Computing Zigzag Vineyard Efficiently Including Expansions and Contractions

Tamal K. Dey
Department of Computer Science, Purdue University, West Lafayette, IN, USA

Tao Hou
School of Computing, DePaul University, Chicago, IL, USA

Abstract

Vines and vineyard connecting a stack of persistence diagrams have been introduced in the non-zigzag setting by Cohen-Steiner et al. [4]. We consider computing these vines over changing filtrations for zigzag persistence while incorporating two more operations: expansions and contractions in addition to the transpositions considered in the non-zigzag setting. Although expansions and contractions can be implemented in quadratic time in the non-zigzag case by utilizing the linear-time transpositions, it is not obvious how they can be carried out under the zigzag framework with the same complexity. While transpositions alone can be easily conducted in linear time using the recent FastZigzag algorithm [5], expansions and contractions pose difficulty in breaking the barrier of cubic complexity [6]. Our main result is that, the half-way constructed up-down filtration in the FastZigzag algorithm indeed can be used to achieve linear time complexity for transpositions and quadratic time complexity for expansions and contractions, matching the time complexity of all corresponding operations in the non-zigzag case.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Algebraic topology

Keywords and phrases zigzag persistence, vines and vineyard, update operations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2024.49

Related Version *Full Version*: `https://arxiv.org/pdf/2307.07462.pdf`

Funding This research is partially supported by NSF grants CCF 2049010 and 2301360.

1 Introduction

Computation of the persistence diagram (PD) also called the barcode from a given filtration has turned out to be a central task in topological data analysis (TDA). Such a filtration usually represents a nested sequence of sublevel sets of a function. In scenarios where the function changes, the filtration and hence the PD may also change. The authors in [4] provided an efficient algorithm for updating the PD over an atomic operation which *transposes* two consecutive simplex additions in the filtration. Using this atomic operation repeatedly, one can connect a series of filtrations derived from a time-varying function and obtain a stack of PD's called the *vineyard*. The authors [4] show that updating the PD due to the atomic transposition can be computed in $O(m)$ time if m simplices constitute the filtration. Recently, zigzag persistence [1, 2, 5, 7, 12, 13, 14] has drawn considerable attention in the community of topological data analysis. In this paper, we explore efficient updates of barcodes over such atomic operations for *zigzag filtrations* and include two additional types of operations [6] which are necessary for converting a zigzag filtration to any other: *expansions* that enlarge the filtration and *contractions* that shrink the filtration. Motivating examples are given in Appendix A of [6] and Section 3.1 of [16] for supporting these operations efficiently.

Although the work [4] introducing vineyard does not consider expanding and contracting a filtration, the two operations can be naturally defined in the non-zigzag setting. An expansion would involve inserting a simplex addition at a certain location in the non-zigzag filtration and a contraction would be its reverse. For implementing an expansion, one can attach a new simplex addition to the end of the current non-zigzag filtration and then bring it to the intended position by a series of transpositions [4]. For a contraction, one does the reverse. The cost of the update is then $O(m^2)$ because: (i) attaching a simplex addition to the filtration involves a reduction which takes $O(m^2)$ time; (ii) bringing it to correct position needs $O(m)$ transpositions each consuming $O(m)$ time. The main finding of this paper is that, although zigzag filtrations introduce more complications to the barcode computation, updating zigzag barcodes can be done with the *same* complexity as for the non-zigzag case over all the operations.

As recognized in [6], eight atomic operations are necessary for any zigzag filtration to transform to any other, including four implementing transpositions [1, 2, 7, 13, 14, 15] (henceforth called *switches*) and another four implementing expansions [13, 14] and contractions. An immediate temptation would be to handle the updates by maintaining a non-zigzag filtration obtained from straightening the zigzag filtration using a recently proposed algorithm called FastZigzag [5]. Although we can easily implement the switches in linear time using this conversion [5] and the algorithm in [4], implementing the expansions and contractions in quadratic time becomes difficult. In fact, in a previous arXiv paper [6], we adopt this approach but cannot implement all operations due to an adjacency change observed for certain operations.

To see the challenge posed by the adjacency change, recall that the conversion to non-zigzag in [5] first goes through an (implicit) conversion to an *up-down* filtration [2], which is then converted to a non-zigzag one. Since the first part of the up-down filtration is the same as the first part of the final non-zigzag filtration, the adjacency change can be observed from the up-down filtration. In the constructed up-down filtration, the first part consists of all added simplices (with the same order) in the original zigzag filtration $\mathcal{F}$. Consequently, the first part may contain multiple additions of the same simplex σ without any deletion in between [5]. These multiple additions of σ are distinguished by Δ-*cells* [9, 11] which are copies of σ sharing boundaries *non-trivially* as in Figure 3. Also, one needs to consider two different types of expansions and contractions, where an expansion inserts two inclusion arrows involving the same simplex σ in a zigzag filtration, and a contraction as a reverse operation removes them (see Section 2.1 for formal definitions). Since the two inclusion arrows of σ being inserted or removed could either point toward or away from each other, we have *inward* and *outward* expansions along with the corresponding *inward* and *outward* contractions.

The outward expansions and contractions turn out to be harder than their inward counterparts. To see this, consider an outward expansion shown in Figure 1, where simplices added in the original zigzag filtrations and corresponding Δ-cells added in the up-down filtrations are connected by dotted blue lines. Moreover, in the figure, we connect certain Δ-cells (resp. simplices) and their faces by blue arrows. Observe that before the expansion, the Δ-cells τ_0, τ_0' have the Δ-cell σ_1 in their boundaries. However, after the expansion, the boundary cell of τ_0, τ_0' changes to σ_2, which is the Δ-cell corresponding to the newly inserted addition of σ (in red). Assuming working on non-zigzag filtrations, such a change would mean that one Δ-cell in the boundary column of τ_0 or τ_0' in the "persistence boundary matrix" [8] changes into a *much later* Δ-cell. Notice that there could be $O(m)$ such changes. In [4], a transposition swaps only two *consecutive* columns and rows in the persistence boundary matrix [8] without any change on the simplices' boundaries. The simplicity and locality of the transposition [4] make the linear algorithm possible. However, for the outward expansions and contractions, it is not at all clear that the change in adjacency arising in our case can

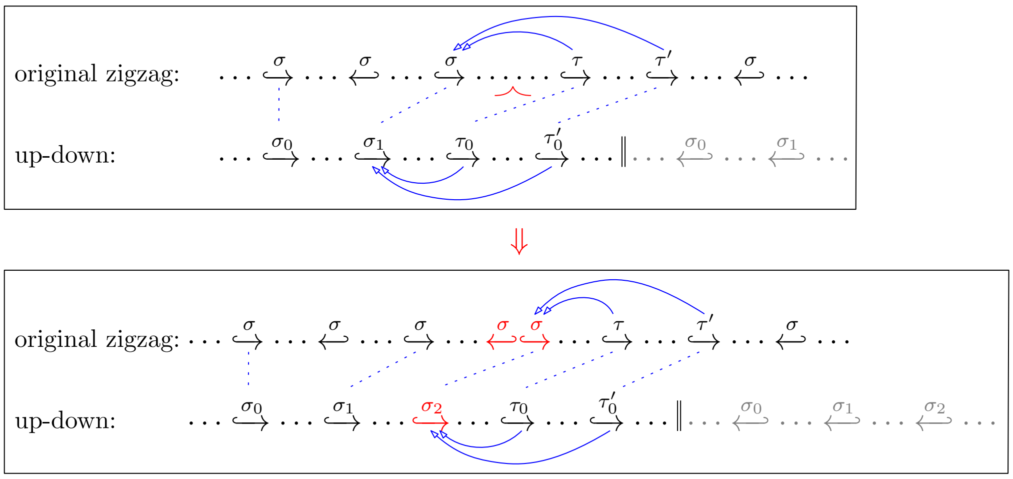

Figure 1 (upper box) Δ-cells τ_0 and τ'_0 corresponding to simplices τ and τ' in the original filtration have Δ-cell σ_1 (corresponding to an addition of σ) in their boundaries in the up-down filtration; (lower box) Δ-cells τ_0 and τ'_0 now have Δ-cell σ_2 (corresponding to the newly inserted σ) in their boundaries once an outward expansion with σ takes place.

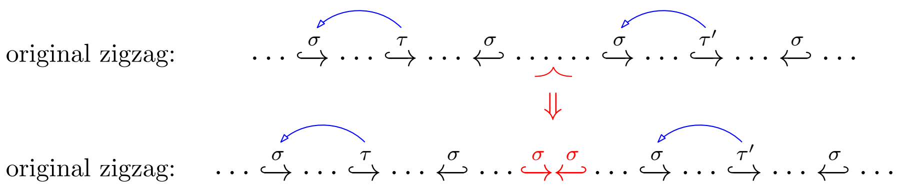

Figure 2 Each copy of simplex added in the original zigzag filtrations has the same copies of simplices in its boundary before and after an inward expansion. So the boundary Δ-cells in the up-down filtrations also stay the same.

be handled by a series of these transpositions [4]. A straightforward approach would be to perform the $O(m^2)$ reductions [8] on each changed column in the persistence boundary matrix. But then one may incur a cubic complexity to take care of the possible $O(m)$ such changes. Notice that similar difficulty arises for outward contraction but not for inward expansion or contraction (see Figure 2).

