SAND2017-12740J

Journal of Applied and Computational Topology manuscript No.
(will be inserted by the editor)

The Computational Complexity of Multidimensional Persistence

Jacek Skryzalin · Pawin Vongmasa

Received: date / Accepted: date

Abstract We present findings on the computational complexity of computing multidimensional persistent homology. We first show that the worst-case computational complexity of multidimensional persistence is exponential. We then present an algorithm for computing multidimensional persistence which extends the algorithm given by Zomorodian and Carlsson for computing one-dimensional persistence [14]. The computational complexity of our algorithm is polynomial in the size of the persistence module and exponential in the persistence dimension.

Keywords multidimensional persistent homology · computational complexity · applied topology

1 Introduction

Over the past decade, developments to the theory of persistent homology [1], [2] have allowed scientists and analysts to apply topological techniques to the study of data (see [3], [7], [9], [13], for example). However, almost all concrete applications of persistent homology have used one-dimensional persistence. We believe that this is the case for two main reasons. First, one-dimensional persistence can be computed in $O(n^3)$ time [14], where n represents the number of generators of the underlying (filtered) complex. Second, the output of one-dimensional persistence has a pleasingly intuitive interpretation in terms of the "birth" and "death" times of geometrical features of the complex. Both of these attractive qualities of one-dimensional persistence arise from the equivalence between the category of one-dimensional persistence modules and

Jacek Skryzalin
Sandia National Laboratories
Tel.: 1-505-284-9487
E-mail: jskryza@sandia.gov

Pawin Vongmasa
Google, Inc.
E-mail: pawin.vongmasa@gmail.com

the category of finitely presented graded modules over a PID, and from the structure theorem of such modules (see [8], for example).

Unfortunately, there is no analogous structure theorem for the category of multidimensional persistence modules. As such, the calculation of a multidimensional persistence module and its subsequent analysis must be considered as two related but separate problems. We build off the approach of Carlsson, Singh, and Zomorodian [4], which rephrases the calculation of the homology of a multifiltered complex in terms of a problem in computational commutative algebra. We first provide a family of complexes whose homology requires exponential time to compute in this computational commutative algebra setting. This disproves a theorem of [4]. We then provide an algorithm for the computation of the homology of a multifiltered complex whose runtime is (approximately) $O(n^{d+1})$, where d is the persistence dimension and n is the number of generators of the multi-filtered complex underlying the persistence module. We note that our algorithm is highly parallelizeable; with infinite parallelism available, our algorithm's runtime would be (approximately) $O(d^2 n^2 + dn^4)$.

2 Algebraic Preliminaries

Throughout this paper, let k denote a field of arbitrary characteristic. We denote elements of $\mathbb{N}^d$ using bold letters (e.g., $\mathbf{a} \in \mathbb{N}^d$), although we represent elements of $\mathbb{N}$ using non-bold, italic letters (e.g., $a \in \mathbb{N}$). Furthermore, it is understood that the ith coordinate of $\mathbf{a} \in \mathbb{N}^d$ is a_i. We impose two orderings on $\mathbb{N}^d$. First, for $\mathbf{a}, \mathbf{b} \in \mathbb{N}^d$, we say that $\mathbf{a} \leq \mathbf{b}$ if $a_i \leq b_i$ for all i. Second, we say that $\mathbf{a} \leq_\ell \mathbf{b}$ if $\mathbf{a}$ precedes $\mathbf{b}$ in the lexicographic ordering on $\mathbb{N}^d$.

We let $k[x_1, ...x_d]$ denote the polynomial ring in d variables over k, and we denote the element $x_1^{a_1} x_2^{a_2} \dots x_d^{a_d}$ by $\mathbf{x}^\mathbf{a}$. Given any ordering $\leq$ on $\mathbb{N}^d$, we may extend the ordering to monomials in $k[\mathbf{x}]$ by declaring that $\mathbf{x}^\mathbf{a} \leq \mathbf{x}^\mathbf{b}$ if and only if $\mathbf{a} \leq \mathbf{b}$. We may further extend this ordering to the free module $k[\mathbf{x}]^m$ which is freely generated by the standard basis elements e_i (ordered so that $e_i \geq e_j$ if $i \leq j$) by declaring that $\mathbf{x}^\mathbf{a}\, e_i \leq \mathbf{x}^\mathbf{b}\, e_j$ if and only if (a) $i \geq j$ or (b) $i = j$ and $\mathbf{a} \leq \mathbf{b}$.

Given any element $f \in k[\mathbf{x}]^m$, we may write f as a sum of *terms*, each of the form $c\,\mathbf{x}^\mathbf{u}\, e_i$, where $c \in k$ is nonzero. The greatest such summand $c\,\mathbf{x}^\mathbf{u}\, e_i$ under any total order (we'll be using the lexicographic ordering) is called the *leading term*. The element $c \in k$ is called the *leading coefficient*, and $\mathbf{x}^\mathbf{u}\, e_i$ is called the *leading monomial*. We now very briefly introduce the notion of a Gröbner basis. More details can be found in [6].

Definition 1 Let M be a submodule of the free module $k[x_1, ..., x_d]^m$. Assume that $k[x_1, ..., x_d]^m$ is freely generated by the standard basis elements e_i. A *Gröbner basis* for M with respect to a total order $\leq$ on $k[x_1, ..., x_d]^m$ is a set $GB(M)$ of generators of M such that the leading terms of elements in $GB(M)$ generate the $k[\mathbf{x}]$-module consisting of all leading terms of all elements of M. A Gröbner basis G is *minimal* if no leading term in any element of the basis is in the ideal generated by the leading terms of the other elements of the basis. A Gröbner basis G is *reduced* if G is minimal, the leading coefficient of each element of G is 1, and no non-leading monomial of any element in G is divisible by any leading monomial of any element of G.

