



Cup Product Persistence and Its Efficient Computation

Tamal K. Dey  

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abhishek Rathod  

Department of Computer Science, Ben Gurion University, Beersheba, Israel

Abstract

It is well-known that the cohomology ring has a richer structure than homology groups. However, until recently, the use of cohomology in persistence setting has been limited to speeding up of barcode computations. Some of the recently introduced invariants, namely, persistent cup-length, persistent cup modules and persistent Steenrod modules, to some extent, fill this gap. When added to the standard persistence barcode, they lead to invariants that are more discriminative than the standard persistence barcode. In this work, we devise an $O(dn^4)$ algorithm for computing the persistent k -cup modules for all $k \in \{2, \dots, d\}$, where d denotes the dimension of the filtered complex, and n denotes its size. Moreover, we note that since the persistent cup length can be obtained as a byproduct of our computations, this leads to a faster algorithm for computing it for $d \geq 3$. Finally, we introduce a new stable invariant called partition modules of cup product that is more discriminative than persistent cup modules and devise an $O(c(d)n^4)$ algorithm for computing it, where $c(d)$ is subexponential in d .

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

Keywords and phrases Persistent cohomology, cup product, image persistence, persistent cup module

Digital Object Identifier 10.4230/LIPIcs.SoCG.2024.50

Related Version *Full Version*: <https://arxiv.org/abs/2212.01633> [18]

Funding This work is partially supported by the NSF grant CCF 2049010, DMS 2301360 and ERC 101039913.

Acknowledgements We wish to acknowledge helpful initial discussions with Ulrich Bauer and Fabian Lenzen.

1 Introduction

Persistent homology is one of the principal tools in the fast growing field of topological data analysis. A solid algebraic framework [34], a well-established theory of stability [4, 10–12] along with fast algorithms and software [1–3, 8, 26] to compute complete invariants called barcodes of filtrations have led to the successful adoption of single parameter persistent homology as a data analysis tool [19, 20]. This standard persistence framework operates in each (co)homology degree separately and thus cannot capture the interactions across degrees in an apparent way. To achieve this, one may endow a cohomology vector space with the well-known *cup product* forming a graded algebra. Then, the isomorphism type of such graded algebras can reveal information including interactions across degrees. However, even the best known algorithms for determining isomorphism of graded algebras run in exponential time in the worst case [9]. So it is not immediately clear how one may extract new (persistent) invariants from the product structure efficiently in practice.

Cohomology has already shown to be useful in speeding up persistence computations before [1, 2, 8]. It has also been noted that additional structures on cohomology provide an avenue to extract rich topological information [7, 14, 24, 25, 33]. To this end, in a recent study,



© Tamal K. Dey and Abhishek Rathod;

licensed under Creative Commons License CC-BY 4.0

40th International Symposium on Computational Geometry (SoCG 2024).

Editors: Wolfgang Mulzer and Jeff M. Phillips; Article No. 50; pp. 50:1–50:15

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



the authors of [14] introduced the notion of (the persistent version of) an invariant called the *cup length*, which is the maximum number of cocycles with a nonzero product. In another version [15], the authors of [14] introduced an invariant called *barcodes of persistent k -cup modules* which are stable, and can add more discriminating ability (Figure 1). Computing this invariant allows us to capture interactions among various degrees. In Example 1, we provide simple examples for which persistent cup modules can disambiguate filtered spaces where ordinary persistence and persistent cup-length fail. Notice that for a filtered d -complex, the k -cup modules for $k \in \{2, \dots, d\}$ may not be a strictly finer invariant on its own compared to ordinary persistence. It can however add more information as Example 1 illustrates.

► **Example 1.** See Figure 1. Let K^1 be a cell complex obtained by taking a wedge of four circles and two 2-spheres. Let K^2 be a cell complex obtained by taking a wedge of two circles, a sphere and a 2-torus. Let K^3 be a cell complex obtained by taking a wedge of two tori.

► **Remark 2.** Throughout, for a cell complex C , the filtration for which all the k -dimensional cells of C arrive at the same index is referred to as the *natural cell filtration associated to C* .

Consider the natural cell filtrations K^1_\bullet , K^2_\bullet and K^3_\bullet . Standard persistence cannot tell apart K^1_\bullet , K^2_\bullet and K^3_\bullet as the barcode for the three filtrations are the same. Persistent cup length cannot distinguish K^2_\bullet from K^3_\bullet , whereas the barcodes for persistent cup modules for K^1_\bullet , K^2_\bullet and K^3_\bullet are all different.

In Section 3 and 4, we show how to compute the persistent k -cup modules for all $k \in \{2, \dots, d\}$ in $O(dn^4)$ time, where d denotes the dimension of the filtered complex, and n denotes its size. Moreover, since the persistent cup length of a filtration can be obtained as a byproduct of cup modules computation [14], we get an efficient algorithm to compute this invariant as well. Our approach for computing barcodes of persistent k -cup modules involves computing the image persistence of the cup product viewed as a map from the tensor product of the cohomology vector space to the cohomology vector space itself. This approach requires careful bookkeeping of restrictions of cocycles as one processes the simplices in the reverse filtration order. Algorithms for computing image persistence have been studied earlier by Cohen-Steiner et al. [13] and recently by Bauer and Schmah [6]. However, the algorithms in [6, 13] work only for monomorphisms of filtrations making them inapplicable to our setting.

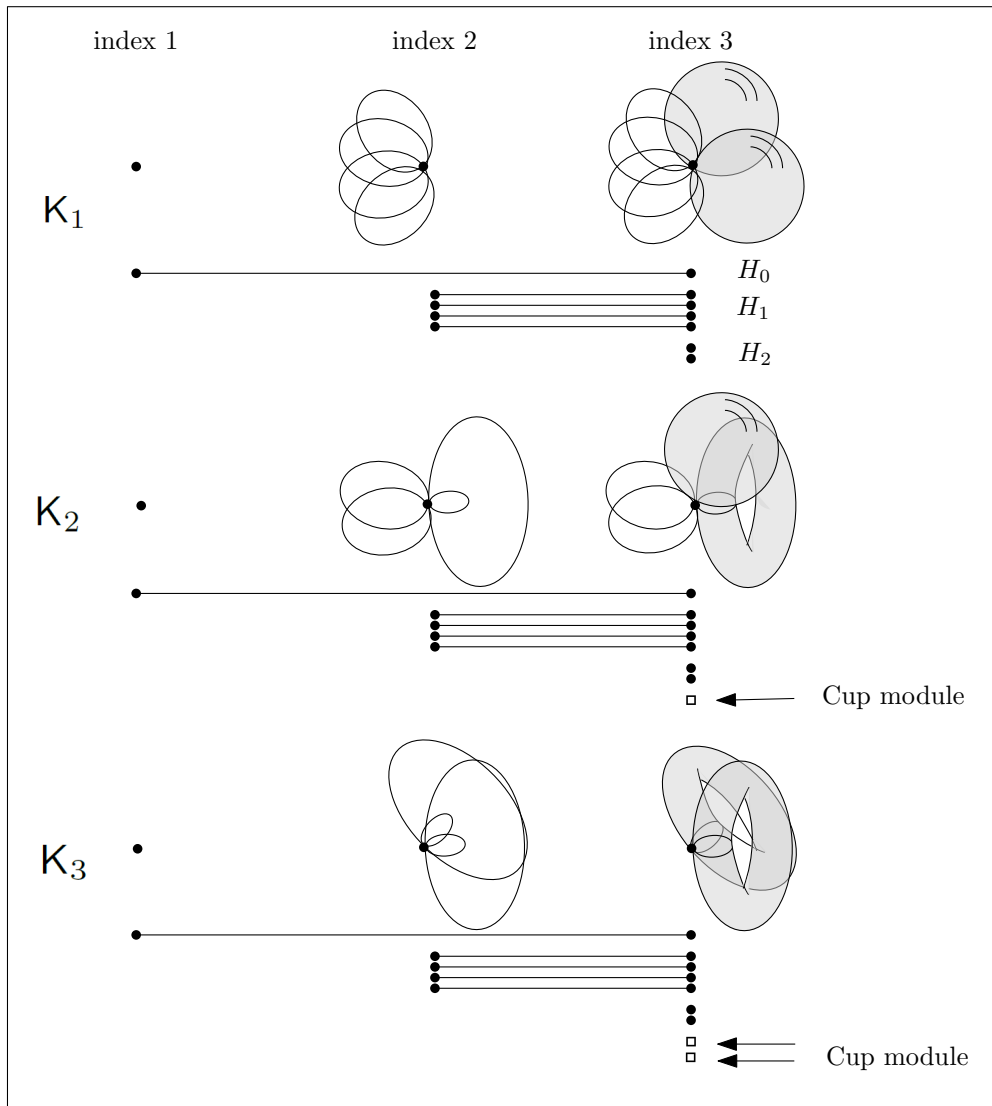
In Section 5, we introduce a new invariant called partition modules of the cup product which is more discriminative than cup modules. In the extended version, we observe that this invariant is stable for Rips and Čech filtrations [18, Section 5.1], and provide an algorithm for computing it in $O(c(d)n^4)$ time, where $c(d)$ is subexponential in d [18, Section 5.2].