In this paper, we find a way to tame down the curse of adjacency change by adopting primarily two ideas. First, to reduce the effect of the adjacency change in the outward expansions and contractions, we choose to work on up-down filtrations [2] instead. Up-down filtrations eliminate the further complication on the adjacency introduced by the "coning" [3] of Δ-cells which is the final step of the conversion in [5]. Second, we introduce a collapsing strategy for identifying boundaries which allows us to nullify the effect of adjacency changes on the time complexity. To explain intuitively the strategy, consider an outward contraction, where the adjacency change can be briefly stated as two p-cells σ_1, σ_2 being identified as the same cell. Such a change can be enacted by introducing an extra $(p+1)$-cell χ whose boundary equals $\sigma_1 + \sigma_2$ and then collapsing χ (see Figure 5). Section 4 details the adjacency change and the algorithm. Another key advantage of working on up-down filtrations is that updates on up-down filtrations only need to maintain at most three chains per bar (see Remark 5), which is much less than the $O(m)$ chains needed if working directly on the original zigzag filtrations [6].

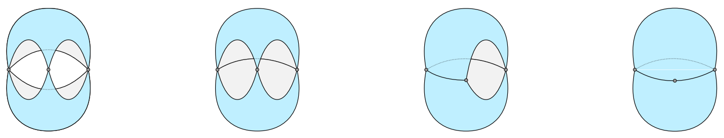

Figure 3 Examples of Δ-complexes with two triangles *having the same set of vertices* sharing 0, 1, 2, or 3 edges on their boundaries [5].

Working on up-down filtrations, however, makes inward contraction non-trivial. One may wonder the following: since inward contraction is the reverse of inward expansion which already has an update algorithm [13, 14], can we not simply "reverse" this algorithm to get an algorithm for inward contraction? We notice that this is usually not the case. In fact, we discover an interval mapping for inward contraction different from that for inward expansion presented in [13, 14], namely, the persistence pairs are alternatively re-linked. See Section 6.

2 Preliminaries

Since we utilize (partially) the conversion in [5], we work on Δ-*complexes* [9, 11] instead of simplicial complexes in this paper. Building blocks of Δ-complexes are called *cells* or Δ-*cells*, which are similar to simplices but could have common faces in more relaxed ways (see Figure 3 and also Definition 1 of [5] for a precise definition). A *zigzag filtration* (or simply *filtration*) is a sequence of Δ-complexes

$$\mathcal{F} : K_0 \leftrightarrow K_1 \leftrightarrow \cdots \leftrightarrow K_m, \tag{1}$$

in which each $K_i \leftrightarrow K_{i+1}$ is either a forward inclusion $K_i \hookrightarrow K_{i+1}$ (an addition of several cells) or a backward inclusion $K_i \hookleftarrow K_{i+1}$ (a deletion of several cells). Taking the p-th homology H_p, we derive a *zigzag module*

$$\mathsf{H}_p(\mathcal{F}) : \mathsf{H}_p(K_0) \leftrightarrow \mathsf{H}_p(K_1) \leftrightarrow \cdots \leftrightarrow \mathsf{H}_p(K_m).$$

In $\mathsf{H}_p(\mathcal{F})$, each $\mathsf{H}_p(K_i) \leftrightarrow \mathsf{H}_p(K_{i+1})$ is a linear map induced by inclusion. In this paper, we take the coefficient $\mathbb{Z}_2$ for H_p and hence chains or cycles can be treated as sets of cells. The zigzag module $\mathsf{H}_p(\mathcal{F})$ has a decomposition [1, 10] of the form $\mathsf{H}_p(\mathcal{F}) \simeq \bigoplus_{k\in\Lambda} \mathcal{I}^{[b_k,d_k]}$, in which each $\mathcal{I}^{[b_k,d_k]}$ is an *interval module* over the interval $[b_k, d_k] \subseteq \{0, 1, \ldots, m\}$. The (multi-)set of intervals $\mathsf{Pers}_p(\mathcal{F}) := \{[b_k, d_k] \mid k \in \Lambda\}$ is an invariant of $\mathsf{H}_p(\mathcal{F})$ and is called the p-th *zigzag barcode* (or simply *barcode*) of $\mathcal{F}$. Each interval in $\mathsf{Pers}_p(\mathcal{F})$ is called a p-th *persistence interval*. We usually consider the homology H_* in all dimensions and take the zigzag module $\mathsf{H}_*(\mathcal{F})$, for which we have $\mathsf{Pers}_*(\mathcal{F}) = \bigsqcup_{p\geq 0} \mathsf{Pers}_p(\mathcal{F})$. In this paper, sometimes a filtration may have nonconsecutive indices on the complexes (i.e., some indices are skipped); notice that the barcode is still well-defined.

An inclusion in a filtration is called *cell-wise* if it is an addition or deletion of a single cell σ, which we sometimes denote as, e.g., $K_i \xleftrightarrow{\sigma} K_{i+1}$. A filtration is called *cell-wise* if it contains only cell-wise inclusions. For computational purposes, filtrations in this paper are by default cell-wise filtrations starting and ending with *empty* complexes (as adopted in [5, 13]); notice that any filtration can be converted into this standard form by expanding the inclusions and attaching complexes to both ends.

2.1 Update operations

We now present all the update operations. In this subsection, $\mathcal{F}$ and $\mathcal{F}'$ are both simplex-wise zigzag filtrations consisting of simplicial complexes which start and end with empty complexes.

Forward switch [13] swaps two forward inclusions and requires $\sigma \nsubseteq \tau$:

$$\begin{tikzcd}[row sep=small]
\mathcal{F}: K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-1} \arrow[r, hook, "\sigma"] & K_i \arrow[r, hook, "\tau"] & K_{i+1} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m \arrow[d, dashed, bend left] \\
\mathcal{F}': K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-1} \arrow[r, hook, "\tau"] & K'_i \arrow[r, hook, "\sigma"] & K_{i+1} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m
\end{tikzcd} \tag{2}$$

Notice that if $\sigma \subseteq \tau$, then adding τ to K_{i-1} in $\mathcal{F}'$ does not produce a simplicial complex.

Backward switch swaps two backward inclusions and requires $\tau \nsubseteq \sigma$:

$$\begin{tikzcd}[row sep=small]
\mathcal{F}: K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-1} & K_i \arrow[l, hook', "\sigma"'] & K_{i+1} \arrow[l, hook', "\tau"'] \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m \arrow[d, dashed, bend left] \\
\mathcal{F}': K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-1} & K'_i \arrow[l, hook', "\tau"'] & K_{i+1} \arrow[l, hook', "\sigma"'] \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m
\end{tikzcd} \tag{3}$$

Outward/inward switch [1] swaps two inclusions of opposite directions and requires $\sigma \neq \tau$:

$$\begin{tikzcd}[row sep=small]
\mathcal{F}: K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-1} \arrow[r, hook, "\sigma"] & K_i & K_{i+1} \arrow[l, hook', "\tau"'] \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m \arrow[d, dashed, bend left, leftrightarrow] \\
\mathcal{F}': K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-1} & K'_i \arrow[l, hook', "\tau"'] \arrow[r, hook, "\sigma"] & K_{i+1} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m
\end{tikzcd} \tag{4}$$

The switch from $\mathcal{F}$ to $\mathcal{F}'$ is an outward switch and the switch from $\mathcal{F}'$ to $\mathcal{F}$ is an inward switch. Notice that if $\sigma = \tau$, then e.g., for outward switch, we cannot delete τ from K_{i-1} in $\mathcal{F}'$ because $\tau \notin K_{i-1}$.

Inward contraction/expansion [13] is as follows:

$$\begin{tikzcd}[row sep=small]
\mathcal{F}: K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-2} \arrow[r, leftrightarrow] & K_{i-1} \arrow[r, hook, "\sigma"] & K_i & K_{i+1} \arrow[l, hook', "\sigma"'] \arrow[r, leftrightarrow] & K_{i+2} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m \arrow[d, dashed, bend left, leftrightarrow] \\
 & & \mathcal{F}': K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-2} \arrow[r, leftrightarrow] & K'_i \arrow[r, leftrightarrow] & K_{i+2} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m
\end{tikzcd} \tag{5}$$

From $\mathcal{F}$ to $\mathcal{F}'$ we have an inward contraction and from $\mathcal{F}'$ to $\mathcal{F}$ we have an inward expansion. To clearly show the correspondence of complexes in $\mathcal{F}$ and $\mathcal{F}'$, indices for $\mathcal{F}'$ are made non-consecutive in which $i-1$ and $i+1$ are skipped. We also have $K_{i-1} = K'_i = K_{i+1}$.