Gröbner bases enjoy a number of attractive properties. For example, for a fixed monomial ordering, all minimal Gröbner bases for a module M have the same cardinality, and M has a unique reduced Gröbner basis. Furthermore,

Lemma 1 *Let $M \subseteq k[\mathbf{x}]^m$. Assume that we have fixed an ordering on monomials in $k[\mathbf{x}]^m$. Then the following are equivalent:*

- *G is a Gröbner basis of M.*
- *The leading term of any $f \in M$ is divisible by some $g \in G$.*
- *The division of any polynomial in $k[\mathbf{x}]^m$ by G gives a unique remainder.*
- *For $f \in M$, the division of f by G yields* 0.

Remark 1 In general, given some polynomial module M, one can construct a generating set G and an element $f \in M$ such that reducing f modulo G does not yield 0. This fact suggests that whenever we wish to test membership in a complicated polynomial ring or module, it is imperative to work with Gröbner bases.

Remark 2 Definition 1 and Lemma 1 depart somewhat from the standard definition of Gröbner basis, which is typically defined for ideals of a polynomial ring $k[x_1, ..., x_d]$ rather than for submodules of $k[x_1, ..., x_d]^m$. We note that we may identify submodules M of $k[x_1, ..., x_d]^m$ with ideals I of $k[x_1, ..., x_d, e_1, ..., e_m]/\langle e_i e_j \rangle_{i \leq j}$ which are contained in the ideal $\langle e_i \rangle_{1 \leq i \leq m}$. Because Gröbner bases are typically given for ideals of polynomial rings (rather than any of their quotients), we may compute a Gröbner basis for a module $M \subseteq k[\mathbf{x}]^m$ by computing a Gröbner basis G for the inverse image of the corresponding ideal I in $k[x_1, ..., x_d, e_1, ..., e_m]$ and subsequently removing all monomial summands contained in the ideal $\langle e_i e_j \rangle_{i \leq j}$ from all generators in G.

3 A Review of Multidimensional Persistence

In this section, we discuss the theory of multidimensional persistence [5] and outline the established approach to computing multidimensional persistence [4].

Definition 2 A *persistence module M* indexed by the partially ordered set V is a family of k-modules $\{M_\mathbf{v}\}_{\mathbf{v} \in V}$ together with homomorphisms $\phi_{\mathbf{u},\mathbf{v}} : M_\mathbf{u} \to M_\mathbf{v}$ for all $\mathbf{u} \leq \mathbf{v}$, such that $\phi_{\mathbf{v},\mathbf{w}} \circ \phi_{\mathbf{u},\mathbf{v}} = \phi_{\mathbf{u},\mathbf{w}}$ whenever $\mathbf{u} \leq \mathbf{v} \leq \mathbf{w}$.

In this work, we are concerned specifically with the case $V = \mathbb{N}^d$ for some integer $d > 1$.

Definition 3 A *multidimensional persistence module M* is a persistence module whose underlying indexing set V is $\mathbb{N}^d$ for some $d > 1$. We say that d is the *persistence dimension* of M.

We now explain how multidimensional persistence arises naturally by taking the homology of a multifiltered complex. We briefly review definitions and refer the reader to [12] for a more in-depth discussion of the underlying theory.

Definition 4 Let V be a set. For a nonnegative integer i, an *i-simplex* is defined to be an ordered list $[v_0, v_1, ..., v_i]$ of elements in V. Moreover, for $j < i$, we say that a j-simplex σ' is a *face* of an i-simplex σ if σ' can be obtained by removing some number of elements from the ordered list describing σ.

Furthermore, a *simplicial complex* X is a set of simplices such that:

- any face of any simplex in X is also a simplex in X, and
- the intersection of two simplices σ and σ' in X is either the empty list or a face of both σ and σ'.

Example 1 In topological data analysis, one is often given a finite set $S \subseteq \mathbb{R}^n$ of data points. We assume that a (possibly arbitrary) total order has been imposed on this set of data points. There are many established methods for constructing complexes from data [1]. One popular choice is the *Vietoris-Rips* construction. Given some real number $\epsilon > 0$, we define the Vietoris-Rips complex $VR(X; \epsilon)$ with scale parameter ϵ as the simplicial complex with vertex set S, where $[s_0, s_1, ..., s_i]$ spans an i-simplex if and only if $s_0 \le s_1 \le \cdots \le s_i$ in the total order on S and $d(s_a, s_b) < \epsilon$ for all $0 \le a, b \le i$.

Definition 5 For a field k and simplicial complex X, the group $C_i(X; k)$ of *i-chains* of X is defined to be the free k-module generated by the i-simplices of X. When the field k is understood, we abbreviate $C_i(X; k)$ as $C_i(X)$. We define the *boundary* of the i-simplex $[v_0, v_1, ..., v_i]$ to be the formal sum

$$\partial\left([v_0, v_1, ..., v_i]\right) = \sum_{j=0}^{i} (-1)^j [v_0, v_1, ..., v_{j-1}, \widehat{v_j}, v_{j+1}, ..., v_i] \in C_{i-1}(X),$$

where $\widehat{v_j}$ denotes that v_j has been removed from the ordered list $[v_0, v_1, ..., v_i]$. The boundary map thus defines a homomorphism

$$\partial_i : C_i(X) \to C_{i-1}(X).$$

We denote the group $\ker(\partial_i) \subseteq C_i(X)$ of *i-cycles* of X by $Z_i(X)$, and we denote the group $\operatorname{im}(\partial_{i+1}) \subseteq C_i(X)$ of *i-boundaries* by $B_i(X)$. Because $\partial_i \circ \partial_{i+1} = 0$, we may define the ith *homology* group $H_i(X)$ of a simplicial complex X as the quotient

$$H_i(X) = \frac{Z_i(X)}{B_i(X)} = \frac{\ker(\partial_i : C_i(X) \to C_{i-1}(X))}{\operatorname{im}(\partial_{i+1} : C_{i+1}(X) \to C_i(X))}.$$

