrations ϕ, ψ of Z' (the 1-skeleton of the cone of X). We write $d'_B(\mathrm{PH}_0^F(Z'; \phi), \mathrm{PH}_0^F(Z'; h_{t_{i+1}}))$ as the term

$$\inf_{\varphi \text{ bijections } \mathrm{PH}_0^F(Z';\phi)-(b_+,d_+)\to \mathrm{PH}_0^F(Z';\psi)-(b'_+,d'_+)} ||(b, d) - \varphi(b, d)||_\infty.$$

Observe that when $\phi = h$ and $\psi = h'$, one has that

$$d_B^{\mathrm{FB}}(H_0(X, f, g), H_0(X, f', g')) = d'_B(\mathrm{PH}_0^F(Z'; h), \mathrm{PH}_0^F(Z'; h')).$$

Now d'_B can fail the non-degeneracy condition for being a metric, but it will still have a triangle inequality!

Now each tuple in the persistence diagram here actually comes from a **vertex-edge pair** (v, e), where v makes birth to the tuple and e kills it. Choosing π_1 as the bijection here, we have

$$\begin{aligned} d'_B(\mathrm{PH}_0^F(Z'; h_{t_i}), \mathrm{PH}_0^F(Z'; h_{t_{i+1}})) &\leq ||(b, d) - \pi_1(b, d)|| \\ &\leq \max_{(v,e)\in X} ||(h_{t_i}(v), h_{t_i}(e)) - (h_{t_{i+1}}(v), h_{t_{i+1}}(e)||_\infty \\ &\qquad \text{Definition of } \pi_1 \text{ and associating pairs} \\ &\leq \max_{v\in X} ||h_{t_i}(v) - h_{t_{i+1}}(v)||_\infty + \max_{e\in X} ||h_{t_i}(e) - h_{t_{i+1}}(e)||_\infty \\ &\leq (t_{i+1} - t_i)||f - f'||_\infty + \max_{e\in X} ||h_{t_i}(e) - h_{t_{i+1}}(e)||_\infty \end{aligned}$$

For the second term, there are two possibilities for the edge e - either it comes from the filtration or it comes from the contraction. Thus, we have that

$$\begin{aligned} \max_{e\in X} ||h_{t_i}(e) - h_{t_{i+1}}(e)||_\infty \leq \max((t_{i+1} - t_i)||f - f'||_\infty, \\ (t_{i+1} - t_i)||(g - g') + (\max(f) - \max(f')||_\infty) \\ \leq (t_{i+1} - t_i)(||f - f'||_\infty + ||(g - g')||_\infty + |\max(f) - \max(f')|) \end{aligned}$$

Now decomposing $d'_B(\mathrm{PH}_0^F(Z'; h), \mathrm{PH}_0^F(Z'; h'))$ into the sum $\sum_{i=0}^m d'_B(\mathrm{PH}_0^F(Z'; h_{t_i}), \mathrm{PH}_0^F(Z'; h_{t_{i+1}}))$ and using the bound above, we have that

$$d_B^{\mathrm{FB}}(H_0(X, f, g), H_0(X, f', g')) \leq 2||f - f'||_\infty + ||(g - g')||_\infty + |\max(f) - \max(f')|.$$

□

E More Discussions on Comparison with Other Methods

E.1 Zigzag Filtration

In the definition of (f,g)-FB-persistence, the filtration function g is specifying the order of subgraphs being contracted. This is different from the use case of zigzag filtration Carlsson & de Silva (2010), where they are looking at a sequence of insertions and removals of a graph, but the removal process is fundamentally not continuous, so their persistence diagrams do not follow a linear time (whereas our set-up does).