2 Background and preliminaries

Throughout, we use n to denote the size of the filtered complex K , $[n]$ to denote the set $\{1, 2, \dots, n\}$ and I to denote the set $\{0, 1, 2, \dots, n\}$.

2.1 Persistent cohomology

In this paper, we work with mod-2 cohomology. We refer the reader to [22, 23] for topological preliminaries. Let P denote a poset category such as \mathbb{N} , \mathbb{Z} , or \mathbb{R} , and **Simp** denote the category of finite simplicial complexes. A P -indexed filtration is a functor $\mathcal{F} : P \rightarrow \mathbf{Simp}$ such that $\mathcal{F}_s \subseteq \mathcal{F}_t$ whenever $s \leq t$. A P -indexed persistence module V_\bullet is a functor from a poset category P to the category of (graded) vector spaces. The morphisms $\psi_{s,t} : V_s \rightarrow V_t$ for $s \leq t$ are referred to as *structure maps*. We assume it to be of *finite type*, that is, V_\bullet is



■ **Figure 1** Example 1 Persistent cup modules distinguishes all three cellular filtrations.

pointwise finite dimensional and all morphisms $\psi_{s,t}$ for $s \leq t$ are isomorphisms outside a finite subset of P . A P -indexed module W is a *submodule* of V if $W_s \subset V_s$ for all $s \in P$ and the structure maps $W_s \rightarrow W_t$ are restrictions of $\psi_{s,t}$ to W_s .

A persistence module V_\bullet defined on a totally ordered set such as \mathbb{N} , \mathbb{Z} , or \mathbb{R} decomposes uniquely up to isomorphism into simple modules called *interval modules* whose structure maps are identity and the vector spaces have dimension one. The support of these interval modules collectively constitute what is called the barcode of V_\bullet and denoted by $B(V_\bullet)$.

When we have a filtration \mathcal{F} on P where the complexes change only at a finite set of values $a_1 < a_2 < \dots < a_n$, we can reindex the filtration with integers, and refine it so that only one simplex is added at every index. Reindexing and refining in this manner one can obtain a simplex-wise filtration of the final simplicial complex K defined on an indexing set with integers. For the remainder of the paper, we assume that the original filtration on P is simplex-wise to begin with. This only simplifies our presentation, and we do not lose generality. With this assumption, we obtain a filtration indexed on I after writing $K_{a_i} = K_i$,

$$K_\bullet : \emptyset = K_0 \hookrightarrow K_1 \hookrightarrow \dots \hookrightarrow K_n = K.$$

50:4 Cup Product Persistence and Its Efficient Computation

Applying the functor C^* , we obtain a persistence module $C^*(K_\bullet)$ of cochain complexes whose structure maps are cochain maps defined by restrictions induced by inclusions:

$$C^*(K_\bullet) : C^*(K_n) \rightarrow C^*(K_{n-1}) \rightarrow \cdots \rightarrow C^*(K_0),$$

and applying the functor H^* , we get a persistence module $H^*(K_\bullet)$ of graded cohomology vector spaces whose structure maps are linear maps induced by the above-mentioned restrictions:

$$H^*(K_\bullet) : H^*(K_n) \rightarrow H^*(K_{n-1}) \rightarrow \cdots \rightarrow H^*(K_0).$$

For simplifying the description of the algorithm, we work with I^{op} -indexed modules $H^*(K_\bullet)$ and $C^*(K_\bullet)$. The barcode $B(M)$ (see section 2.4) of a finite-type P^{op} -module M can be obtained from the barcode $B(N)$ of its associated I^{op} -module N by writing the interval $(j, i) \in B(N)$ for $j < i < n$ as $[a_{j+1}, a_{i+1}) \in B(M)$, and the interval $(j, n) \in B(N)$ as $[a_{j+1}, \infty) \in B(M)$. In this convention, we refer to i (or n) as a *birth index*, j as a *death index*, and intervals of the form $(j, n]$ as *essential bars*.

► **Definition 3** (Restriction of cocycles). *For a filtration K_\bullet , if ζ is a cocycle in complex K_b , but ceases to be a cocycle at K_{b+1} , then ζ^i is defined as $\zeta^i = \zeta \cap C^*(K_i)$ for $i \leq b$, and in this case, we say that ζ^i is the restriction of ζ to index i . For $i > b$, ζ^i is set to the zero cocycle.*

► **Definition 4** (Persistent cohomology basis). *Let $\Omega_K = \{\zeta_{\mathbf{i}} \mid \mathbf{i} \in B(H^*(K_\bullet))\}$ be a set of cocycles, where for every $\mathbf{i} = (d_i, b_i)$, $\zeta_{\mathbf{i}}$ is a cocycle in K_{b_i} but no more a cocycle in K_{b_i+1} . If for every index $j \in [n]$, the cocycle classes $\{[\zeta_{\mathbf{i}}^j] \mid \zeta_{\mathbf{i}} \in \Omega_K\}$ form a basis for $H^*(K_j)$, then we say that Ω_K is a persistent cohomology basis for K_\bullet , and the cocycle $\zeta_{\mathbf{i}}$ is called a representative cocycle for the interval \mathbf{i} . If $b_i = n$, $[\zeta_{\mathbf{i}}]$ is called an essential class.*

2.2 Simplicial cup product

Simplicial cup products connect cohomology groups across degrees. Let \prec be an arbitrary but fixed total order on the vertex set of K . Let ξ and ζ be cocycles of degrees p and q respectively. The cup product of ξ and ζ is the $(p+q)$ -cocycle $\xi \smile \zeta$ whose evaluation on any $(p+q)$ -simplex $\sigma = \{v_0, \dots, v_{p+q}\}$ is given by

$$(\xi \smile \zeta)(\sigma) = \xi(\{v_0, \dots, v_p\}) \cdot \zeta(\{v_p, \dots, v_{p+q}\}). \quad (1)$$

This defines a map $\smile : C^p(K) \times C^q(K) \rightarrow C^{p+q}(K)$, which assembles to give a map $\smile : C^*(K) \times C^*(K) \rightarrow C^*(K)$ for the cochain complex $C^*(K)$. Using the fact that $\delta(\zeta \smile \xi) = \delta\xi \smile \zeta + \xi \smile \delta\zeta$, it follows that \smile induces a map $\smile : H^*(K) \times H^*(K) \rightarrow H^*(K)$. It can be shown that the map \smile is independent of the ordering \prec .

Using the universal property for tensor products and linearity, the bilinear maps for

$$\smile : C^p(K) \times C^q(K) \rightarrow C^{p+q}(K) \quad \text{assemble to give a linear map} \quad \smile : C^*(K) \otimes C^*(K) \rightarrow C^*(K).$$

and the bilinear maps for

$$\smile : H^p(K) \times H^q(K) \rightarrow H^{p+q}(K) \quad \text{assemble to give a linear map} \quad \smile : H^*(K) \otimes H^*(K) \rightarrow H^*(K).$$

Finally, we state two well-known facts about cup products that are used throughout.