Definition 6 A d-filtered simplicial complex X is a collection of simplicial complexes $\{X_{\mathbf{v}}\}_{\mathbf{v} \in \mathbb{N}^d}$ such that for $\mathbf{u}, \mathbf{v} \in \mathbb{N}^d$ such that $\mathbf{u} \le \mathbf{v}$, we have that $X_{\mathbf{u}} \subseteq X_{\mathbf{v}}$.

For $\mathbf{u}, \mathbf{v} \in \mathbb{N}^d$, the inclusion map $X_{\mathbf{u}} \hookrightarrow X_{\mathbf{v}}$ descends to a map on homology $H_i(X_{\mathbf{u}}) \to H_i(X_{\mathbf{v}})$. We may thus form a d-dimensional persistence module $H_i(X) = \{H_i(X_{\mathbf{v}})\}_{\mathbf{v} \in \mathbb{N}^d}$ from any d-filtered simplicial complex X by applying the functor $H_i(-)$.

Example 2 We show how to extend the definition of the Vietoris-Rips complex in Example 1 to obtain a multifiltered simplicial complex. We assume that our input data $S \subseteq \mathbb{R}^n$ has been equipped with $d-1$ *filter* functions $f_i : S \to \mathbb{N}$, which we collectively represent as $\mathbf{f}$. For $\mathbf{u} \in \mathbb{R}^{d-1}$, let $VR(X; \epsilon; \mathbf{u})$ denote the subcomplex of $VR(X; \epsilon)$ such that:

- the set of vertices of $VR(X;\epsilon;\mathbf{u})$ is $\{s \in S \mid \mathbf{f}(s) \le \mathbf{u}\}$, and
- vertices $s_0, ..., s_i$ of $VR(X;\epsilon;\mathbf{u})$ span an i-simplex in $VR(X;\epsilon;\mathbf{u})$ if and only if these vertices span an i-simplex in $VR(X;\epsilon)$.

We now note for $\epsilon_1 \le \epsilon_2$ and $\mathbf{u}_1 \le \mathbf{u}_2$, there is an inclusion

$$VR(X;\epsilon_1;\mathbf{u}_1) \hookrightarrow VR(X;\epsilon_2;\mathbf{u}_2).$$

The set of permissible values for ϵ and $\mathbf{u}$ is isomorphic to $\mathbb{R}^d$. By choosing an appropriate partially ordered subset V of these permissible values such that V is isomorphic as a partially ordered set to $\mathbb{N}^d$, we may obtain a d-filtered simplicial complex. Taking the homology of this complex leads to the notion of *level-set persistence*.

The study of multidimensional persistence stems from the following fact:

Theorem 1 ([5]) *For any multidimensional persistence module M indexed by $\mathbb{N}^d$, we can define a d-graded $k[\mathbf{x}]$-module $\alpha(M)$ via*

$$\alpha(M) = \bigoplus_{\mathbf{v}\in\mathbb{N}^d} M_{\mathbf{v}},$$

where for $\mathbf{u} \le \mathbf{v}$, we define $\mathbf{x}^{\mathbf{v}-\mathbf{u}} : M_{\mathbf{u}} \to M_{\mathbf{v}}$ via the homomorphism $\phi_{\mathbf{u},\mathbf{v}}$. Furthermore, α defines an equivalence of categories between the category of finite persistence modules over k indexed by $\mathbb{N}^d$ and the category of finitely presented d-graded modules over $k[x_1, ..., x_d]$.

The graded version of the structure theorem for finitely generated modules over a PID allows one to understand any one-dimensional (i.e., indexed over $\mathbb{N}$) persistence module in terms of a finite set of intervals in $\mathbb{N}$. This correspondence provides a unique and intuitive geometric depiction of any one-dimensional persistence module.

Unfortunately, there is no satisfactory analogue for the multidimensional case, nor does there exist *any* discrete invariant (one which assigns to any module a point in an algebraic variety in a way that does not depend on k) that is complete for the category of multidimensional persistence modules [5]. Although the problem of extracting useful information from multidimensional persistence modules is currently an active area of research, we nonetheless can identify one basic quantity which we should be able to compute:

Definition 7 Let M be a persistence module indexed over the partially ordered set V. For $\mathbf{u}, \mathbf{v} \in V$, the *rank* $\rho_{\mathbf{u},\mathbf{v}}(M)$ of M is defined as the rank of the k-linear map $M_{\mathbf{u}} \to M_{\mathbf{v}}$.

We now describe the approach of Carlsson, Singh, and Zomorodian [4] for computing multidimensional persistent homology. We note that our exposition differs slightly from that of [4], but the general ideas remain the same. As input we assume that we are given the following:

- A d-filtered complex X consisting of a set $X^{(i)}$ of i-simplices for each $i \in \mathbb{N}$.
- For each i, a total order on the set $X^{(i)}$.