Outward contraction/expansion is similar to the inward version with the difference that the two center arrows now pointing away from each other:

$$\begin{tikzcd}[row sep=small]
\mathcal{F}: K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-2} \arrow[r, leftrightarrow] & K_{i-1} & K_i \arrow[l, hook', "\sigma"'] \arrow[r, hook, "\sigma"] & K_{i+1} \arrow[r, leftrightarrow] & K_{i+2} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m \arrow[d, dashed, bend left, leftrightarrow] \\
 & & \mathcal{F}': K_0 \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_{i-2} \arrow[r, leftrightarrow] & K'_i \arrow[r, leftrightarrow] & K_{i+2} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & K_m
\end{tikzcd} \tag{6}$$

Notice that we also have $K_{i-1} = K'_i = K_{i+1}$.

3 Up-down conversion and easier updates

In this section, we first briefly overview the conversion to up-down filtrations in [5] and provide some definitions for up-down filtrations. We then describe how the conversion facilitates the updates for certain operations based on existing algorithms [7, 13]. For operations whose updates are more involved, we provide the algorithms in Section 4–6.

3.1 Conversion to up-down filtrations

Given a *simplex*-wise zigzag filtration

$$\mathcal{F} : \varnothing = K_0 \xleftrightarrow{\sigma_0} K_1 \xleftrightarrow{\sigma_1} \cdots \xleftrightarrow{\sigma_{m-1}} K_m = \varnothing \tag{7}$$

consisting of simplicial complexes, we convert $\mathcal{F}$ into the following *cell*-wise *up-down* filtration consisting of Δ-complexes [5]:

$$\mathcal{U} : \varnothing = L_0 \xhookrightarrow{\xi_0} L_1 \xhookrightarrow{\xi_1} \cdots \xhookrightarrow{\xi_{n-1}} L_n \xhookleftarrow{\xi_n} L_{n+1} \xhookleftarrow{\xi_{n+1}} \cdots \xhookleftarrow{\xi_{m-1}} L_m = \varnothing.$$

Notice that an up-down filtration is a special type of zigzag filtration where the first half consists of only additions and the second half consists of only deletions [2]. In $\mathcal{U}$, cells $\xi_0, \xi_1, \ldots, \xi_{n-1}$ are distinct copies of simplices added in $\mathcal{F}$ with the addition order preserved (notice that the same simplex could be added more than once in $\mathcal{F}$). Symmetrically, cells $\xi_n, \xi_{n+1}, \ldots, \xi_{m-1}$ are distinct copies of simplices deleted in $\mathcal{F}$, with the order also preserved. To build $\mathcal{U}$ from $\mathcal{F}$, one only needs to list all the additions first and then the deletions in $\mathcal{F}$, following their orders in $\mathcal{F}$, with each addition of a simplex in $\mathcal{F}$ corresponding to a distinct Δ-cell in $\mathcal{U}$ [5]. We have that the conversion from $\mathcal{F}$ to $\mathcal{U}$ can be done in $O(m)$ time [5]. We also have $m = 2n$ because each added simplex must be eventually deleted in $\mathcal{F}$. Figure 4 provides an example of the conversion, where an edge is added twice in $\mathcal{F}$ with σ_1 corresponding to its first addition and σ_2 corresponding to its second addition. In the up-down filtration $\mathcal{U}$, σ_1 and σ_2 appear in the same complex forming parallel edges (also called multi-edges).

We then briefly describe how the intervals in $\mathsf{Pers}_*(\mathcal{F})$ and $\mathsf{Pers}_*(\mathcal{U})$ correspond. First notice that each interval in $\mathsf{Pers}_*(\mathcal{F})$ or $\mathsf{Pers}_*(\mathcal{U})$ can be considered as "generated" by a pair of simplex-wise or cell-wise inclusions (i.e., simplex/cell additions or deletions). For example, an interval $[b, d] \in \mathsf{Pers}_*(\mathcal{F})$ is generated by the inclusion $K_{b-1} \leftrightarrow K_b$ on the left and the inclusion $K_d \leftrightarrow K_{d+1}$ on the right. As indicated, there is a natural bijection ϕ from the cell-wise inclusions in $\mathcal{U}$ to the simplex-wise inclusions in $\mathcal{F}$. For ease of presentation, we assign each simplex-wise inclusion $K_i \xleftrightarrow{\sigma_i} K_{i+1}$ in $\mathcal{F}$ the unique *index* i. We also similarly index cell-wise inclusions in $\mathcal{U}$. We then let the domain and codomain of ϕ be the sets of indices for the simplex-wise or cell-wise inclusions. We have the following fact from [5]:

▶ **Theorem 1.** *There is a bijection from $\mathsf{Pers}_*(\mathcal{U})$ to $\mathsf{Pers}_*(\mathcal{F})$ s.t. corresponding intervals are generated by corresponding pairs of inclusions. Specifically, suppose that an interval in $\mathsf{Pers}_*(\mathcal{U})$ is generated by two cell-wise inclusions indexed at l and r ($l < r$). Then its corresponding interval in $\mathsf{Pers}_*(\mathcal{F})$ is generated by two simplex-wise inclusions indexed at $\phi(l)$ and $\phi(r)$ (notice that $\phi(r)$ may be less than $\phi(l)$). Therefore, in order to map an interval in $\mathsf{Pers}_*(\mathcal{U})$ back to the interval in $\mathsf{Pers}_*(\mathcal{F})$, one only needs to look up the map ϕ which takes constant time.*

3.2 Representatives for up-down filtrations

Consider an up-down filtration $\mathcal{U}$ as built by the conversion in Section 3.1. Since each cell in $\mathcal{U}$ is added exactly once, we can uniquely denote each addition (resp. deletion) of a cell ξ in $\mathcal{U}$ as $\searrow\xi$ (resp. $\nwarrow\xi$). We also use $\nwsearrow\xi$ to denote either the addition or deletion of ξ. As mentioned, each interval $[b, d] \in \mathsf{Pers}_*(\mathcal{U})$ is generated by a pair $(\nwsearrow\xi, \nwsearrow\xi')$, where the locations of $\nwsearrow\xi$ and $\nwsearrow\xi'$ are as follows:

$$\mathcal{U} : \cdots \leftrightarrow L_{b-1} \xleftrightarrow{\xi} L_b \leftrightarrow \cdots \leftrightarrow L_d \xleftrightarrow{\xi'} L_{d+1} \leftrightarrow \cdots$$

In such a pair, we call $\nwsearrow\xi$ *positive* and $\nwsearrow\xi'$ *negative*. From now on, for an $\mathcal{U}$, we interchangeably consider $\mathsf{Pers}_*(\mathcal{U})$ as all pairs of additions and deletions generating the intervals. We also call a pair from $\mathsf{Pers}_p(\mathcal{U})$ a *p-pair*.

▶ **Definition 2** (Creators of chains). *Let $\searrow\xi$ be an addition in $\mathcal{U}$. A chain A is said to be* ***created by*** *$\searrow\xi$ if $\xi \in A$ and all other cells in A are added before ξ in $\mathcal{U}$. Moreover, Let $\nwarrow\xi$ be a deletion in $\mathcal{U}$. A chain A is said to be* ***created by*** *$\nwarrow\xi$ if $\xi \in A$ and all other cells in A are deleted after ξ in $\mathcal{U}$. We also say that $\searrow\xi$ or $\nwarrow\xi$ is the* ***creator*** *of A. When the direction of the arrow for ξ is clear in the context, we sometimes drop the arrow and simply say that A is created by ξ, or ξ is the creator of A.*

▶ **Definition 3.** *We classify the pairs in $\mathsf{Pers}_*(\mathcal{U})$ and define their* ***representatives*** *respectively as follows:*

- *A p-pair of the form $(\searrow\gamma, \searrow\eta)$ is called* ***closed-open****. The representative for $(\searrow\gamma, \searrow\eta)$ is a tuple of chains (z, A) with $z = \partial(A)$, where z is a p-chain created by $\searrow\gamma$ and A is a $(p+1)$-chain created by $\searrow\eta$.*
- *A p-pair of the form $(\nwarrow\gamma, \nwarrow\eta)$ is called* ***open-closed****. The representative for $(\nwarrow\gamma, \nwarrow\eta)$ is a tuple of chains (A, z) with $z = \partial(A)$, where A is a $(p+1)$-chain created by $\nwarrow\gamma$ and z is a p-chain created by $\nwarrow\eta$.*
- *A p-pair of the form $(\searrow\gamma, \nwarrow\eta)$ is called* ***closed-closed****. The representative for $(\searrow\gamma, \nwarrow\eta)$ is a tuple of chains (z, A, z') with $z + z' = \partial(A)$, where z is a p-chain created by $\searrow\gamma$ and z' is a p-chain created by $\nwarrow\eta$. Notice that A can contain any $(p+1)$-cells in $\mathcal{U}$.*

▶ Remark 4. Representatives defined above are a special case of general zigzag representatives defined in [13]. For example, in a representative (z, A, z') for a closed-closed pair, having $z + z' = \partial(A)$ ensures that $[z] = [z']$ in $\mathsf{H}_*(L_n)$, and the definition in [13] has similar requirements. Notice that we may have $z = z'$, for which A is simply empty.