► **Theorem 5** (Commutativity [23]). $[\xi] \smile [\zeta] = [\zeta] \smile [\xi]$ for all $[\xi], [\zeta] \in H^*(K)$.

► **Theorem 6** (Functoriality of the cup product [23]). *Let $f : K \rightarrow L$ be a simplicial map and let $f^* : H^*(L) \rightarrow H^*(K)$ be the induced map on cohomology. Then, $f^*([\xi] \smile [\zeta]) = f^*([\xi]) \smile f^*([\zeta])$ for all $[\xi], [\zeta] \in H^*(K)$.*

2.3 Image persistence

The category of persistence modules is abelian since the indexing category P is small and the category of vector spaces is abelian. Thus, kernels, cokernels, and direct sums are well-defined. Persistence modules obtained as images, kernels and cokernels of morphisms were first studied in [13]. In this section, we provide a brief overview of image persistence modules.

Let C_\bullet and D_\bullet be two persistence modules of cochain complexes:

$$C_n^* \xrightarrow{\varphi_n} C_{n-1}^* \xrightarrow{\varphi_{n-1}} \dots \xrightarrow{\varphi_1} C_0^* \quad \text{and} \quad D_n^* \xrightarrow{\psi_n} D_{n-1}^* \xrightarrow{\psi_{n-1}} \dots \xrightarrow{\psi_1} D_0^*,$$

such that for $0 \leq i \leq n$ the graded vector spaces C_i^* and D_i^* (along with the respective coboundary maps) are cochain complexes, and the structure maps $\{\varphi_i : C_i^* \rightarrow C_{i-1}^* \mid i \in [n]\}$ and $\{\psi_i : D_i^* \rightarrow D_{i-1}^* \mid i \in [n]\}$ are cochain maps. Let $G_\bullet : C_\bullet \rightarrow D_\bullet$ be a *morphism of persistence modules of cochain complexes*, that is, there exists a set of cochain maps $G_i : C_i^* \rightarrow D_i^* \forall i \in \{0, \dots, n\}$, and the following diagram commutes for every $i \in [n]$.

$$\begin{array}{ccc} C_i^* & \xrightarrow{G_i} & D_i^* \\ \varphi_i \downarrow & & \downarrow \psi_i \\ C_{i-1}^* & \xrightarrow{G_{i-1}} & D_{i-1}^* \end{array}$$

Applying the cohomology functor H^* to the morphism $G_\bullet : C_\bullet \rightarrow D_\bullet$ induces another morphism of persistence modules, namely, $H^*(G_\bullet) : H^*(C_\bullet) \rightarrow H^*(D_\bullet)$. Moreover, the image $\text{im } H^*(G_\bullet)$ is a persistence module. Like any other single-parameter persistence module, an image persistence module decomposes uniquely into intervals called its *barcode* [34].

As noted in [6], a natural strategy for computing the image of $H^*(G_\bullet)$ is to write it as

$$\text{im } H^*(G_\bullet) \cong \frac{G_\bullet(Z^*(C_\bullet))}{G_\bullet(Z^*(C_\bullet)) \cap B^*(D_\bullet)},$$

where the i -th terms for the numerator and the denominator are given respectively by $(G_\bullet(Z^*(C_\bullet)))_i = G_i(Z^*(C_i))$ and $(G_\bullet(Z^*(C_\bullet)) \cap B^*(D_\bullet))_i = G_i(Z^*(C_i)) \cap B^*(D_i)$.

Tensor product image persistence. Consider the following map:

$$\smile_\bullet : C^*(K_\bullet) \otimes C^*(K_\bullet) \rightarrow C^*(K_\bullet). \tag{2}$$

Taking $G_\bullet = \smile_\bullet$ in the definition of image persistence, we get a persistence module, denoted by $\text{im } H^*(\smile K_\bullet)$, which is the same as the persistent cup module introduced in [15]. Whenever the underlying filtered complex is clear from the context, we use the shorthand notation $\text{im } H^*(\smile_\bullet)$ instead of $\text{im } H^*(\smile K_\bullet)$. Our aim is to compute its barcode denoted by $B(\text{im } H^*(\smile_\bullet))$.

2.4 Barcodes

Let K_\bullet denote a filtration on the index set $I = \{0, 1, \dots, n\}$. Assume that K_\bullet is simplex-wise, that is, $K_i \setminus K_{i-1}$ is a single simplex. Consider the persistence module H_\bullet^* obtained by applying the cohomology functor H^* on the filtration K_\bullet , that is, $H_i^* = H^*(K_i)$. The structure maps $\{\varphi_i^* : H^*(K_i) \rightarrow H^*(K_{i-1}) \mid i \in [n]\}$ for this module are induced by the cochain maps $\{\varphi_i : C^*(K_i) \rightarrow C^*(K_{i-1}) \mid i \in [n]\}$. Since K_\bullet is simplex-wise, each linear map φ_i^* is either injective with a cokernel of dimension one, or surjective with a kernel of dimension one, but not

both. Such a persistence module H_\bullet^* decomposes into interval modules supported on a unique set of intervals, namely the barcode of H_\bullet^* written as $B(H_\bullet^*) = \{(d_i, b_i] \mid b_i \geq d_i, b_i, d_i \in I\}$. Notice that since I is the indexing poset of K_\bullet , I^{op} is the indexing poset of H_\bullet^* . For $r > s$, we define $\varphi_{r,s}^* = \varphi_{s+1}^* \circ \cdots \circ \varphi_{r-1}^* \circ \varphi_r^*$ and $\varphi_{r,s} = \varphi_{s+1} \circ \cdots \circ \varphi_{r-1} \circ \varphi_r$.

► **Remark 7.** Since $\text{im } H^*(\smile_\bullet)$ is a submodule of $H^*(K_\bullet)$, the structure maps of $\text{im } H^*(\smile_\bullet)$ for every $i \in I$, namely, $\text{im } H^*(\smile_i) \rightarrow \text{im } H^*(\smile_{i-1})$ are given by restrictions of φ_i^* to $\text{im } H^*(\smile_i)$.

► **Definition 8.** For any $i \in \{0, \dots, n\}$, a nontrivial cocycle $\zeta \in Z^*(K_i)$ is said to be a product cocycle of K_i if $[\zeta] \in \text{im } H^*(\smile_i)$.

► **Proposition 9.** For a filtration K_\bullet , the birth indices (resp. death indices) of $B(\text{im } H^*(\smile_\bullet))$ are a subset of the birth indices (resp. death indices) of $B(H^*(K_\bullet))$.

Proof. Let $(d_i, b_i]$ and $(d_j, b_j]$ be (not necessarily distinct) intervals in $B(H^*(K_\bullet))$, where $b_j \geq b_i$. Let ξ_i and ξ_j be representatives for $(d_i, b_i]$ and $(d_j, b_j]$ respectively. If $\xi_i \smile \xi_j^{b_i}$ is trivial, then by the functoriality of cup product, $\varphi_{b_i, r}(\xi_i \smile \xi_j^{b_i}) = \varphi_{b_i, r}(\xi_i) \smile \varphi_{b_i, r}(\xi_j^{b_i}) = \xi_i^r \smile \xi_j^r$ is trivial $\forall r < b_i$. Writing contrapositively, if $\exists r < b_i$ for which $\xi_i^r \smile \xi_j^r$ is nontrivial, then $\xi_i \smile \xi_j^{b_i}$ is nontrivial. Noting that $\text{im } H^*(\smile_\ell)$ for any $\ell \in \{0, \dots, n\}$ is generated by $\{[\xi_i^\ell] \smile [\xi_j^\ell] \mid \xi_i, \xi_j \in \Omega_K\}$, it follows that an index b is the birth index of a bar in $B(\text{im } H^*(\smile_\bullet))$ only if it is the birth index of a bar in $B(H^*(K_\bullet))$, proving the first claim.