– For each simplex $\sigma \in X$, we are given the set

$$Birth(\sigma) = \left\{ \mathbf{u} \in \mathbb{N}^d \,\middle|\, \sigma \in X_{\mathbf{u}} \text{ and } \sigma \notin X_{\mathbf{v}} \text{ for every } \mathbf{v} < \mathbf{u} \right\}$$

of gradings $\mathbf{u} \in \mathbb{N}^d$ for each simplex which represent the gradings at which the simplex first appears.

Remark 3 We note that our setting is strictly more general than that of [4], in which the complex X is assumed to be *one-critical* (i.e., the sets $Birth(\sigma)$ are required to have cardinality one). We further note that the extended Vietoris-Rips complex presented in Example 2 is one-critical.

We represent the boundary maps ∂_i of the d-filtered complex X as matrices M_i with entries in the polynomial ring $k[x_1, ..., x_d]$. The columns of M_i are indexed by pairs $(\sigma, \mathbf{u})$, where $\sigma \in X^{(i)}$ and $\mathbf{u} \in Birth(\sigma)$. The rows of M_i are indexed by elements $\tau \in X^{(i-1)}$. Furthermore, the rows of M should be ordered in decreasing order of their corresponding indices. The entry of M_i corresponding to column $(\sigma, \mathbf{u})$ and row τ is the product of $\mathbf{x}^{\mathbf{u}}$ and the coefficient of τ in $\partial_i(\sigma)$. We extend the given total order $\leq$ on $X^{(i)}$ to a total order on $X^{(i)} \times \mathbb{N}^d$ by declaring that $(\sigma, \mathbf{u}) \geq (\tau, \mathbf{v})$ if and only if (a) $\sigma \geq \tau$ or (b) $\sigma = \tau$ and $\mathbf{u} \geq_\ell \mathbf{v}$.

To compute $H_i(X)$, we must first "calculate" $Z_i(X)$ (the nullspace of M_i) and $B_i(X)$ (the columnspace of M_{i+1}). If M_i has m rows and n columns, then $B_{i-1}(X)$ is a submodule of the free module $k[\mathbf{x}]^m$ and $Z_i(X)$ is a submodule of the free module $k[\mathbf{x}]^n$. The authors of [4] argue that the proper definition of "calculate" in this setting is "compute a Gröbner basis for". Indeed, after calculating Gröbner bases for $Z_i(X)$ and $B_i(X)$, it is easy to calculate their quotient $H_i(X_{\mathbf{v}})$ for any $\mathbf{v} \in \mathbb{N}^d$ [6].

Carlsson, Singh, and Zomorodian [4] recommend Buchberger's algorithm and Schreyer's algorithm [6] for these computations. We first provide a lower bound to the worst-case runtime of any algorithm which calculates a Gröbner basis for $B_i(X)$ and $Z_i(X)$. We then provide an alternative algorithm for computing multidimensional persistence.

4 A Lower Bound on the Complexity of Computing Multidimensional Persistence

In this section we provide a counterexample to the claim of [4] that multidimensional persistence can be computed in polynomial time.

Example 3 Let n be an even positive integer. We construct an n-filtered complex $X = X(n)$. Let $\tau_0, \tau_1, ..., \tau_{\frac{n}{2}}$ be a set of $\frac{n}{2} + 1$ 0-simplices ordered so that $\tau_i > \tau_j$ if $i < j$. Assume that $Birth(\tau_j) = 0 \in \mathbb{N}^n$ for all j. Further assume that we have $\frac{n}{2}$ 1-simplices $\sigma_1, \sigma_2, ..., \sigma_{\frac{n}{2}}$ such that $Birth(\sigma_j) = \{e_{2j-1}, e_{2j}\}$, where e_j denotes the jth standard basis vector in $\mathbb{N}^n$. Set

$$\partial(\sigma_j) = \begin{cases} \tau_0 - \tau_1 & j = 1 \\ \tau_{j-2} - \tau_j & j > 1 \end{cases}.$$

The matrix representation of ∂_1 is given below. We assume that the column labeled e_i below is understood as $\left(\sigma_{\lceil \frac{i}{2} \rceil}, e_i\right)$.

$$\left[\begin{array}{c||cccccccc}
 & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & \cdots & e_{n-1} & e_n \\
\hline\hline
\tau_0 & x_1 & x_2 & x_3 & x_4 & 0 & 0 & \cdots & 0 & 0 \\
\tau_1 & -x_1 & -x_2 & 0 & 0 & x_5 & x_6 & \cdots & 0 & 0 \\
\tau_2 & 0 & 0 & -x_3 & -x_4 & 0 & 0 & \cdots & 0 & 0 \\
\tau_3 & 0 & 0 & 0 & 0 & -x_5 & -x_6 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\tau_{\frac{n}{2}-2} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & x_{n-1} & x_n \\
\tau_{\frac{n}{2}-1} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\tau_{\frac{n}{2}} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -x_{n-1} & -x_n
\end{array}\right]$$

Theorem 2 *Let $X(n)$ be the n-filtered complex of Example 3. Any minimal Gröbner basis for $B_0(X(n))$ contains $2^{\frac{n}{2}+1} + n - 4$ elements.*

Proof We prove Theorem 2 by induction on the following statements:

1. Any minimal Gröbner basis for $B_0(X(n))$ contains $2^{\frac{n}{2}+1} + n - 4$ elements.
2. Any leading monomial $\mathbf{x}^{\mathbf{a}}\,\tau_j$ of an element in a minimal Gröbner basis for $B_0(X(n))$ satisfies $j \leq \frac{n}{2} - 1$.
3. There are $2^{n/2}$ elements L_n in any minimal Gröbner basis whose leading monomial is a multiple of $\tau_{\frac{n}{2}-1}$. They are precisely the elements of the form
$$\mathbf{x}^{\mathbf{a}}\,\tau_{\frac{n}{2}-1} - \mathbf{x}^{\mathbf{a}}\,\tau_{\frac{n}{2}},$$
where
$$\mathbf{a} \in \left\{ \{0,1\}^n \;\middle|\; a_{2j-1} + a_{2j} = 1 \text{ for all } 1 \leq j \leq \frac{n}{2} \right\}.$$

For $n = 2$, the three inductive hypotheses follow from the statement that the set $\{\partial(\sigma_1), \partial(\sigma_2)\}$ is itself a Gröbner basis for $B_0(X(2))$.