▶ Remark 5. To perform the update, we shall maintain a representative for each pair in the up-down filtration that we work on, which consists of at most three chains. This is only a slight difference from the two chains per bar maintained in [4], where one chain is from the "R-matrix" and another is from the "V-matrix".

We present below a fact helpful for the proof of correctness of algorithms in this paper:

▶ **Proposition 6.** *Let π be a set of pairs of additions and deletions in $\mathcal{U}$ s.t. each addition and deletion in $\mathcal{U}$ appears exactly once in π. If each pair in π admits a representative, then the pairs in π generate the intervals in $\mathsf{Pers}_*(\mathcal{U})$.*

Proof. See the full version of the paper. ◀

3.3 Using conversion for easier updates

Utilizing the conversion in Section 3.1, our algorithms work on the corresponding up-down filtrations of $\mathcal{F}$ and $\mathcal{F}'$ for the operations in Section 2.1. Specifically, we maintain the barcode and representatives for the corresponding up-down filtration, and the barcode for the original zigzag filtration can be derived from the bijection ϕ as in Theorem 1. Notice that representatives for the pairs in the up-down filtration are essential for performing the update. Moreover, by Proposition 6, the correctness of our update algorithms follows from the correctness of the representatives that we maintain during the update operations. We briefly describe the ideas for the update in this section; see the full version of the paper for details.

Outward/inward switch. Since the switch in Equation (4) swaps two inclusions of different directions, the corresponding up-down filtrations before and after the switch are the same (see Section 3.1). So the time complexity for the update is $O(1)$.

Forward/backward switch. Corresponding to a forward (resp. backward) switch on the original zigzag filtration, there is a forward (resp. backward) switch in the up-down filtration [7]. To perform the update for forward/backward switches on up-down filtrations, we utilize the algorithm for the transposition diamond in [13], which takes $O(m)$ time. Observe that the algorithm in [4] cannot be used on up-down filtrations.

Inward expansion. For the inward expansion from $\mathcal{F}'$ to $\mathcal{F}$ in Equation (5), assume that the up-down filtration $\mathcal{U}'$ corresponding to $\mathcal{F}'$ has the following $m-2$ additions and deletions:

$$\mathcal{U}' : \bullet \xhookrightarrow{\xi_0} \bullet \xhookrightarrow{\xi_1} \bullet \cdots \bullet \xhookrightarrow{\xi_{n-2}} \bullet \xhookleftarrow{\xi_{n-1}} \bullet \xhookleftarrow{\xi_n} \bullet \cdots \bullet \xhookleftarrow{\xi_{m-3}} \bullet$$

Let $\hat{\sigma}$ be the Δ-cell corresponding to the addition of the simplex σ to K_{i-1} in $\mathcal{F}$. We then insert the addition and deletion of $\hat{\sigma}$ in the middle of $\mathcal{U}'$ to get an up-down filtration $\mathcal{U}^+$:

$$\mathcal{U}^+ : \bullet \xhookrightarrow{\xi_0} \bullet \xhookrightarrow{\xi_1} \bullet \cdots \bullet \xhookrightarrow{\xi_{n-2}} \bullet \xhookrightarrow{\hat{\sigma}} \bullet \xhookleftarrow{\hat{\sigma}} \bullet \xhookleftarrow{\xi_{n-1}} \bullet \xhookleftarrow{\xi_n} \bullet \cdots \bullet \xhookleftarrow{\xi_{m-3}} \bullet \tag{8}$$

To get the barcode and representatives for $\mathcal{U}^+$ from those for $\mathcal{U}'$, we utilize the algorithm for the injective and surjective diamond in [13], which takes $O(m^2)$ time. After this, we perform forward and backward switches on $\mathcal{U}^+$ to place the two arrows $\xhookrightarrow{\hat{\sigma}}$ and $\xhookleftarrow{\hat{\sigma}}$ in appropriate positions to form $\mathcal{U}$, which is the up-down filtration corresponding to $\mathcal{F}$. Since no more than $O(m)$ switches are performed, the switches in total takes $O(m^2)$ time. Therefore, the update for inward expansion takes $O(m^2)$ time. Notice that the above transition from $\mathcal{U}'$ to $\mathcal{U}^+$ and from $\mathcal{U}^+$ to $\mathcal{U}$ is valid because $\hat{\sigma}$ has no cofaces in $\mathcal{U}$ based on the conversion in [5].

Computing representatives for an initial up-down filtration. In order to perform a sequence of updates starting from an initial zigzag filtration, we need to compute the barcode and representatives for the initial up-down filtration. This can be done by adapting the algorithm in [13] which takes $O(m^3)$ time for an initial filtration of length m.

4 Outward contraction

Recall that an outward contraction is the following operation:

$$\begin{array}{c} \mathcal{F} : K_0 \leftrightarrow \cdots \leftrightarrow K_{i-2} \leftrightarrow K_{i-1} \xhookleftarrow{\sigma} K_i \xhookrightarrow{\sigma} K_{i+1} \leftrightarrow K_{i+2} \leftrightarrow \cdots \leftrightarrow K_m \\ \Downarrow \\ \mathcal{F}' : K_0 \leftrightarrow \cdots \leftrightarrow K_{i-2} \leftrightarrow K'_i \leftrightarrow K_{i+2} \leftrightarrow \cdots \leftrightarrow K_m \end{array} \tag{9}$$

where $K'_i = K_{i-1} = K_{i+1}$. We also assume that σ is a p-simplex. Notice that the indices for $\mathcal{F}'$ are not consecutive, i.e., $i-1$ and $i+1$ are skipped.

Adjacency change. We first describe the challenge in updating the barcodes. Let $\mathcal{U}$ and $\mathcal{U}'$ be the up-down filtrations corresponding to $\mathcal{F}$ and $\mathcal{F}'$ respectively. According to [5], the first step of the conversion from $\mathcal{F}$ to $\mathcal{U}$ is to treat each repeatedly added simplex in $\mathcal{F}$ as a new copy of a Δ-cell. This results in a zigzag filtration of Δ-complexes which we name as $\hat{\mathcal{F}}$. Notice that $\mathcal{F}$ and $\hat{\mathcal{F}}$ are still isomorphic (see the example in Figure 4). However, by treating simplices as distinct Δ-cells in $\hat{\mathcal{F}}$, one can achieve the conversion to the up-down filtration [5]. Since every simplex added in $\mathcal{F}$ must be deleted later, let $K_j \overset{\sigma}{\hookrightarrow} K_{j+1}$ be the addition associated to the deletion $K_{i-1} \overset{\sigma}{\hookleftarrow} K_i$ and let $K_k \overset{\sigma}{\hookleftarrow} K_{k+1}$ be the deletion associated to the addition $K_i \overset{\sigma}{\hookrightarrow} K_{i+1}$ in $\mathcal{F}$:

$$\begin{array}{c} \mathcal{F}: \cdots \leftrightarrow K_j \overset{\sigma}{\hookrightarrow} K_{j+1} \leftrightarrow \cdots \leftrightarrow K_{i-1} \overset{\sigma}{\hookleftarrow} K_i \overset{\sigma}{\hookrightarrow} K_{i+1} \leftrightarrow \cdots \leftrightarrow K_k \overset{\sigma}{\hookleftarrow} K_{k+1} \leftrightarrow \cdots \\ \Downarrow \\ \hat{\mathcal{F}}: \cdots \leftrightarrow \hat{K}_j \overset{\sigma_1}{\hookrightarrow} \hat{K}_{j+1} \leftrightarrow \cdots \leftrightarrow \hat{K}_{i-1} \overset{\sigma_1}{\hookleftarrow} \hat{K}_i \overset{\sigma_2}{\hookrightarrow} \hat{K}_{i+1} \leftrightarrow \cdots \leftrightarrow \hat{K}_k \overset{\sigma_2}{\hookleftarrow} \hat{K}_{k+1} \leftrightarrow \cdots \end{array}$$

In the above sequences, when σ is added to K_j, we denote the corresponding Δ-cell in $\hat{\mathcal{F}}$ as σ_1, and when σ is added to K_i, we denote the corresponding Δ-cell in $\hat{\mathcal{F}}$ as σ_2. Let Σ_1 be the set of $(p+1)$-cofaces of σ added between $K_j \overset{\sigma}{\hookrightarrow} K_{j+1}$ and $K_{i-1} \overset{\sigma}{\hookleftarrow} K_i$ in $\mathcal{F}$, and let Σ_2 be the set of $(p+1)$-cofaces of σ added between $K_i \overset{\sigma}{\hookrightarrow} K_{i+1}$ and $K_k \overset{\sigma}{\hookleftarrow} K_{k+1}$ in $\mathcal{F}$. Furthermore, let $\hat{\Sigma}_1, \hat{\Sigma}_2$ be the sets of Δ-cells in $\hat{\mathcal{F}}$ corresponding to Σ_1, Σ_2 respectively. Since adjacency of simplices/cells in $\mathcal{F}$ and $\hat{\mathcal{F}}$ are the same, we have that cells in $\hat{\Sigma}_1$ have σ_1 in their boundaries and cells in $\hat{\Sigma}_2$ have σ_2 in their boundaries.