Let $\Omega'_{j+1} = \{[\tau_1], \dots, [\tau_k]\}$ be a basis for $\text{im } H^*(\smile_{j+1})$. Then, Ω'_{j+1} extends to a basis Ω_{j+1} of $H^*(K_{j+1})$. If j is not a death index of $B(H^*(K_\bullet))$, then $\varphi_{j+1}(\tau_1), \dots, \varphi_{j+1}(\tau_k)$ are all nontrivial and linearly independent. From Remark 7, it follows that j is not a death index of $B(\text{im } H^*(\smile_\bullet))$, proving the second claim. ◀

► **Corollary 10.** For a filtration K_\bullet , if d is a death index of $B(\text{im } H^*(\smile_\bullet))$, then at most one bar of $B(\text{im } H^*(\smile_\bullet))$ has death index d .

Proof. Using the fact that if the rank of a linear map $f : V_1 \rightarrow V_2$ is $\dim V_1 - 1$, then the rank of $f|_{W_1}$ for a subspace $W_1 \subset V_1$ is at least $\dim W_1 - 1$, from Remark 7 it follows that if $\dim H^*(K_d) = \dim H^*(K_{d+1}) - 1$, then

$$\dim(\text{im } H^*(\smile_d)) + 1 \geq \dim(\text{im } H^*(\smile_{d+1})) \geq \dim(\text{im } H^*(\smile_d)) \quad \text{proving the claim.} \quad \blacktriangleleft$$

► **Remark 11.** The persistent cup module is a submodule of the original persistence module. Let $\dim(\text{im } H_i^p)$ denote $\dim(\text{im } H^p(\smile_i))$. In the barcode $B(\text{im } H^*(\smile_\bullet))$, if $K_i = K_{i-1} \cup \{\sigma^p\}$, then either (i) $\dim(\text{im } H_i^p) > \dim(\text{im } H_{i-1}^p)$, or (ii) $\dim(\text{im } H_i^{p-1}) < \dim(\text{im } H_{i-1}^{p-1})$, or (iii) there is no change: $\dim(\text{im } H_i^p) = \dim(\text{im } H_{i-1}^p)$ and $\dim(\text{im } H_i^{p-1}) = \dim(\text{im } H_{i-1}^{p-1})$. The decrease (increase) in persistent cup modules happens only if there is a decrease (increase) in ordinary cohomology. Multiple bars of $B(\text{im } H^*(\smile_\bullet))$ may have the same birth index. But, if i is a death index, then Corollary 10 says that it is so for at most one bar in $B(\text{im } H^*(\smile_\bullet))$.

3 Algorithm: barcode of persistent cup module

Our goal is to compute the barcode of $\text{im } H^*(\smile_\bullet)$, which being an image module is a submodule of $H^*(K_\bullet)$. The vector space $\text{im } H^*(\smile_i)$ is a subspace of the cohomology vector space $H^*(K_i)$. Let us call this subspace the *cup space* of $H^*(K_i)$. Our algorithm keeps track of a basis of this cup space as it processes the filtration in the reverse order. This backward processing is needed because the structure maps between the cup spaces are induced by restrictions $\varphi_{j,i} : C^*(K_j) \rightarrow C^*(K_i)$ that are, in turn, induced by inclusions $K_j \supseteq K_i$, $i \leq j$. In particular, a cocycle/coboundary in K_j is taken to its restriction in K_i for $i \leq j$. Our

algorithm keeps track of the birth and death of the cocycle classes in the cup spaces as it proceeds through the restrictions in the reverse filtration order. We maintain a basis of nontrivial product cocycles in a matrix \mathbf{S} whose classes S form a basis for the cup spaces. In particular, cocycles in \mathbf{S} are born and die with birth and death of the elements in cup spaces.

A cocycle class from $H^*(K_i)$ may enter the cup space $\text{im } H^*(\smile_i)$ signalling a birth or may leave (become zero) the cohomology vector space and hence the cup space signalling a death. Interestingly, multiple births may happen, meaning that multiple independent cocycle classes may enter the cup space, whereas at most a single class can die because of Corollary 10. To determine which class from the cohomology vector space enters the cup space and which one leaves it, we make use of the barcode of $H^*(K_\bullet)$. However, the classes of the bases maintained in \mathbf{H} do not directly provide bases for the cup spaces. Hence, we need to compute and maintain \mathbf{S} separately, of course, with the help of \mathbf{H} .

Let us consider the case of birth first. Suppose that a cocycle ξ at degree p is born at index $k = b_i$ for $H^*(K_\bullet)$. With ξ , a set of product cocycles are born in some of the degrees $p + q$ for $q \geq 1$. To detect them, we first compute a set of candidate cocycles by taking the cup product of cocycles $\xi \smile \zeta$, for all cocycles $\zeta \in \mathbf{H}$ at b_i which can potentially augment the basis maintained in \mathbf{S} . The ones among the candidate cocycles whose classes are independent w.r.t. the current basis maintained in \mathbf{S} are determined to be born at b_i . Next, consider the case of death. A product cocycle ζ in degree r ceases to exist if it becomes linearly dependent of other product cocycles. This can happen only if the dimension of $H^r(K_\bullet)$ itself has reduced under the structure map going from $k + 1$ to k . It suffices to check if any of the nontrivial cocycles in \mathbf{S} have become linearly dependent or trivial after applying restrictions. In what follows, we use $\text{deg}(\zeta)$ to denote the degree of a cocycle ζ .

■ **Algorithm 1** CUPPERS (K_\bullet).

-
- Step 1. Compute barcode $B(\mathcal{F}) = \{(d_i, b_i)\}$ of $H^*(K_\bullet)$ with representative cocycles ξ_i ; Let $\mathbf{H} = \{\xi_i \mid [\xi_i] \text{ essential and } \text{deg}(\xi_i) > 0\}$; Initialize \mathbf{S} with the coboundary matrix ∂^\perp obtained by taking transpose of the boundary matrix ∂ ;
 - Step 2. For $k := n$ to 1 do
 - Restrict the cocycles in \mathbf{S} and \mathbf{H} to index k ;
 - Step 2.1 For every i with $k = b_i$ (k is a birth-index) and $\text{deg}(\xi_i) > 0$
 - * Step 2.1.1 If $k \neq n$, update $\mathbf{H} := [\mathbf{H} \mid \xi_i]$
 - * Step 2.1.2 For every $\xi_j \in \mathbf{H}$
 - i. If $(\zeta \leftarrow \xi_i \smile \xi_j) \neq 0$ and ζ is independent in \mathbf{S} , then $\mathbf{S} := [\mathbf{S} \mid \zeta]$ with column ζ annotated as $\zeta \cdot \text{birth} := k$ and $\zeta \cdot \text{rep@birth} := \zeta$
 - Step 2.2 If $k = d_i$ (k is a death-index) for some i and $\text{deg}(\xi_i) > 0$ then
 - * Step 2.2.1 Reduce \mathbf{S} with left-to-right column additions
 - * Step 2.2.2 If a nontrivial cocycle ζ is zeroed out, remove ζ from \mathbf{S} , generate the bar-representative pair $\{(k, \zeta \cdot \text{birth}), \zeta \cdot \text{rep@birth}\}$
 - * Step 2.2.3 Update \mathbf{H} by removing the column ξ_i
-

Algorithm CUPPERS describes this algorithm with a pseudocode. First, in Step 1, we compute the barcode of the cohomology persistence module $H^*(K_\bullet)$ along with a persistent cohomology basis. This can be achieved in $O(n^3)$ time using either the annotation algorithm [8, 19] or the pCoH algorithm [17]. The basis H is maintained with the matrix \mathbf{H} whose columns are cocycles represented as the support vectors on simplices. The matrix \mathbf{H} is initialized with all cocycles ξ_i that are computed as representatives of the bars $(d_i, b_i]$ for the module $H^*(K_\bullet)$ which get born at the first (w.r.t. reverse order) complex $K_n = K$. The matrix \mathbf{S}

is initialized with the coboundary matrix ∂^\perp with standard cochain basis. Subsequently, nontrivial cocycle vectors are added to \mathbf{S} . The classes of the nontrivial cocycles in matrix \mathbf{S} form a basis S for the cup space at any point in the course of the algorithm.