Now assume that $n > 2$ is even and that $\tilde{G}$ is a minimal Gröbner basis for $B_0(X(n-2))$. By inductive hypotheses (2) and (3), a minimal Gröbner basis G for $B_0(X(n))$ can be obtained from $\tilde{G}$ by the insertion $\partial(\sigma_{n-1})$, $\partial(\sigma_n)$, and the result of the reduction of each of $\partial(\sigma_{n-1})$ and $\partial(\sigma_n)$ with each of the elements in L_{n-2}. The reduction of each of $\partial(\sigma_{n-1})$ and $\partial(\sigma_n)$ with the elements in L_{n-2} produces the set L_n, and
$$|L_n| = |\{\partial(\sigma_{n-1}), \partial(\sigma_n)\}| * |L_{n-2}| = 2 * 2^{\frac{n-2}{2}} = 2^{\frac{n}{2}}.$$
By inspection, we see that (2) still holds and that the elements of L_n are of the required form, proving (3). Finally, (1) holds as well, since G is obtained from $\tilde{G}$ by the addition of $2 + 2^{\frac{n}{2}}$ elements (i.e., 2 elements from $\partial(\sigma_{n-1})$ and $\partial(\sigma_n)$ and $2^{\frac{n}{2}}$ from L_n).

Remark 4 In Example 3, we provide an example of a multifiltered complex X such that the Gröbner basis for $B_0(X)$ has cardinality exponential in the number of 1-simplices. We now explain how to generalize Example 3 to provide a multifiltered complex such that the Gröbner basis for $B_i(X)$ has cardinality exponential in the number of $(i+1)$-simplices for $i > 0$.

For an even integer n, we construct another n-filtered complex X with i-simplices $\tau_0, \tau_1, ..., \tau_{\frac{n}{2}}, \upsilon_1, ..., \upsilon_i$, where $Birth(\tau_j) = Birth(\upsilon_j) = 0$. Assume that $\tau_i > \tau_j$ for $i < j$, that $\upsilon_i > \upsilon_j$ for $i < j$, and that $\tau_i > \upsilon_j$ for all i and j. Let $\sigma_1, ..., \sigma_{\frac{n}{2}}$ be a set of $(i+1)$-simplices with $Birth(\sigma_j) = \{e_{2j-1}, e_j\}$. We set

$$\partial(\sigma_j) = \begin{cases} \tau_0 - \tau_1 + \sum_{k=1}^{i} \upsilon_k & j \in \{1, 2\} \\ \tau_{j-2} - \tau_j + \sum_{k=1}^{i} \upsilon_k & j > 2 \end{cases}.$$

A similar argument to that in Theorem 2 shows that $GB(B_i(X))$ has cardinality exponential in the number of $(i+1)$-simplices.

Corollary 1 *Let $i \in \mathbb{N}$, let X be a multifiltered complex, and let*

$$n = \sum_{\sigma \in X^{(i+1)}} |\{Birth(\sigma)\}|.$$

Then the worst-case computational complexity for computing $GB(B_i(X))$ is $\Omega\left(2^{\frac{n}{2}}\right)$.

Remark 5 The bound of Corollary 1 contradicts Lemma 4 (and thus Theorem 3) of [4]. That is, multidimensional persistence cannot always be computed in time polynomial in the number of simplices of the complex.

Remark 6 It should be noted that the multifiltered complexes in Example 3 and Remark 4 are not one-critical. We explain how to construct a one-critical complex $X(n)$ with $\Theta(n)$ 1-simplices such that $GB(B_0(X(n)))$ grows exponentially in n. Assume that we have 0-simplices $\kappa_1, \kappa_2, ..., \kappa_{\frac{n}{2}}, \tau_0, \tau_1, ..., \tau_{\frac{n}{2}}$ (arranged in decreasing order) such that $Birth(\kappa_j) = Birth(\tau_j) = 0$ for all j.

Further assume that we have 1-simplices $\sigma_j, \lambda_j, \mu_j$ (for $1 \le j \le \frac{n}{2}$) such that $Birth(\sigma_j) = \{e_{2j-1}\}$ and $Birth(\lambda_j) = Birth(\mu_j) = \{e_{2j}\}$. Set

$$\partial(\sigma_j) = \begin{cases} \tau_0 - \tau_1 & j = 1 \\ \tau_{j-2} - \tau_j & j > 1 \end{cases}.$$

$$\partial(\lambda_j) = \begin{cases} \kappa_j - \tau_0 & j = 1 \\ \kappa_j - \tau_{j-2} & j > 1 \end{cases}.$$

$$\partial(\mu_j) = \kappa_j - \tau_j.$$

Using an argument similar to that in the proof of Theorem 2, one can show that the size of $GB(B_0(X(n)))$ grows exponentially in n (although slower than in the family of complexes introduced in Example 3). Furthermore, as in Remark 4, one can construct complexes $X(n)$ with $\Theta(n)$ $(i+1)$-simplices such that $GB(B_i(X(n)))$ grows exponentially in n.