Now consider $\mathcal{F}'$ and the corresponding $\hat{\mathcal{F}}'$ after the contraction:

$$\begin{array}{c} \mathcal{F}': \cdots \leftrightarrow K_j \overset{\sigma}{\hookrightarrow} K_{j+1} \leftrightarrow \cdots \leftrightarrow K'_i \leftrightarrow \cdots \leftrightarrow K_k \overset{\sigma}{\hookleftarrow} K_{k+1} \leftrightarrow \cdots \\ \Downarrow \\ \hat{\mathcal{F}}': \cdots \leftrightarrow \hat{K}_j \overset{\sigma_0}{\hookrightarrow} \hat{K}_{j+1} \leftrightarrow \cdots \leftrightarrow \hat{K}'_i \leftrightarrow \cdots \leftrightarrow \hat{K}_k \overset{\sigma_0}{\hookleftarrow} \hat{K}_{k+1} \leftrightarrow \cdots \end{array}$$

Because of the contraction, $K_j \overset{\sigma}{\hookrightarrow} K_{j+1}$ and $K_k \overset{\sigma}{\hookleftarrow} K_{k+1}$ become associated addition and deletion of σ in $\mathcal{F}'$. For clarity, we name the Δ-cell in $\hat{\mathcal{F}}'$ corresponding to the σ added in K_j as σ_0. Based on the conversion in [5], we have that after the contraction, cells in $\hat{\Sigma}_1 \cup \hat{\Sigma}_2$ now both have σ_0 in their boundaries in $\hat{\mathcal{F}}'$ (see τ_1, τ_2 in Figure 4 for an example). We also notice that σ_1 and σ_2 have the same boundary because after σ is added to K_j in $\mathcal{F}$, the boundary simplices of σ are never deleted before σ is deleted from K_k (in general, different Δ-cells in the converted up-down filtration corresponding to the same simplex could have different boundaries). The change can then be formally stated as follows:

▶ **Observation 7.** *After the outward contraction, the boundary p-cell σ_1 of $(p+1)$-cells in $\hat{\Sigma}_1$ and the boundary p-cell σ_2 of $(p+1)$-cells in $\hat{\Sigma}_2$ are identified as the same p-cell σ_0 in $\hat{\mathcal{F}}'$ (and $\mathcal{U}'$).*

While the cell identification seems to have no effect in $\hat{\mathcal{F}}$ and $\hat{\mathcal{F}}'$ as in Figure 4 because σ_1 and σ_2 never appear in the same Δ-complex, the effect is evident in the up-down filtrations $\mathcal{U}$ and $\mathcal{U}'$ (which we work on) where all cells eventually appear in the same Δ-complex. In Figure 4, σ_1 is a boundary edge of τ_1 and σ_2 is a boundary edge of τ_2 in $\mathcal{U}$. After the contraction, both τ_1 and τ_2 have σ_0 in their boundaries in $\mathcal{U}'$. To better understand the conversion, we notice that to build $\mathcal{U}$ from $\hat{\mathcal{F}}$, one only needs to list all the additions first and then the deletions in $\hat{\mathcal{F}}$, following their orders in $\hat{\mathcal{F}}$ [5] (conversion from $\hat{\mathcal{F}}'$ to $\mathcal{U}'$ is the same); see also Section 3.1.

Key ideas for the computation. We need to identify σ_1, σ_2 in $\mathcal{U}$ to obtain $\mathcal{U}'$. This can be alternatively done by attaching an additional $(p+1)$-cell χ whose boundary equals $\sigma_1 + \sigma_2$ and then "collapsing" χ onto one of its boundary p-cell (see Figure 5 for $p = 1$). Therefore, our solution is that we first perform an inward expansion in the middle of $\mathcal{U}$ with $\searrow\chi$ and

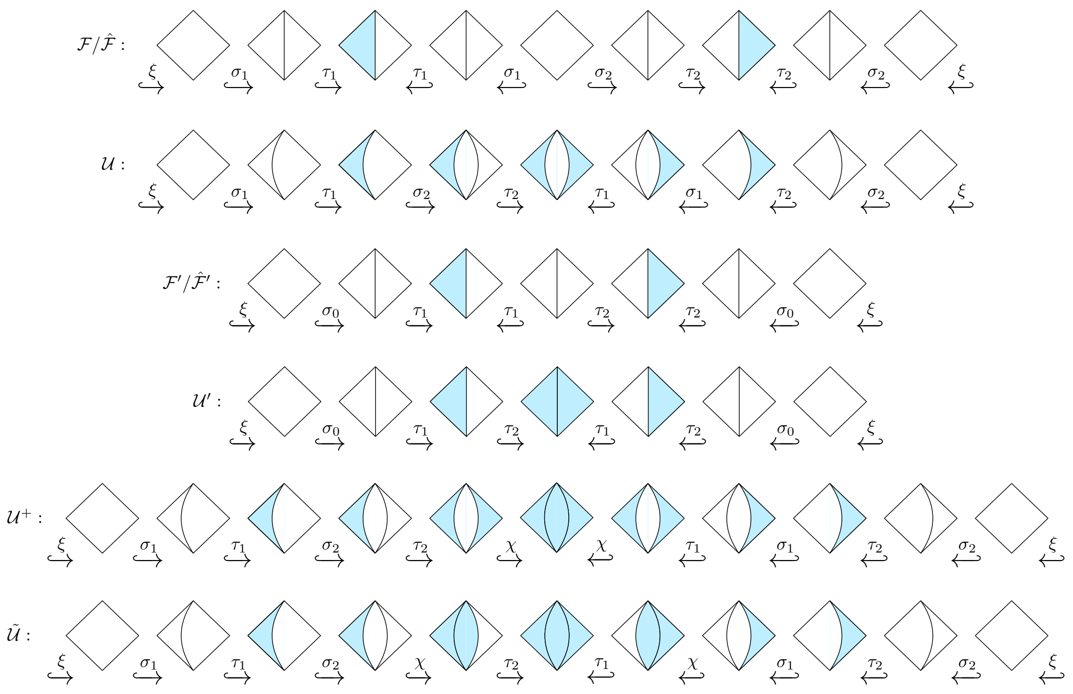
$\mathcal{F}/\hat{\mathcal{F}}$:
ξ σ_1 τ_1 τ_1 σ_1 σ_2 τ_2 τ_2 σ_2 ξ
$\mathcal{U}$:
ξ σ_1 τ_1 σ_2 τ_2 τ_1 σ_1 τ_2 σ_2 ξ
$\mathcal{F}'/\hat{\mathcal{F}}'$:
ξ σ_0 τ_1 τ_1 τ_2 τ_2 σ_0 ξ
$\mathcal{U}'$:
ξ σ_0 τ_1 τ_2 τ_1 τ_2 σ_0 ξ
$\mathcal{U}^+$:
ξ σ_1 τ_1 σ_2 τ_2 χ χ τ_1 σ_1 τ_2 σ_2 ξ
$\tilde{\mathcal{U}}$:
ξ σ_1 τ_1 σ_2 χ τ_2 τ_1 χ σ_1 τ_2 σ_2 ξ

Figure 4 An example illustrating the adjacency change of Δ-cells and the algorithm ideas for outward contraction. Complexes in $\mathcal{F}$ and $\hat{\mathcal{F}}$ are isomorphic and so $\mathcal{F}$ and $\hat{\mathcal{F}}$ ($\mathcal{F}'$ and $\hat{\mathcal{F}}'$ as well) are combined together where the arrows are labelled by the Δ-cells. For simplicity, the filtrations do not start and end with empty complexes.

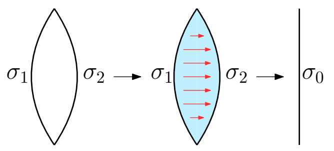

Figure 5 Attaching a $(p+1)$-cell χ with boundary being $\sigma_1+\sigma_2$ and "collapsing" χ to identify σ_1 and σ_2.