In Step 2, we process cocycles in the reverse filtration order. At each index k , we do the following. If k is a birth index for a bar $(-, b_i]$ (Step 2.1), that is, $k = b_i$ for a bar with representative ξ_i in the barcode of $H^*(K_\bullet)$, first we augment \mathbf{H} with ξ_i to keep it current as a basis for the vector space $H^*(K_k)$ (Step 2.2.1). Now, a new bar for the persistent cup module can potentially be born at k . To determine this, we take the cup product of ξ_i with all cocycles in \mathbf{H} and check if the cup product cocycle is non-trivial and is independent of the cocycles in \mathbf{S} . If so, a product cocycle is born at k that is added to \mathbf{S} (Step 2.1.2). To check this independence, we need \mathbf{S} to have current coboundary basis along with current nontrivial product cocycle basis S that are both updated with restrictions. Note that we need a for loop in Step 2.1 because at $k = n$, there can be multiple births in $H^*(K_\bullet)$.

► **Remark 12.** Restrictions in \mathbf{H} and \mathbf{S} are implemented by zeroing out the corresponding row associated to the simplex σ_i when we go from K_i to K_{i-1} and $K_i \setminus K_{i-1} = \{\sigma_i\}$.

If k is a death index (Step 2.2), potentially the class of a product cocycle from \mathbf{S} can be a linear combination of the classes of other product cocycles after \mathbf{S} has been updated with restriction. We reduce \mathbf{S} with left-to-right column additions and detect the column that is zeroed out (Step 2.2.1). If the column ζ is zeroed out, the class $[\zeta]$ dies at k and we generate a bar with death index k and birth index equal to the index when ζ was born (Step 2.2.2). Finally, we update \mathbf{H} by removing the column for ξ_i (Step 2.2.3).

3.1 Rank functions and barcodes

Let $P \subseteq \mathbb{Z}$ be a finite set with induced poset structure from \mathbb{Z} . Let $\mathbf{Int}(P)$ denote the set of all intervals in P . Recall that P^{op} denotes the opposite poset category. Given a P^{op} -indexed persistence module V_\bullet , the rank function $\text{rk}_{V_\bullet} : \mathbf{Int}(P) \rightarrow \mathbb{Z}$ assigns to each interval $I = [a, b] \in \mathbf{Int}(P)$ the rank of the linear map $V_b \rightarrow V_a$. It is well known that (see [12, 20]) the barcode of V_\bullet viewed as a function $\text{Dgm}_{V_\bullet} : \mathbf{Int}(P) \rightarrow \mathbb{Z}$ can be obtained from the rank function by the inclusion-exclusion formula:

$$\text{Dgm}_{V_\bullet}([a, b]) = \text{rk}_{V_\bullet}[a, b] - \text{rk}_{V_\bullet}[a - 1, b] + \text{rk}_{V_\bullet}[a, b + 1] - \text{rk}_{V_\bullet}[a - 1, b + 1] \quad (3)$$

To prove the correctness of Algorithm CUPPERS, we use the following elementary fact.

► **Fact 1.** A class that is born at an index $\geq b$ dies at a iff $\text{rk}_{V_\bullet}([a, b]) < \text{rk}_{V_\bullet}([a + 1, b])$.

3.2 Correctness of Algorithm CUPPERS

► **Theorem 13.** *Algorithm CUPPERS computes the barcode of the persistent cup module.*

Proof. In what follows, we abuse notation by denoting the restriction at index k of a cocycle ζ born at b also by the symbol ζ . That is, index-wise restrictions are always performed, but not always explicitly mentioned. We use $\{\xi_i\}$ to denote cocycles in the persistent cohomology basis computed in Step 1. The proof uses induction to show that for an arbitrary birth index b in $B(H^*(K_\bullet))$, if all bars for the persistent cup module with birth indices $b' > b$ are correctly computed, then the bars beginning with b are also correctly computed.

To begin with we note that in Algorithm CUPPERS, as a consequence of Proposition 9, we need to check if an index k is a birth (death) index of $B(\text{im } H^*(\smile_\bullet))$ only when it is a birth (death) index of $B(H^*(K_\bullet))$. Also, from Corollary 10, we know that at most one cycle dies at a death index of $B(\text{im } H^*(\smile_\bullet))$ (justifying Step 2.2.2).

We now introduce some notation. In what follows, we denote the persistent cup module by V_\bullet . For a birth index b , let S_b be the cup space at index b . Let C_b be the vector space of the product cocycle classes created at index b . In particular, the classes in C_b are linearly independent of classes in S_{b+1} . For a birth index $b < n$, S_b can be written as a direct sum $S_b = S_{b+1} \oplus C_b$. For index n , we set $S_n = C_n$. Then, for a birth index $b \in \{0, \dots, n\}$, C_b is a subspace of $H^*(K_b)$. C_b can be written as:

$$C_b = \begin{cases} \langle [\xi_i] \smile [\xi_j] \mid \xi_i, \xi_j \text{ are essential cocycles of } H^*(K_\bullet) \rangle & \text{if } b = n \\ \langle [\xi_i] \smile [\xi_j] \mid \xi_i \text{ is born at } b, \text{ and } \xi_j \text{ is born at an index } \geq b \rangle & \text{if } b < n \end{cases}$$

For a birth index b , let \mathbf{C}_b be the submatrix of \mathbf{S} formed by representatives whose classes generate C_b , which augments \mathbf{S} in Step 2.1.2 (i) when $k = b$ in the **for** loop. The cocycles in \mathbf{C}_b are maintained for $k \in \{b, \dots, 1\}$ via subsequent restrictions to index k . Let \mathbf{S}_b be the submatrix of \mathbf{S} containing representative product cocycles that are born at index $\geq b$. Clearly, \mathbf{C}_b is a submatrix of \mathbf{S}_b for $b < n$, and $\mathbf{C}_n = \mathbf{S}_n$.

Let DP_b be the set of filtration indices for which the cocycles in \mathbf{C}_b become successively linearly dependent to other cocycles in \mathbf{S}_b . That is, $d \in DP_b$ if and only if there exists a cocycle ζ in \mathbf{C}_b such that ζ is independent of all cocycles to its left in matrix \mathbf{S} at index $d + 1$, but ζ is either trivial or a linear combination of cocycles to its left at index d .

For the base case, we show that the death indices of the essential bars are correctly computed. First, we observe that for all $d \in DP_n$, $rk_{V_\bullet}([d, n]) = rk_{V_\bullet}([d + 1, n]) - 1$. Using Fact 1, it follows that the algorithm computes the correct barcode for $\text{im } H^*(\smile_\bullet)$ only if the indices in DP_n are the respective death indices for the essential bars. Since the leftmost columns of \mathbf{S} are coboundaries from ∂^\perp followed by cocycles from \mathbf{C}_n , and since we perform only left-to-right column additions in Step 2.2.1 to zero out cocycles in \mathbf{C}_n , the base case holds true. By (another) simple inductive argument, it follows that the computation of indices in DP_n does not depend on the specific ordering of representatives within \mathbf{C}_n .

Let $b < n$ be a birth index in $B(H^*(K_\bullet))$. For induction hypothesis, assume that for every birth index $b' > b$ the indices in $DP_{b'}$ are the respective death indices of the bars of $\text{im } H^*(\smile_\bullet)$ born at b' . By construction, the cocycles $\{\zeta_1, \zeta_2, \dots\}$ in \mathbf{S} are sequentially arranged by the following rule: If ζ_i and ζ_j are two representative product cocycles in \mathbf{S} , then $i < j$ if the birth index b_i of the interval represented by ζ_i is greater than or equal to the birth index b_j of the interval represented by ζ_j . Then, as a consequence of the induction hypothesis, for a cocycle $\zeta \in \mathbf{C}_b \setminus \mathbf{S}_b$, we assign the correct birth index to the interval represented by ζ only if ζ can be written as a linear combination of cocycles to its left in matrix \mathbf{S} .

Now, suppose that at some index $d \in DP_b$ we can write a cocycle ζ in submatrix \mathbf{C}_b as a linear combination of cocycles to its left in \mathbf{S} . For such a $d \in DP_b$, $rk_{V_\bullet}([d, b]) = rk_{V_\bullet}([d + 1, b]) - 1$. Hence, using Fact 1, a birth index $\geq b$ must be paired with d .