We now provide a family $X(n)$ of simplicial complexes such that the size of the Gröbner basis for $Z_i(X(n))$ grows exponentially.

Example 4 In this example, we construct a family of multifiltered complexes $X(n)$ with $\Theta(n)$ i-simplices such that the size of $GB(Z_i(X(n)))$ grows exponentially in n. Let $m \geq i+2$ be a positive integer, and let $\sigma_1, ..., \sigma_m$ be a set of i-simplices with $Birth(\sigma_j) = \{e_{2j-1}, e_{2j}\}$ such that the set $\{\sigma_1, ..., \sigma_m\}$ is a triangulation of the i-sphere S^i.

Theorem 3 *Let i be a positive integer, let $X(n)$ be the multifiltered complex constructed in Example 4, and let $n = \sum_{\sigma \in X^{(i)}} |Birth(\sigma)|$. Then $GB(Z_i(X(n)))$ contains $\Omega\left(2^{\frac{n}{2}}\right)$ elements.*

Proof For any

$$\mathbf{a} \in S = \left\{ \{0,1\}^n \;\middle|\; a_{2j-1} + a_{2j} = 1 \text{ for all } 1 \leq j \leq n \right\},$$

$\dim(H_i(X(n)_{\mathbf{a}})) = 1$. However, for any $\mathbf{a}'$ strictly less than all $\mathbf{a} \in S$ under the partial order on $\mathbb{N}^n$, we have that $\dim(H_i(X(n)_{\mathbf{a}'})) = 0$. Since $|S| = 2^{\frac{n}{2}}$, we have that the size of $GB(Z_i(X(n)))$ is $\Omega\left(2^{\frac{n}{2}}\right)$.

Corollary 2 *Let $i \in \mathbb{N}$, let X be a multifiltered complex, and let*

$$n = \sum_{\sigma \in X^{(i)}} |\{Birth(\sigma)\}| .$$

Then the worst-case computational complexity for computing $GB(Z_i(X))$ is $\Omega\left(2^{\frac{n}{2}}\right)$.

Remark 7 Note that the complexes $X(n)$ in Example 4 are not one-critical. One can use a technique similar to that of Remark 6 to construct a one-critical simplicial complex $X(n)$ with n i-simplices (for $n \geq (i+2)^2$) such that $GB(Z_i(X(n)))$ has $\Omega\left(2^{\frac{n}{i+2}}\right)$ elements.

Remark 8 We have assumed in this paper that we are given a total order on the set $X^{(i)}$ of i-simplices. We note that if we reverse the order of 0-simplices in Example 3 so that $\tau_i < \tau_j$ whenever $i < j$, then the columns of the resulting boundary matrix serve as a minimal Gröbner basis for $B_0(X)$. In this case, the minimal Gröbner basis for $B_0(X)$ now contains exactly n elements. We leave as open the question of whether it is possible to calculate efficiently an ordering on the simplices of a complex in order to obtain a polynomially-sized Gröbner basis for the various cycle and boundary modules of a multifiltered complex.

5 Fast Computation of Multidimensional Persistence: Our Approach

We now provide and analyze an algorithm for computing multidimensional persistence. This algorithm extends the algorithm given in [14] for computing one-dimensional persistence. We note that our algorithm is inspired by Faugère's improvements [10], [11] to Buchberger's algorithm; indeed, it would not be wholly inaccurate to view the algorithms we provide as adaptations of Faugère's F4 and F5 algorithms to the computation of multidimensional persistent homology. Additionally, we show how our approach allows us to derive bounds on the complexity of computing multidimensional persistence.

Our main insight is that despite Corollaries 1 and 2, if we fix a persistence dimension, we can compute Gröbner bases for $Z_i(X)$ and $B_i(X)$ in time polynomial in the number of simplices of X. Although the runtime of our algorithm is exponential in the persistence dimension, our algorithm nevertheless has utility in the wild, where high persistence dimensions are unlikely to be encountered. The results of [14] show that it is possible to compute persistent homology for $d = 1$ in time $O(n^3)$, where n is the number of simplices of our complex X. Our results show that, surprisingly, it is also possible to compute persistent homology in time $O(n^3)$ for $d = 2$.

Our algorithm heavily exploits the multigraded structure of multidimensional persistence modules in order to reduce the computation of multidimensional persistence to the one-dimensional setting. We now provide theoretical results which will serve as the basis for our algorithm.

Lemma 2 *Let $M = \bigoplus_{\mathbf{v}\in\mathbb{N}^d} M_{\mathbf{v}}$ be a d-graded submodule of $k[x_1,\dots,x_d]^m$. Let $GB(M_{\mathbf{v}})$ denote a Gröbner basis for the k-module $M_{\mathbf{v}}$. Then*

$$\bigcup_{\mathbf{v}\in\mathbb{N}^d} GB(M_{\mathbf{v}})$$

is a Gröbner basis for M.

Proof The proof is a routine check of the properties of a Gröbner basis.

Remark 9 A Gröbner basis for any k-module V with respect to an ordered basis $\{e_j\}$ can be constructed from a set of generators by converting into column echelon form the matrix M whose rows are indexed by the e_j and whose columns correspond to generators (as in Example 3).