$\nwarrow\chi$ added, to get a new up-down filtration $\mathcal{U}^+$ (as shown in Figure 4). Then, we apply several forward and backward switches to make $\searrow\chi$ immediately after $\searrow\sigma_2$ and make $\nwarrow\chi$ immediately before $\nwarrow\sigma_1$, getting an up-down filtration $\tilde{\mathcal{U}}$ (see Figure 4). The reason for constructing the additional filtration $\mathcal{U}^+$ and then $\tilde{\mathcal{U}}$ is that it is not obvious how to compute $\mathsf{Pers}_*(\mathcal{U}')$ directly in an efficient way but $\mathsf{Pers}_*(\tilde{\mathcal{U}})$ can be computed from $\mathsf{Pers}_*(\mathcal{U})$ in $O(m^2)$ time (see the algorithm for inward expansion). We observe that $\mathsf{Pers}_*(\tilde{\mathcal{U}})$ and $\mathsf{Pers}_*(\mathcal{U}')$ are "almost" the same (see Proposition 9 for a formal statement), which leads to an $O(m^2)$ algorithm for computing $\mathsf{Pers}_*(\mathcal{U}')$. To see the relevance of $\mathsf{Pers}_*(\tilde{\mathcal{U}})$ and $\mathsf{Pers}_*(\mathcal{U}')$, first consider the two *non-zigzag* filtrations forming $\mathcal{U}$:

$$\mathcal{U}_u : L^u_0 \hookrightarrow \cdots \hookrightarrow L^u_s \xhookrightarrow{\sigma_1} L^u_{s+1} \hookrightarrow \cdots \xhookrightarrow{\mu} L^u_t \xhookrightarrow{\sigma_2} L^u_{t+1} \xhookrightarrow{\mu'} L^u_{t+2} \hookrightarrow \cdots \hookrightarrow L^u_n,$$
$$\mathcal{U}_d : L^d_0 \hookrightarrow \cdots \hookrightarrow L^d_x \xhookrightarrow{\sigma_2} L^d_{x+1} \hookrightarrow \cdots \xhookrightarrow{\rho} L^d_y \xhookrightarrow{\sigma_1} L^d_{y+1} \xhookrightarrow{\rho'} L^d_{y+2} \hookrightarrow \cdots \hookrightarrow L^d_n.$$

In the above sequences, $\mathcal{U}_u$ is the *ascending* part of $\mathcal{U}$ consisting of all the additions and $\mathcal{U}_d$ is the *descending* part of $\mathcal{U}$ consisting of all the deletions so that $\mathcal{U}$ equals $\mathcal{U}_u$ concatenated with the reversed $\mathcal{U}_d$. Similarly, $\mathcal{U}'$ can be represented as follows:

$$\mathcal{U}'_u : L^u_0 \hookrightarrow \cdots \hookrightarrow L^u_s \xhookrightarrow{\sigma_0} L'^u_{s+1} \hookrightarrow \cdots \xhookrightarrow{\mu} L'^u_t \xhookrightarrow{\mu'} L'^u_{t+2} \hookrightarrow \cdots \hookrightarrow L'^u_n,$$
$$\mathcal{U}'_d : L^d_0 \hookrightarrow \cdots \hookrightarrow L^d_x \xhookrightarrow{\sigma_0} L'^d_{x+1} \hookrightarrow \cdots \xhookrightarrow{\rho} L'^d_y \xhookrightarrow{\rho'} L'^d_{y+2} \hookrightarrow \cdots \hookrightarrow L'^d_n.$$

By the conversion from $\hat{\mathcal{F}}$ to $\mathcal{U}$ and the conversion from $\hat{\mathcal{F}}'$ to $\mathcal{U}'$ [5] described in Section 3.1, the only difference of the sequences of additions in $\mathcal{U}_u$ and $\mathcal{U}'_u$ is that the addition of σ_2 disappears in $\mathcal{U}'_u$ and σ_1 in $\mathcal{U}_u$ is renamed as σ_0 in $\mathcal{U}'_u$. Therefore, the only difference of L'^u_a and L^u_a for $s+1 \le a \le t$ is the renaming of σ_1 to σ_0 in L'^u_a. Furthermore, because of the adjacency change, L'^u_a for $a \ge t+2$ can be considered as derived from L^u_a by identifying σ_2 to σ_1 (renamed as σ_0). Notice that we have symmetric changes from $\mathcal{U}_d$ to $\mathcal{U}'_d$, i.e., the addition of σ_1 disappears and σ_1 is identified to σ_2 which is renamed as σ_0.

Moreover, we write $\tilde{\mathcal{U}}$ as follows:

$$\tilde{\mathcal{U}}_u : L^u_0 \hookrightarrow \cdots \hookrightarrow L^u_s \xhookrightarrow{\sigma_1} L^u_{s+1} \cdots \xhookrightarrow{\mu} L^u_t \xhookrightarrow{\sigma_2} \tilde{L}^u_{t+1} \xhookrightarrow{\chi} \tilde{L}^u_{t+2} \xhookrightarrow{\mu'} \tilde{L}^u_{t+3} \hookrightarrow \cdots \hookrightarrow \tilde{L}^u_{n+1},$$
$$\tilde{\mathcal{U}}_d : L^d_0 \hookrightarrow \cdots \hookrightarrow L^d_x \xhookrightarrow{\sigma_2} L^d_{x+1} \cdots \xhookrightarrow{\rho} L^d_y \xhookrightarrow{\sigma_1} \tilde{L}^d_{y+1} \xhookrightarrow{\chi} \tilde{L}^d_{y+2} \xhookrightarrow{\rho'} \tilde{L}^d_{y+3} \hookrightarrow \cdots \hookrightarrow \tilde{L}^d_{n+1}.$$

We then observe the following: In $\tilde{\mathcal{U}}_u$, the effect of the addition of σ_2 is immediately annulled due to the subsequent addition of χ because σ_2 can then be identified with σ_1 by collapsing χ. Specifically, we have that $\tilde{L}^u_a$ is homotopy equivalent to L'^u_{a-1} for each $a \ge t+3$ (see Figure 4 for an example). Similarly, in $\tilde{\mathcal{U}}_d$, the effect of the addition of σ_1 is annulled due to the addition of χ and we have that $\tilde{L}^d_a$ is homotopy equivalent to L'^d_{a-1} for each $a \ge y+3$. Furthermore, by the nature of persistence [8], σ_2 must be paired with χ in $\tilde{\mathcal{U}}_u$ (because adding σ_2 creates a cycle $\sigma_1 + \sigma_2$ which is immediately killed by adding χ) and σ_1 must be paired with χ in $\tilde{\mathcal{U}}_d$ (with a similar reason). We now have:

▶ **Definition 8.** *For each addition or deletion $\nwsearrow\eta$ in $\tilde{\mathcal{U}}$ s.t. $\nwsearrow\eta \notin \{\searrow\sigma_2, \searrow\chi, \nwarrow\chi, \nwarrow\sigma_1\}$, define its corresponding addition/deletion in $\mathcal{U}'$, denoted $\theta(\nwsearrow\eta)$, as follows: $\theta(\searrow\sigma_1) = \searrow\sigma_0$; $\theta(\nwarrow\sigma_2) = \nwarrow\sigma_0$; for any remaining $\nwsearrow\eta$, the corresponding $\theta(\nwsearrow\eta)$ is the same.*

▶ **Proposition 9.** *Given $\mathsf{Pers}_*(\tilde{\mathcal{U}})$, one only needs to do the following to obtain $\mathsf{Pers}_*(\mathcal{U}')$: Ignoring the pairs $(\searrow\sigma_2, \searrow\chi)$ and $(\nwarrow\chi, \nwarrow\sigma_1)$ in $\mathsf{Pers}_*(\tilde{\mathcal{U}})$, for each remaining pair $(\nwsearrow\eta, \nwsearrow\gamma) \in \mathsf{Pers}_*(\tilde{\mathcal{U}})$, produce a corresponding pair $(\theta(\nwsearrow\eta), \theta(\nwsearrow\gamma)) \in \mathsf{Pers}_*(\mathcal{U}')$.*

Proof. As mentioned, we have the following equivalence of homotopy types between complexes in $\tilde{\mathcal{U}}_u$ and $\mathcal{U}'_u$: (i) complexes at indices from 0 to s in $\tilde{\mathcal{U}}_u$ and $\mathcal{U}'_u$ are the same; (ii) L^u_a and L'^u_a are the same complex for $s+1 \le a \le t$ with the only difference on the naming of σ_0 and σ_1; (iii) $\tilde{L}^u_a$ is homotopy equivalent to L'^u_{a-1} for $a \ge t+3$. Therefore, if we view $L^u_t \hookrightarrow \tilde{L}^u_{t+3}$ as a single inclusion in $\tilde{\mathcal{U}}_u$, the induced zigzag modules $\mathsf{H}_*(\tilde{\mathcal{U}}_u)$ and $\mathsf{H}_*(\mathcal{U}'_u)$ are isomorphic. We also have similar results for $\mathsf{H}_*(\tilde{\mathcal{U}}_d)$ and $\mathsf{H}_*(\mathcal{U}'_d)$. Hence, $\mathsf{H}_*(\tilde{\mathcal{U}})$ and $\mathsf{H}_*(\mathcal{U}')$ are isomorphic (by treating $L^u_t \hookrightarrow \tilde{L}^u_{t+3}$ and $L^d_y \hookrightarrow \tilde{L}^d_{y+3}$ as single inclusions). The proposition then follows. ◀

▶ Remark. An alternative proof of Proposition 9 follows from Proposition 6 and the correctness of the representatives we obtain for $\mathsf{Pers}_*(\mathcal{U}')$ (see the full version of the paper for details).