However, since $DP_b \cap DP_{b'} = \emptyset$ for $b < b'$, it follows from the inductive hypothesis that the only birth index that can be paired to d is b . Moreover, since we take restrictions of cocycles in \mathbf{S} , all cocycles in \mathbf{C}_b eventually become trivial or linearly dependent on cocycles to its left in \mathbf{S} . So, DP_b has the same cardinality as the number of cocycles in \mathbf{C}_b , and all the bars that are born at b must die at some index in DP_b . As a final remark, it is easy to check that the computation of indices in DP_b is independent of the specific ordering of representatives within \mathbf{S}_b by a simple inductive argument. ◀

Time complexity of CUPPERS. Let the input simplex-wise filtration have n additions and hence the complex K have n simplices. Step 1 of CUPPERS can be executed in $O(n^3)$ time using algorithms in [8, 17]. The outer loop in Step 2 runs $O(n)$ times. For each death index

in Step 2.2, we perform left-to-right column additions as done in the standard persistence algorithm to bring the matrix in reduced form. Hence, for each death index, Step 2.2 can be performed in $O(n^3)$ time. Since there are at most $O(n)$ death indices, the total cost for Step 2.2 in the course of the algorithm is $O(n^4)$.

Step 2.1 apparently incurs higher cost than Step 2.2. This is because at each birth point, we have to test the product of multiple pairs of cocycles stored in \mathbf{H} . However, we observe that there are at most $O(n^2)$ products of pairs of representative cocycles that are each computed and tested for linear independence at most once. In particular, if ξ_i and ξ_j represent $(d_i, b_i]$ and $(d_j, b_j]$ resp. with $b_i \leq b_j$, then $\xi_i \smile \xi_j$ is computed and tested for independence iff $b_i > d_j$ and the test happens at b_i . Using Equation (1), computing $\xi_i \smile \xi_j$ takes linear time. So the cost of computing the $O(n^2)$ products is $O(n^3)$. Moreover, since each independence test takes $O(n^2)$ time with the assumption that \mathbf{S} is kept reduced all the time, Step 2.1 can be implemented to run in $O(n^4)$ time over the entire algorithm.

Finally, since restrictions of cocycles in \mathbf{S} and \mathbf{H} are computed by zeroing out corresponding rows, the total time to compute restrictions over the course of the algorithm is $O(n^2)$. Combining all costs, we get an $O(n^4)$ complexity bound for CUPPERS.

4 Algorithm: barcode of persistent k-cup modules

While considering the *persistent 2-cup modules* (referred to as *persistent cup modules* in Section 3) is the natural first step, it must be noted that the invariants thus computed can still be enriched by considering *persistent k-cup modules*. As a next step, we consider image persistence of the k -fold tensor products.

Image persistence of k -fold product. Consider image persistence of the map

$$\smile_{\bullet}^k: C^*(\mathbf{K}_{\bullet}) \otimes C^*(\mathbf{K}_{\bullet}) \otimes \cdots \otimes C^*(\mathbf{K}_{\bullet}) \rightarrow C^*(\mathbf{K}_{\bullet}) \quad (4)$$

where the tensor product is taken k times. Taking $G_{\bullet} = \smile_{\bullet}^k$ in the definition of image persistence, we get the module $\text{im } H^*(\smile_{\bullet}^k)$ which is same as the persistent k -cup module introduced in [15]. Our aim is to compute $B(\text{im } H^*(\smile^k \mathbf{K}_{\bullet}))$ (written as $B(\text{im } H^*(\smile_{\bullet}^k))$ when the complex is clear from the context). Likewise, the degree-wise barcodes $B(\text{im } H^p(\smile_{\bullet}^k))$ and $B(\text{im } H^p(\smile^k))$ can also be defined and computed. We omit the details for brevity.

► **Definition 14.** For any $i \in \{0, \dots, n\}$, a nontrivial cocycle $\zeta \in Z^*(\mathbf{K}_i)$ is said to be an order- k product cocycle of \mathbf{K}_i if $[\zeta] \in \text{im } H^*(\smile_i^k)$.

4.1 Computing barcode of persistent k-cup modules

The order- k product cocycles can be viewed recursively as cup products of order- $(k-1)$ product cocycles with another cocycle. This suggests a recursive algorithm for computing the barcode of persistent k -cup module: compute the barcode of persistent $(k-1)$ -cup module recursively and then use that to compute the barcode of persistent k -cup module just like the way we computed persistent 2-cup module using the bars for ordinary persistence. In the algorithm ORDERKCUPPERS, we assume that the barcode with representatives for $H^*(\mathbf{K}_{\bullet})$ has been precomputed which is denoted by the pair of sets $(\{(d_{i,1}, b_{i,1}], \{\xi_{i,1}\})$. For simplicity, we assume that this pair is accessed by the recursive algorithm as a global variable and is not passed at each recursion level. At each recursion level k , the algorithm computes the barcode-representative pair denoted as $(\{(d_{i,k}, b_{i,k}], \{\xi_{i,k}\})$. Here, the cocycles $\xi_{i,k}$ are the initial cocycle representatives (before restrictions) for the bars $(d_{i,k}, b_{i,k}]$. At the time of their respective births $b_{i,k}$, they are stored in the field $\xi_{i,k} \cdot \text{rep@birth}$.

■ **Algorithm 2** ORDERKCUPPERS (K_\bullet, k).

-
- Step 1. If $k = 2$, return the barcode with representatives $\{(d_{i,2}, b_{i,2}], \xi_{i,2}\}$ computed by CUPPERS on K_\bullet
 else $\{(d_{i,k-1}, b_{i,k-1}], \xi_{i,k-1}\} \leftarrow \text{ORDERKCUPPERS}(K_\bullet, k-1)$
 Let $\mathbf{H} = \{\xi_{i,1} \mid [\xi_{i,1}] \text{ essential \& } \deg(\xi_{i,1}) > 0\}$; $\mathbf{R} := \{\xi_{i,k-1} \mid b_{i,k-1} = n\}$; $\mathbf{S} := \partial^\perp$;
 - Step 2. For $\ell := n$ to 1 do
 - Restrict the cocycles in \mathbf{S} , \mathbf{R} , and \mathbf{H} to index ℓ ;
 - Step 2.1 For every r s.t. $b_{r,1} = \ell \neq n$ (i.e., ℓ is a birth-index) and $\deg(\xi_{r,1}) > 0$
 - * Step 2.1.1 Update $\mathbf{H} := [\mathbf{H} \mid \xi_{r,1}]$
 - * Step 2.1.2 For every $\xi_{j,k-1} \in \mathbf{R}$
 - i. If $(\zeta \leftarrow \xi_{r,1} \smile \xi_{j,k-1}) \neq 0$ and ζ is independent in \mathbf{S} , then $\mathbf{S} := [\mathbf{S} \mid \zeta]$ with column ζ annotated as $\zeta \cdot \text{birth} := \ell$ and $\zeta \cdot \text{rep@birth} := \zeta$
 - Step 2.2 For all s such that $\ell = b_{s,k-1}$
 - * Step 2.2.1 If $\ell \neq n$, update $\mathbf{R} := [\mathbf{R} \mid \xi_{s,k-1}]$
 - * Step 2.2.2 For every $\xi_{i,1} \in \mathbf{H}$
 - i. If $(\zeta \leftarrow \xi_{s,k-1} \smile \xi_{i,1}) \neq 0$ and ζ is independent in \mathbf{S} , then $\mathbf{S} := [\mathbf{S} \mid \zeta]$ with column ζ annotated as $\zeta \cdot \text{birth} := \ell$ and $\zeta \cdot \text{rep@birth} := \zeta$
 - Step 2.3 If $\ell = d_{i,1}$ (i.e. ℓ is a death-index) and $\deg(\xi_{i,1}) > 0$ for some i then
 - * Step 2.3.1 Reduce \mathbf{S} with left-to-right column additions
 - * Step 2.3.2 If a nontrivial cocycle ζ is zeroed out, remove ζ from \mathbf{S} , generate the bar-representative pair $\{(\ell, \zeta \cdot \text{birth}), \zeta \cdot \text{rep@birth}\}$
 - * Step 2.3.3 Remove the column $\xi_{i,1}$ from \mathbf{H}
 - * Step 2.3.4 Remove the column $\xi_{j,k-1}$ from \mathbf{R} if $d_{j,k-1} = \ell$ for some j
-