For any $\mathbf{v} \in \mathbb{N}^d$, we may write $\mathbf{v} = (\overline{\mathbf{v}}, \mathbf{v}_\ell) \in \mathbb{N}^{d-1} \times \mathbb{N}$. If $M = \bigoplus_{\mathbf{v}\in\mathbb{N}^d} M_{\mathbf{v}}$ is a d-graded submodule of $k[x_1,\dots,x_d]^m$, we can construct a $(d-1)$-graded submodule $\overline{M}$ of $(k[x_d])[x_1,\dots,x_{d-1}]^m$ via

$$\overline{M} = \bigoplus_{\overline{\mathbf{v}}\in\mathbb{N}^{d-1}} \overline{M}_{\overline{\mathbf{v}}},$$

where $\overline{M}_{\overline{\mathbf{v}}} = \bigoplus_\ell M_{(\overline{\mathbf{v}},\mathbf{v}_\ell)}$ is itself a graded $k[x_d]$-module. Furthermore, note that $\mathbf{x}^{\overline{\mathbf{u}}} \cdot \overline{M}_{\overline{\mathbf{v}}} \subseteq \overline{M}_{\overline{\mathbf{u}}+\overline{\mathbf{v}}}$ for all $\overline{\mathbf{u}}, \overline{\mathbf{v}} \in \mathbb{N}^{d-1}$.

Lemma 2 can be generalized slightly:

Lemma 3 *Let $M = \bigoplus_{\mathbf{v}\in\mathbb{N}^d} M_{\mathbf{v}}$ be a d-graded submodule of $k[x_1,\dots,x_d]^m$. Let $GB(\overline{M}_{\overline{\mathbf{v}}})$ denote a Gröbner basis for the graded $k[x_d]$-module $\overline{M}_{\overline{\mathbf{v}}}$. Then*

$$\bigcup_{\overline{\mathbf{v}}\in\mathbb{N}^{d-1}} GB(\overline{M}_{\overline{\mathbf{v}}})$$

is a Gröbner basis of M.

We will use Lemma 3 with M as either $Z_i(X)$ or $B_i(X)$ for a finite d-filtered complex X. Of course, $\mathbb{N}^{d-1}$ is infinite, so it is necessary to choose a finite subset $S \subseteq \mathbb{N}^{d-1}$ so that $\bigcup_{\overline{\mathbf{v}}\in S} GB(\overline{M}_{\overline{\mathbf{v}}})$ serves as a *finite* Gröbner basis of M.

Let $X^{(i)}$ be the totally ordered set of i-simplices of the d-filtered complex X, and recall that for each $\sigma \in X^{(i)}$, we are given a set $Birth(\sigma)$ of gradings $\mathbf{u} \in \mathbb{N}^d$ where the simplex is born. We recall that we denote the coordinates of $\mathbf{u} \in \mathbb{N}^d$ by $u_1, u_2, ..., u_d$. We now define

$$V_j^{(i)}(X) = \left\{ u_j \;\middle|\; \mathbf{u} \in \bigcup_{\sigma\in X^{(i)}} Birth(\sigma) \right\} \subseteq \mathbb{N},$$

$$V^{(i)}(X) = \prod_{j=1}^{d} V_j^{(i)}(X) \subseteq \mathbb{N}^d,$$

and

$$\overline{V^{(i)}}(X) = \prod_{j=1}^{d-1} V_j^{(i)}(X) \subseteq \mathbb{N}^{d-1}.$$

If our input is of size n (i.e., $n = |\bigcup_{\sigma\in X^{(i)}} Birth(\sigma)|$), then $\left|V_j^{(i)}(X)\right| \leq n$, and hence $\left|V^{(i)}(X)\right| \leq n^d$ and $\left|\overline{V^{(i)}}(X)\right| \leq n^{d-1}$. When computing a Gröbner basis for $B_{i-1}(X)$ or $Z_i(X)$, we need only focus on the (at most) n^d gradings in $V^{(i)}(X)$ (or the (at most) n^{d-1} gradings in $\overline{V^{(i)}}(X)$), and thus we can tailor Lemma 3 to our setting:

Lemma 4 *Let X be a finite d-filtered complex. Let M denote either $Z_i(X)$ or $B_{i-1}(X)$, and let $GB(\overline{M}_{\overline{\mathbf{v}}})$ denote a Gröbner basis for the graded $k[x_d]$-module $\overline{M}_{\overline{\mathbf{v}}}$. Then*

$$\bigcup_{\overline{\mathbf{v}}\in\overline{V^{(i)}}(X)} GB(\overline{M}_{\overline{\mathbf{v}}})$$

is a Gröbner basis for M.

Lemma 4 allows us to split up the computation of the Gröbner basis $GB(Z_i(X))$ or $GB(B_i(X))$ of any d-filtered complex X into at most n^{d-1} separate computations of Gröbner bases of the graded modules $\overline{M}_{\overline{\mathbf{v}}}$ over the PID $k[x_d]$. This is wonderful news, because finitely generated (graded) modules over a (graded) PID are well-understood. Indeed, for any d-filtered complex X, specific algorithms for computing Gröbner bases of $Z_i(\overline{X}_{\overline{\mathbf{v}}})$ and $B_i(\overline{X}_{\overline{\mathbf{v}}})$ can be given by modifying slightly the algorithms given in [14].

However, the Gröbner basis guaranteed by Lemma 4 for $M = Z_i(X)$ or for $M = B_i(X)$ is not necessarily minimal. The solution to this problem is to compute minimal Gröbner bases for each individual $\overline{M}_{\overline{\mathbf{v}}}$ sequentially while maintaining separately a master list of generators. This master list is a subset of $\bigcup_{\overline{\mathbf{v}}\in\overline{V^{(i)}}(X)} GB(\overline{M}_{\overline{\mathbf{v}}})$ constructed throughout the computation of $GB(M)$ so as to form a *minimal* Gröbner basis for M at the conclusion of the algorithm. Specific implementation details are given in the following section.