To give an example, consider G the path graph on 2 vertices, H1 the emptyset, H2 a subgraph of G which is discrete with 2 vertices. Then the following is a diagram that can occur in zigzag filtration

$$H_1 \subset G \supset H_2$$

Observe that there is no map from G to H_2 that is continuous, since we cannot split G into 2 components. In general, the arrows in zigzag filtration go in different directions, and they are all inclusion maps. On the other hand, the class of diagrams we always consider in (f,g)-FB persistence is a sequence of continuous maps

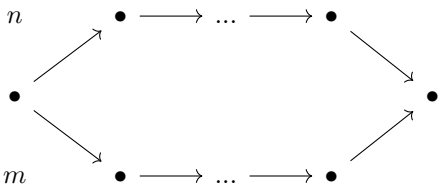
$$G_1 \to G_2 \to G_3 \to ... \to G_n$$

which are inclusions followed by contractions. Here the arrows all go in the same direction, but they can be either inclusions or contractions (this is even more apparent in the setting of hourglass persistence).

More technically, zigzag filtration is considering quiver spaces on the path graph P_n with any possible orientations but with inclusions, whereas we are considering quiver spaces on P_n oriented in a uniform direction with both inclusions and contractions. Generally, we expect zigzag persistence and (f, g)-FB persistence to be incomparable, and their methods can complement each other.

E.2 Bipath Filtration

In Aoki et al. (2025), the authors introduced bipath persistence built on bipath filtrations, which are quiver spaces on the Hasse quiver $B_{n,m}$, which is of the form



Similar to the case of zigzag filtration, the filtrations that appear in bipath filtration on Page 1 of Aoki et al. (2025) is again using inclusions only, whereas our settings consider the usage of contractions along with inclusions. Furthermore, bipath filtration considers quiver spaces on the Hasse quiver $B_{n,m}$ with inclusions, but we study quiver spaces on P_n oriented in a uniform direction with both inclusions and contractions. Therefore, we expect that the two methods in general do not subsume one another but are complementary ideas.

E.3 Efficiency Comparison

For persistence diagrams (PDs) of dimension > 0 on simplicial complexes, hourglass persistence itself can be computed using the "simplicial quotient" trick mentioned in Section 5 of the paper using Proposition 7. Proposition 4 reduces the question to computing inclusion-based PH on a simplicial complex, which has a standard runtime dominated by the runtime to perform matrix reduction algorithms (see Otter et al. (2017)). For PDs of dimension 0, this can be reduced to looking at PDs of just underlying 1-skeleton (ie. a graph). In this case, there is a way to optimize the calculation with union-find with a runtime dominated by $O(n \log n + m)$, where n is the number

of edges and m is the number of vertices (see Horn et al. (2021) for more details). Although Horn et al. (2021) only discuss the runtime in the filtration steps, tracking vertex representatives in the contraction steps would also have the same runtime.

When focusing on FB-persistence and $(f, g)-$FB-persistence, though, we expect the algorithms presented in Section 7 (which are not using the "simplicial quotient" trick) to be faster empirically. In general, the ability to reduce the size of object using contractions (ie. using the cellular complex extension in Section 5 instead) would lead to a decrease in the memory complexity. We therefore expect the cellular-based methods of hourglass persistence, $(f, g)-$FB persistence, etc. to have more efficient algorithms in the future.

E.4 Scaling on Large Graphs

Interleaving the filtration and contraction steps avoids the need to filtrate the entire graph before starting contractions, which allows the total size of the graph that appears in its entire lifetime to be bounded.

This can improve over the runtime on general simplicial complexes. Let K be a simplicial complex with n total simplicies. To give a heuristic/informal estimation from a practitioner perspective - the typical runtime of inclusion based PH on K scales approximately $O(n^3)$ as it requires a matrix reduction algorithm.

Suppose we bound the threshold to d simplicies, and a practitioner equipped with the contraction framework can implement a specific instance of hourglass persistence as follows: as soon as we reach more than d simplicies in a given step, we contract everything to a point, and so on. Then we believe the runtime should roughly to be $O(\frac{n}{d} \cdot (d)^3)$.

For practical purposes, it may also be beneficial to not contract everything to a point when the threshold is exceeded. If the contractions are done in multiple stages, then we believe a similar runtime analysis holds. We are excited at the possibilities that this framework can allow our community to address the problem of PH scaling on large graphs/simplicial complexes.

F The Use of Large Language Models

We used large language models to aid in the writing process of the introduction, to help check grammar / suggest edits for the main paper, and to double-check some algorithmic discussions arising from the paper.