▶ **Example.** *Ignoring the two pairs without correspondence, we have the following mapping of pairs from $\mathsf{Pers}(\tilde{\mathcal{U}})$ to $\mathsf{Pers}(\mathcal{U}')$ for the example in Figure 4 (ξ is a fixed edge in the boundary of the diamond): $(\searrow\xi, \searrow\tau_2) \mapsto (\searrow\xi, \searrow\tau_2)$, $(\searrow\sigma_1, \searrow\tau_1) \mapsto (\searrow\sigma_0, \searrow\tau_1)$, $(\nwarrow\tau_1, \nwarrow\xi) \mapsto (\nwarrow\tau_1, \nwarrow\xi)$, $(\nwarrow\tau_2, \nwarrow\sigma_2) \mapsto (\nwarrow\tau_2, \nwarrow\sigma_0)$.*

Algorithm. From the maintained $\mathsf{Pers}_*(\mathcal{U})$ and its representatives, we first compute $\mathsf{Pers}_*(\tilde{\mathcal{U}})$ and the representatives in $O(m^2)$ time. Based on Proposition 9, $\mathsf{Pers}_*(\mathcal{U}')$ can then be easily derived. To get the representatives for $\mathsf{Pers}_*(\mathcal{U}')$, we notice that chains and cycles in $\tilde{\mathcal{U}}$ can be naturally "re-written" as chains and cycles in $\mathcal{U}'$, e.g., the chain $\tau_1 + \tau_2 + \chi$ in $\tilde{\mathcal{U}}$ can be re-written as $\tau_1 + \tau_2$ in $\mathcal{U}'$ where their boundary stays the same. We provide in the full version of the paper the details of re-writing the representatives for $\mathsf{Pers}_*(\tilde{\mathcal{U}})$ to get the representatives for $\mathsf{Pers}_*(\mathcal{U}')$. Since the representative re-writing takes $O(m^2)$ time (see the full version), we conclude the following:

▶ **Theorem 10.** *Zigzag barcode for an outward contraction can be updated in $O(m^2)$ time.*

5 Outward expansion

An outward expansion is the reverse operation of an outward contraction in Equation (9), and we let $\mathcal{F}$, $\mathcal{F}'$, and σ be as defined in Equation (9) throughout the section, where σ is also a p-simplex. Notice that for outward expansion, we are given the barcode and representatives for $\mathcal{U}'$ (as in Section 4), and the goal is to compute those for $\mathcal{U}$. We observe the following adjacency change reverse to that for outward contraction:

▶ **Observation 11.** *After the outward expansion, the boundary p-cell for some $(p+1)$-cofaces of σ_0 in $\mathcal{U}'$ becomes σ_1 in $\mathcal{U}$, while the boundary p-cell for the other $(p+1)$-cofaces of σ_0 in $\mathcal{U}'$ becomes σ_2 in $\mathcal{U}$.*

We thereby do the following:

1. Construct $\tilde{\mathcal{U}}$ (as in Figure 4) by first inserting $\searrow\sigma_2$ and $\searrow\chi$ into the ascending part $\mathcal{U}'_u$ of $\mathcal{U}'$; the position of the insertion can be derived from the position of the expansion in the original $\mathcal{F}$ and $\mathcal{F}'$. Identify σ_0 as the right-most Δ-cell in $\mathcal{U}'_u$ corresponding to σ added before the newly inserted σ_2. Rename σ_0 as σ_1. Change the boundary cell σ_0 (renamed as σ_1) to σ_2 for those $(p+1)$-cofaces of σ_0 in $\mathcal{U}'_u$ added after the newly inserted σ_2. This finishes the construction of the ascending part of $\tilde{\mathcal{U}}$. Constructing the descending part of $\tilde{\mathcal{U}}$ is symmetric.
2. Re-write the representatives for $\mathsf{Pers}_*(\mathcal{U}')$ to get the representatives for the the corresponding pairs in $\mathsf{Pers}_*(\tilde{\mathcal{U}})$ (see Proposition 9). We also let the two extra pairs $(\searrow\sigma_2, \searrow\chi)$ and $(\nwarrow\chi, \nwarrow\sigma_1)$ have the representative $(\sigma_1 + \sigma_2, \chi)$ and $(\chi, \sigma_1 + \sigma_2)$ respectively. Details of the representative re-writing are provided in the full version of the paper.
3. From $\mathsf{Pers}_*(\tilde{\mathcal{U}})$ and the representatives, obtain $\mathsf{Pers}_*(\mathcal{U})$ and the representatives in $O(m^2)$ time (see the algorithm for inward contraction in Section 6).

We conclude:

▶ **Theorem 12.** *Zigzag barcode for an outward expansion can be updated in $O(m^2)$ time.*

6 Inward contraction

Consider the following inward contraction from $\mathcal{F}$ to $\mathcal{F}'$:

$$\begin{array}{c}\mathcal{F}: K_0 \leftrightarrow \cdots \leftrightarrow K_{i-2} \leftrightarrow K_{i-1} \overset{\sigma}{\hookrightarrow} K_i \overset{\sigma}{\hookleftarrow} K_{i+1} \leftrightarrow K_{i+2} \leftrightarrow \cdots \leftrightarrow K_m \\ \Downarrow \\ \mathcal{F}': K_0 \leftrightarrow \cdots \leftrightarrow K_{i-2} \leftrightarrow K'_i \leftrightarrow K_{i+2} \leftrightarrow \cdots \leftrightarrow K_m\end{array}$$

Let $\mathcal{U}$ and $\mathcal{U}'$ be the up-down filtrations corresponding to $\mathcal{F}$ and $\mathcal{F}'$ respectively. Moreover, let $\hat{\sigma}$ be the Δ-cell corresponding to the addition of the simplex σ to K_{i-1} in $\mathcal{F}$. We first perform forward and backward switches on $\mathcal{U}$ to place the additions and deletions of $\hat{\sigma}$ in the center, obtaining the filtration

$$\mathcal{U}^+ : \bullet \xhookrightarrow{\xi_0} \bullet \xhookrightarrow{\xi_1} \bullet \cdots \bullet \xhookrightarrow{\xi_{n-2}} \bullet \xhookrightarrow{\hat{\sigma}} \bullet \xhookleftarrow{\hat{\sigma}} \bullet \xhookleftarrow{\xi_{n-1}} \bullet \xhookleftarrow{\xi_n} \bullet \cdots \bullet \xhookleftarrow{\xi_{m-3}} \bullet$$

as in Equation (8) which we also call $\mathcal{U}^+$. Hence, we can obtain $\mathsf{Pers}_*(\mathcal{U}^+)$ and its representatives in $O(m^2)$ time. To perform the update from $\mathcal{U}^+$ to $\mathcal{U}'$ (an inward contraction on up-down filtrations), we observe an interval mapping behavior (i.e., the alternative re-linking of "unsettled" pairs) different from the one proposed in [13, 14] for inward expansion (see, e.g., Theorem 2.3 in [13]).

The rest of the update has different processes based on whether $\searrow\hat{\sigma}$ is positive or negative in $\mathcal{U}^+$.

6.1 $\searrow\hat{\sigma}$ is positive in $\mathcal{U}^+$

If $\searrow\hat{\sigma}$ is positive in $\mathcal{U}^+$, then $\searrow\hat{\sigma}$ must be paired with $\nwarrow\hat{\sigma}$ in $\mathsf{Pers}_*(\mathcal{U}^+)$ and to obtain $\mathsf{Pers}_*(\mathcal{U}')$ one only needs to delete the pair $(\searrow\hat{\sigma}, \nwarrow\hat{\sigma})$ from $\mathsf{Pers}_*(\mathcal{U}^+)$; see [13, Theorem 2.2]. We then show how to obtain representatives for pairs in $\mathsf{Pers}_*(\mathcal{U}')$. Ignoring the pair $(\searrow\hat{\sigma}, \nwarrow\hat{\sigma})$, call those pairs in $\mathsf{Pers}_*(\mathcal{U}^+)$ whose chains in the representatives contain $\hat{\sigma}$ as *unsettled.* Accordingly, call those pairs in $\mathsf{Pers}_*(\mathcal{U}^+)$ whose chains in the representatives do not contain $\hat{\sigma}$ as *settled.* Notice that representatives for settled pairs stay the same when we consider the pairs to be from $\mathsf{Pers}_*(\mathcal{U}')$. We also have that only closed-closed pairs in $\mathsf{Pers}_*(\mathcal{U}^+)$ can be unsettled because the addition and deletion of $\hat{\sigma}$ are at the center of $\mathcal{U}^+$ (see Definition 3). Let $\overline{z}$ be a cycle containing $\hat{\sigma}$ which is taken from the representative for $(\searrow\hat{\sigma}, \nwarrow\hat{\sigma}) \in \mathsf{Pers}_*(\mathcal{U}^+)$. For an unsettled pair $(\searrow\eta, \nwarrow\gamma) \in \mathsf{Pers}_*(\mathcal{U}^+)$ with a representative (z, A, z'), we have $\hat{\sigma} \in A$. Then, $(z, A + \overline{z}, z')$ is a representative for $(\searrow\eta, \nwarrow\gamma) \in \mathsf{Pers}_*(\mathcal{U}')$, where $z + z' = \partial(A) = \partial(A + \overline{z})$ and $\hat{\sigma} \notin A + \overline{z}$.