A high-level pseudocode for computing the barcode of persistent k -cup module is given by algorithm ORDERKCUPPERS. The algorithm calls itself recursively to generate the sets of bar-representative pairs for the persistent $(k-1)$ -cup module. As in the case of persistent 2-cup modules, birth and death indices of order- k product cocycle classes are subsets of birth and death indices resp. of ordinary persistence. Thus, as before, at each birth index of the cohomology module, we check if the cup product of a representative cocycle (maintained in matrix \mathbf{H}) with a representative for persistent $(k-1)$ -cup module (maintained in matrix \mathbf{R}) generates a new cocycle in the barcode for persistent k -cup module (Steps 2.1.2(i), 2.2.2(i)). If so, we note this birth with the resp. cocycle (by annotating the column) and add it to the matrix \mathbf{S} that maintains a basis for live order- k product cocycles. At each death index, we check if an order- k product cocycle dies by checking if the matrix \mathbf{S} loses a rank through restriction (Step 2.3.1). If so, the cocycle in \mathbf{S} that becomes dependent to other cocycles through a matrix reduction is designated to be killed (Step 2.3.2) and we note the death of a bar in the k -cup module barcode. We update \mathbf{H} , \mathbf{R} appropriately (Steps 2.3.3, 2.3.4). At a high level, this algorithm is similar to CUPPERS with the role of \mathbf{H} played by both \mathbf{H} and \mathbf{R} as they host the cocycles whose products are to be checked during the birth and the role of \mathbf{S} in both algorithms remains the same, that is, check if a product cocycle dies or not.

Correctness and complexity of ORDERKCUPPERS. Correctness can be established the same way as for CUPPERS. See the extended version [18, Appendix B] for a sketch of the proof. For complexity, observe that we incur a cost from recursive calling in Step 1 and $O(n^4)$ cost from Step 2 with a similar analysis as for CUPPERS while noting that there are again $O(n^2)$ cocycles to be checked for independence at birth (Steps 2.1 and 2.2). Then, we get a

recurrence for time complexity as $T(n, k) = T(n, k - 1) + O(n^4)$ and $T(n, 2) = O(n^4)$ which solves to $T(n, k) = O(kn^4)$. Note that $k \leq d$, the dimension of K . This gives an $O(dn^4)$ algorithm for computing the barcodes of k -cup modules for all $k \in \{2, \dots, d\}$.

4.2 Persistent cup-length: faster computation

The *cup length* of a ring is defined as the maximum number of multiplicands that together give a nonzero product in the ring. Let \mathbf{Int}_* denote the set of all closed intervals of \mathbb{R} . Let \mathcal{F} be an \mathbb{R} -indexed filtration of simplicial complexes. The *persistent cup-length function* $\mathbf{cuplength}_\bullet : \mathbf{Int}_* \rightarrow \mathbb{N}$ is defined as a function from the set of closed intervals to the set of non-negative integers, which assigns to each interval $[a, b]$, the cup-length of the image ring $\text{im}(\mathbf{H}^*(K)[a, b])$, which is the ring $\text{im}(\mathbf{H}^*(K_b) \rightarrow \mathbf{H}^*(K_a))$.

Given a P -indexed filtration \mathcal{F} of a d -complex K of size n , let V_\bullet^k denote its k -cup module. Leveraging the fact that $\mathbf{cuplength}_\bullet([a, b]) = \text{argmax}\{k \mid \text{rk}_{V_\bullet^k}([a, b]) \neq 0\}$ (see Proposition 5.9 in [15]), the algorithm described in Section 4 can be used to compute the persistent cup-length in $O(dn^4)$ time, whereas $O(n^{d+2})$ is a coarse estimate for the runtime of the algorithm described in [14]. Thus, for $d \geq 3$, our complexity bound for computing the persistent cup length is strictly better. The details can be found in the extended version [18, Appendix A].

5 Partition modules: a more refined invariant

A partition λ_q of an integer q is a multiset of integers that sum to q , written as $\lambda_q \vdash q$. That is, a multiset $\lambda_q = \{s_1, s_2, \dots, s_\ell\}$ is a partition of q if $s_1 + s_2 + \dots + s_\ell = q$. The integers s_1, s_2, \dots, s_ℓ are non-decreasing. For every partition λ_q of q , we define a submodule $\text{im} \mathbf{H}^{\lambda_q}(\smile K_\bullet)$ (written as $\text{im} \mathbf{H}^{\lambda_q}(\smile \bullet)$) when K is clear from context) of $\text{im} \mathbf{H}^q(\smile \bullet)$:

$$\text{im} \mathbf{H}^{\lambda_q}(\smile_i) = \langle [\alpha_1] \smile [\alpha_2] \smile \dots \smile [\alpha_\ell] \mid [\alpha_j] \in \mathbf{H}^{s_j}(K_i) \text{ for } j \in [\ell] \rangle.$$

The structure map $\text{im} \mathbf{H}^{\lambda_q}(\smile_i) \rightarrow \text{im} \mathbf{H}^{\lambda_q}(\smile_{i-1})$ is the restriction of φ_i^* to $\text{im} \mathbf{H}^{\lambda_q}(\smile_i)$.

For an integer $q \geq 1$, let $\mathcal{P}(q)$ denote the number of partitions of q . In [16], Pribitkin proved that for $q \geq 1$, $\mathcal{P}(q) < \frac{e^{c\sqrt{q}}}{q^{\frac{3}{4}}}$, where $c = \pi\sqrt{2/3}$. For a d -complex K , let $\mathcal{P}^\uparrow(d)$ denote the total number of partition modules. Below, we obtain an upper bound for $\mathcal{P}^\uparrow(d)$.

$$\mathcal{P}^\uparrow(d) = \sum_{q=2}^d \mathcal{P}(q) < \sum_{q=2}^d \frac{e^{c\sqrt{q}}}{q^{\frac{3}{4}}} < d^{\frac{1}{4}} e^{c\sqrt{d}}.$$

When d is small, as is often the case in practice, $\mathcal{P}^\uparrow(d)$ is also small. For instance, $\mathcal{P}^\uparrow(2) = 1$, $\mathcal{P}^\uparrow(3) = 3$, $\mathcal{P}^\uparrow(4) = 7$.

Partition modules are more discriminative. From Remark 15 and Example 16, it follows that barcodes of partition modules are a strictly finer invariant compared to barcodes of cup modules.

► **Remark 15.** Given two filtrations K_\bullet and L_\bullet , suppose that for some ℓ and q , $\text{im} \mathbf{H}^q(\smile^\ell K_\bullet)$ and $\text{im} \mathbf{H}^q(\smile^\ell L_\bullet)$ are distinct. Without loss of generality, there exists a bar $(d, b]$ in $B(\text{im} \mathbf{H}^q(\smile K_\bullet))$ with no matching bar in $B(\text{im} \mathbf{H}^q(\smile L_\bullet))$. Let ζ be a representative for the bar $(d, b]$. Then, $[\zeta]$ can be written as $[\zeta_1] \smile [\zeta_2] \smile \dots \smile [\zeta_\ell]$ in K_b . Let s_i for each $i \in [\ell]$ denote the degree of cocycle class $[\zeta_i]$. Then, $\lambda_q = \{s_1, s_2, \dots, s_\ell\}$ is a partition of q . It follows that the bar $(d, b]$ will be present in $B(\text{im} \mathbf{H}^{\lambda_q}(\smile K_\bullet))$ but not in $B(\text{im} \mathbf{H}^{\lambda_q}(\smile L_\bullet))$.