6.2 $\searrow\hat{\sigma}$ is negative in $\mathcal{U}^+$

The update for this case has two steps: We first preprocess those unsettled pairs (definition similar as above) where $\hat{\sigma}$ in the representatives can be easily eliminated so that the pairs become settled. After this, the remaining unsettled pairs satisfy a certain criteria and we use the alternative re-linking to produce new pairs for $\mathcal{U}'$.

Step I: Preprocessing. Since $\searrow\hat{\sigma}$ is negative in $\mathcal{U}^+$, $\nwarrow\hat{\sigma}$ must be positive in $\mathcal{U}^+$ (if adding $\hat{\sigma}$ decreases the Betti number, then deleting $\hat{\sigma}$ must increase the Betti number). Let $(\searrow\tau, \searrow\hat{\sigma})$ and $(\nwarrow\hat{\sigma}, \nwarrow\tau')$ be pairs in $\mathsf{Pers}_*(\mathcal{U}^+)$ containing $\searrow\hat{\sigma}$ and $\nwarrow\hat{\sigma}$ respectively. Also suppose that $(\searrow\tau, \searrow\hat{\sigma})$ has the representative (z_*, A_*) and $(\nwarrow\hat{\sigma}, \nwarrow\tau')$ has the representative $(A_\circ, z_\circ)$. Moreover, let Λ be the set of unsettled pairs in $\mathsf{Pers}_*(\mathcal{U}^+)$. Notice that we do not include $(\searrow\tau, \searrow\hat{\sigma})$ and $(\nwarrow\hat{\sigma}, \nwarrow\tau')$ into Λ and so pairs in Λ are all closed-closed.

▶ **Definition 13.** *Define a partial order "$\sqsubset$" on closed-closed pairs in* $\mathsf{Pers}_*(\mathcal{U}^+)$ *s.t.* $(\searrow\eta, \nwarrow\gamma) \sqsubset (\searrow\eta', \nwarrow\gamma')$ *if and only if* η *is added after* η' *and* γ *is deleted before* γ' *in* $\mathcal{U}^+$ *(i.e., the persistence interval generated by* $(\searrow\eta, \nwarrow\gamma)$ *is contained in the persistence interval generated by* $(\searrow\eta', \nwarrow\gamma')$*). We also say that two pairs* $(\searrow\eta, \nwarrow\gamma)$ *and* $(\searrow\eta', \nwarrow\gamma')$ *are* ***comparable*** *if* $(\searrow\eta, \nwarrow\gamma) \sqsubset (\searrow\eta', \nwarrow\gamma')$ *or* $(\searrow\eta', \nwarrow\gamma') \sqsubset (\searrow\eta, \nwarrow\gamma)$*.*

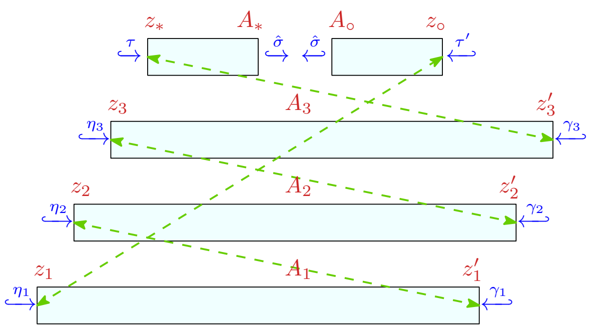

Figure 6 An example of $\ell = 3$ where the dashed lines alternatively re-link the additions and deletions to form new pairs in $\mathsf{Pers}_*(\mathcal{U}')$.

We then make certain pairs in Λ settled and delete these pairs from Λ so that Λ satisfies the following: (i) no two pairs in Λ are comparable; (ii) for each pair $(\searrow\eta, \nwarrow\gamma)$ in Λ, the cell η is added before τ and the cell γ is deleted after τ' in $\mathcal{U}^+$. See the full version of the paper for details of the preprocessing. Finally, we let each settled pair in $\mathsf{Pers}_*(\mathcal{U}^+)$ automatically become a pair in $\mathsf{Pers}_*(\mathcal{U}')$ with the same representative.

Step II: Alternatively re-linking unsettled pairs. After the preprocessing step, if $\Lambda = \varnothing$, then $(\searrow\tau, \nwarrow\tau')$ forms a pair in $\mathsf{Pers}_*(\mathcal{U}')$ with a representative $(z_*, A_* + A_\circ, z_\circ)$ and we have finished the update. If $\Lambda \neq \varnothing$, we order the pairs in Λ as $\{(\searrow\eta_j, \nwarrow\gamma_j) \mid j = 1, 2, \ldots, \ell\}$ s.t. η_j is added before η_{j+1} for each j. Since pairs in Λ are not comparable, we have that γ_j is deleted before γ_{j+1} for each j because it would then be true that $(\searrow\eta_{j+1}, \nwarrow\gamma_{j+1}) \sqsubset (\searrow\eta_j, \nwarrow\gamma_j)$ if γ_j is deleted after γ_{j+1} for a j. See Figure 6. Additionally, let each $(\searrow\eta_j, \nwarrow\gamma_j)$ have a representative (z_j, A_j, z'_j).

Now, alternatively re-link the additions and deletions in the unsettled pairs as follows (see Figure 6 for an example of $\ell = 3$):

1. Form a pair $(\searrow\eta_1, \nwarrow\tau') \in \mathsf{Pers}_*(\mathcal{U}')$ with a representative $(z_1, A_1 + A_\circ, z'_1 + z_\circ)$, where $\hat{\sigma} \notin A_1 + A_\circ$.
2. Form a pair $(\searrow\tau, \nwarrow\gamma_\ell) \in \mathsf{Pers}_*(\mathcal{U}')$ with a representative $(z_\ell + z_*, A_\ell + A_*, z'_\ell)$, where $\hat{\sigma} \notin A_\ell + A_*$.
3. For each $j = 1, \ldots, \ell - 1$: Form a pair $(\searrow\eta_{j+1}, \nwarrow\gamma_j) \in \mathsf{Pers}_*(\mathcal{U}')$ with a representative $(z_j + z_{j+1}, A_j + A_{j+1}, z'_j + z'_{j+1})$, where $\hat{\sigma} \notin A_j + A_{j+1}$.

We now conclude:

▶ **Theorem 14.** *Zigzag barcode for an inward contraction can be updated in $O(m^2)$ time.*

Proof. The correctness of the updating follows from the correctness of the representatives we set for $\mathsf{Pers}_*(\mathcal{U}')$, which can be verified from the operations presented above. The costliest steps in the update are the chain summations, and there are $O(m)$ of them each taking $O(m)$ time. So the time complexity is $O(m^2)$. ◀

7 Conclusion

We have presented update algorithms for maintaining barcodes over a changing zigzag filtration. Two main questions come out of this research: (i) The representatives for the barcodes are maintained only in the up-down filtrations. An open question that remains is whether this can help maintain the representatives for the original zigzag filtration in quadratic time over all operations. (ii) Are there other interesting applications of the update algorithms presented in this paper? Their applications to computing vineyards for dynamic point clouds, multiparameter persistence, and dynamic network are mentioned in [6, 16]. We hope that there are other dynamic settings which can benefit from efficient zigzag updates.

References

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

2 Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, pages 247–256, 2009.

3 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

4 David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy Morozov. Vines and vineyards by updating persistence in linear time. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, pages 119–126, 2006.

5 Tamal K. Dey and Tao Hou. Fast computation of zigzag persistence. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 43:1–43:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

6 Tamal K. Dey and Tao Hou. Updating barcodes and representatives for zigzag persistence. *arXiv preprint*, 2022. `arXiv:2112.02352`.

7 Tamal K. Dey, Tao Hou, and Salman Parsa. Revisiting graph persistence for updates and efficiency. In *18th Algorithms and Data Structures Symposium (WADS 2023),* to appear, 2023.

8 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 454–463. IEEE, 2000.

9 Samuel Eilenberg and J. A. Zilber. Semi-simplicial complexes and singular homology. *Annals of Mathematics*, 51(3):499–513, 1950. URL: `http://www.jstor.org/stable/1969364`.

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

11 Allen Hatcher. *Algebraic Topology.* Cambridge University Press, 2002.

12 Michael Kerber and Hannah Schreiber. Barcodes of towers and a streaming algorithm for persistent homology. *Discrete & computational geometry*, 61:852–879, 2019.

13 Clément Maria and Steve Y. Oudot. Zigzag persistence via reflections and transpositions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 181–199. SIAM, 2014.

14 Clément Maria and Steve Y. Oudot. Computing zigzag persistent cohomology. *arXiv preprint*, 2016. `arXiv:1608.06039`.

15 Steve Y. Oudot and Donald R. Sheehy. Zigzag zoology: Rips zigzags for homology inference. *Foundations of Computational Mathematics*, 15(5):1151–1186, 2015.

16 Sarah Tymochko, Elizabeth Munch, and Firas A. Khasawneh. Using zigzag persistent homology to detect Hopf bifurcations in dynamical systems. *Algorithms*, 13(11):278, 2020.