► **Example 16.** Let $L^1 = (S^3 \times S^1) \vee S^2 \vee S^2$ and $L^2 = (S^2 \times S^2) \vee S^1 \vee S^3$. The natural cell filtrations L_\bullet^1 and L_\bullet^2 have isomorphic persistence modules and cup modules. While L_\bullet^1 has a nontrivial barcode for $\text{im} \mathbf{H}^{(3,1)}$ and a trivial barcode for $\text{im} \mathbf{H}^{(2,2)}$, the opposite is true for L_\bullet^2 .

Partition modules are not a complete invariant. Let C^1 be the 3-torus, and $C^2 = \mathbb{RP}^2 \vee \mathbb{RP}^2 \vee \mathbb{RP}^3$. The natural cell filtrations C^1_\bullet and C^2_\bullet have isomorphic persistence modules, isomorphic persistent cup modules as well as isomorphic partition modules. Yet, C^1 and C^2 have non-isomorphic cohomology algebras.

The barcodes of all partition modules of the cup product can be computed in $O(d^{\frac{1}{4}} e^{c\sqrt{d}} n^4)$ time, where $c = \pi\sqrt{2/3}$ time. Refer to the extended version [18, Section 5.1] for details.

Finally, using functoriality of the cup product, it is observed that partition modules are stable for Čech and Rips filtrations w.r.t. the interleaving distance. We state the theorem below. For a short proof, we refer the reader to the extended version [18, Section 5.2].

► **Theorem 17.** *Let $\lambda_q = \{s_1, s_2, \dots, s_\ell\}$ be a partition of an integer q . Then, for finite point sets X and Y in \mathbb{R}^d , the following identities hold true:*

$$\frac{1}{2} d_1(\text{im } H^{\lambda_q}(\smile \text{VR}_\bullet(X)), \text{im } H^{\lambda_q}(\smile \text{VR}_\bullet(Y))) \leq d_{GH}(X, Y).$$

$$\frac{1}{2} d_1(\text{im } H^{\lambda_q}(\smile \check{C}_\bullet(X)), \text{im } H^{\lambda_q}(\smile \check{C}_\bullet(Y))) \leq d_H(X, Y).$$

6 Conclusion

The cup product, the Massey products and the Steenrod operations are cohomology operations that give the cohomology vector spaces the structure of a graded ring [22, 27, 29, 31]. Recently, Lupo et al. [25] introduced invariants called Steenrod barcodes and devised algorithms for their computation, which were implemented in the software `steenroder`. Our work complements the results in Lupo et al. [25], Contessoto et al. [14] and Mémoli et al. [28]. We believe that the combined advantages of a fast algorithm and stability properties make cup modules and partition modules valuable additions to the topological data analysis pipeline.

We note that although the commonly used algorithms for computing ordinary persistence are worst case $O(n^3)$ time, in practice, on most datasets they run in nearly linear time [3, 5, 8]. Likewise, although the theoretical complexity bound for the algorithm for computing cup modules presented in this work is $O(dn^4)$ time, if implemented, we expect it to run in nearly cubic or even quadratic time on most datasets.

Finally, it would be remiss not to mention the recent application of persistent cup products for quasi-periodicity detection in sliding window embedding of time-series data [30, Section 4.5]. In fact, using extensive experimentation, in [30], Polanco shows that cup product information leads to improved quasi-periodicity detection as compared to using only ordinary persistence as in [32]. However, in [30] only the cup product of cocycle representatives of the two longest intervals in dimension 1 barcode is used. A more recent work [21] uses the persistent Künneth formula for quasiperiodicity detection. It is conceivable that cup modules could be used to improve on the quasi-periodicity detection methods described in [21, 30, 32].

References

- 1 Ulrich Bauer. Ripser: efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, 2021.
- 2 Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological methods in data analysis and visualization III*, pages 103–117. Springer, 2014.
- 3 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017.

- 4 Ulrich Bauer and Michael Lesnick. Induced matchings of barcodes and the algebraic stability of persistence. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 355–364, 2014.
- 5 Ulrich Bauer, Talha Bin Masood, Barbara Giunti, Guillaume Houry, Michael Kerber, and Abhishek Rathod. Keeping it sparse: Computing persistent homology revisited. *arXiv preprint*, 2022. [arXiv:2211.09075](https://arxiv.org/abs/2211.09075).
- 6 Ulrich Bauer and Maximilian Schmahl. Efficient Computation of Image Persistence. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:14, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2023.14.
- 7 Francisco Belchí and Anastasios Stefanou. A-infinity persistent homology estimates detailed topology from point cloud datasets. *Discrete & Computational Geometry*, pages 1–24, 2021.
- 8 Jean-Daniel Boissonnat, Tamal K Dey, and Clément Maria. The compressed annotation matrix: An efficient data structure for computing persistent cohomology. In *European Symposium on Algorithms*, pages 695–706. Springer, 2013.
- 9 Peter Brooksbank, E O’Brien, and James Wilson. Testing isomorphism of graded algebras. *Transactions of the American Mathematical Society*, 372(11):8067–8090, 2019.
- 10 Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*, volume 10. Springer, 2016.
- 11 Frédéric Chazal, Vin De Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, 2014.
- 12 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, January 2007. doi:10.1007/s00454-006-1276-5.
- 13 David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Persistent homology for kernels, images, and cokernels. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1011–1020. SIAM, 2009.
- 14 Marco Contessoto, Facundo Mémoli, Anastasios Stefanou, and Ling Zhou. Persistent cup-length. In *38th International Symposium on Computational Geometry, SoCG 2022, June 7-10, 2022, Berlin, Germany*, volume 224 of *LIPIcs*, pages 31:1–31:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 15 Marco Contessoto, Facundo Mémoli, Anastasios Stefanou, and Ling Zhou. Persistent cup-length, 2021. doi:10.48550/ARXIV.2107.01553.
- 16 Wladimir de Azevedo Pribitkin. Simple upper bounds for partition functions. *The Ramanujan Journal*, 18(1):113–119, 2009.
- 17 Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co) homology. *Inverse Problems*, 27(12):124003, 2011.
- 18 Tamal K. Dey and Abhishek Rathod. Cup product persistence and its efficient computation, 2024. [arXiv:2212.01633](https://arxiv.org/abs/2212.01633).
- 19 Tamal K. Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2022.
- 20 Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010.
- 21 Hitesh Gakhar and Jose A. Perea. Sliding window persistence of quasiperiodic functions. *Journal of Applied and Computational Topology*, 8(1):55–92, 2024.
- 22 Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, 2002.
- 23 Jean-Claude Hausmann. *Mod two homology and cohomology*, volume 10. Springer, 2014.
- 24 Estanislao Herscovich. A higher homotopic extension of persistent (co)homology. *Journal of Homotopy and Related Structures*, 13(3):599–633, 2018.
- 25 Umberto Lupo, Anibal M. Medina-Mardones, and Guillaume Tausin. Persistence Steenrod modules. *Journal of Applied and Computational Topology*, 6(4):475–502, 2022.

- 26 Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The Gudhi library: Simplicial complexes and persistent homology. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 167–174, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- 27 William S. Massey. Higher order linking numbers. *Journal of Knot Theory and Its Ramifications*, 7:393–414, 1998.
- 28 Facundo Mémoli, Anastasios Stefanou, and Ling Zhou. Persistent cup product structures and related invariants. *Journal of Applied and Computational Topology*, 2023.
- 29 Robert E. Mosher and Martin C. Tangora. *Cohomology operations and applications in homotopy theory*. Courier Corporation, 2008.
- 30 Luis Polanco. *Applications of persistent cohomology to dimensionality reduction and classification problems*. Phd thesis, Michigan State University, 2022. doi:doi:10.25335/exk0-fs44.
- 31 Norman E Steenrod. Products of cocycles and extensions of mappings. *Annals of Mathematics*, pages 290–320, 1947.
- 32 Christopher J. Tralie and Jose A. Perea. (quasi)periodicity quantification in video data, using topology. *SIAM Journal on Imaging Sciences*, 11(2):1049–1077, 2018.
- 33 Andrew Yarmola. *Persistence and computation of the cup product*. Undergraduate honors thesis, Stanford University, 2010.
- 34 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 347–356, 2004.