6 Computing Multidimensional Persistence: Algorithm CMP1

In this section we give an algorithm for computing minimal Gröbner bases $GB(B_i(X))$ and $GB(Z_i(X))$ with respect to the lexicographic order discussed in Section 2. In our exposition of the algorithm, we will err on the side of providing too many details for the sake of completeness. For example, we explicitly describe an implementation of sparse vectors and associated operations. We recognize the existence of other equally valid implementations.

We briefly provide intuition for the algorithm which we'll describe throughout this section. First, Gaussian elimination serves as an $O(n^3)$ algorithm for 0-dimensional persistence; repeating this for $O(n^d)$ gradings results in an $O(n^{d+3})$ algorithm for computing (not necessarily minimal) Gröbner bases $GB(B_i(X))$ and $GB(Z_i(X))$. Next, the work of Zomorodian and Carlsson provide a way to calculate 1-dimensional persistence in $O(n^3)$ time by exploiting properties of finitely generated graded modules over a PID [14]; repeating their algorithm $O(n^{d-1})$ times (once for every grading in $\overline{V^{(i)}}(X)$) results in an $O(n^{d+2})$ algorithm for computing (not necessarily minimal) Gröbner bases.

Our algorithm builds upon this in two ways. First, we show that the algorithm of [14] can actually be used to calculate 2-dimensional persistence in $O(n^3)$ time, leading to an $O(n^{d+1})$ algorithm for computing (not necessarily minimal) Gröbner bases. Secondly, we show that calculating *minimal* Gröbner bases can be performed without a large cost in runtime.

Since our algorithm will calculate a Gröbner basis for a submodule M of a free $k[\mathbf{x}]^m$-module, we need to maintain a data structure `Poly` for elements of M and a data structure `GB` for minimal Gröbner bases $GB(B_i(X_{\overline{v}}))$ and $GB(Z_i(X_{\overline{v}}))$. We summarize these data structures in Figure 1. We denote the coordinates of M (which correspond to the totally ordered $(i-1)$- or i- simplexes of X) by `splx_idx` (i.e., "simplex index"). Lastly, we note that all arrays used are zero-indexed.

The elements of a `GB` will be stored in an array `eles` (i.e., "elements"), whose jth element is a `Poly`. For fast reduction, we maintain a hash table `splx2idx` (i.e., "simplex to index") within each `GB`. The member `splx2idx` takes a `splx_idx` (which represents a simplex) and identifies the index within `eles` of the unique `Poly` in the `GB` whose leading term is a multiple of the simplex corresponding to `splx_idx`. `splx2idx` will be used to aid the reduction of a `Poly` modulo the Gröbner basis represented by a `GB`. We assume that all `GB`s are initialized so that `splx2idx` and `eles` are empty.

We represent each `Poly` as a sparse vector. The nonzero entries in the vector corresponding to an element in M will be stored in an array `eles`; `eles[j]` gives a pair `(splx_idx, coeff)`, where `coeff` gives the field coefficient of the term in the `Poly` corresponding to the simplex $\tau_{\texttt{splx\_idx}}$. The elements of `eles` should be arranged in increasing order: for $i < j$, we should have that `eles[i].splx_idx` < `eles[j].splx_idx`. Within each `Poly`, we maintain a hash table `splx2idx`, which takes a `splx_idx` and identifies the index within `eles` of the term in the `Poly` corresponding to $\tau_{\texttt{splx\_idx}}$.

We further maintain in each `Poly` the grading (i.e., a tuple representing an element of $V^{(i)}(X)$) at which this `Poly` was generated in the field `grading`. We decompose

```
Grading: (first, last)
GB:
    - eles : int --> Poly
    - splx2idx : splx_idx --> int
    - combineWith : (Poly, Grading, GB) --> ()
Poly:
    - eles : int --> (splx_idx, coeff)
    - splx2idx : splx_idx --> int
    - grading : Grading
    - LC : Poly (or null)
reduceWith : (Poly, Poly, k, Grading) --> Poly
```

Fig. 1 Data structures used in CMP1.

each grading in $\mathbb{N}^d$ into the first $d-1$ coordinates (in the field `first` of the grading) and the last coordinate (in the field `last` of the grading). Each `Poly` will be reduced during computation, and we'll use the field `LC` to record the linear combination of i-simplexes which produce each `Poly` in $B_{i-1}(X_{\overline{\mathrm{v}}})$. The field `LC` will be set to `null` for elements of the GBs used to calculate $GB(Z_i(X_{\overline{\mathrm{v}}}))$. The `grading` of each `Poly` generator of $B_{i-1}(X)$ is initialized with the birth grading of the generator.

We also define two methods. The method `reduceWith` (Algorithm 1) is responsible for arithmetic on the `Poly`s. For `Poly`s f and g and a field coefficient `c`, calling `reduceWith(f, g, c, v)` will return the `Poly` `f+(c*g)` with its `grading` set to `v`.

`GB.combineWith` (Algorithm 2) is used to calculate minimal Gröbner bases $GB(\overline{M}_{\overline{\mathrm{v}}})$ (cf. Lemma 4). `GB.combineWith` takes as input a `Poly`, reduces it with respect to the `GB`, and adds it to the `GB` if the reduced `Poly` is nonzero. Furthermore, `GB.combineWith` is responsible for ensuring that its `GB` is a *minimal* Gröbner basis. The second argument to `GB.combineWith` is used to record at which grading the current reduction is taking place. There is a third argument to `GB.combineWith` which is used if and only if `GB` is a Gröbner basis for $B_{i-1}(X_{\overline{\mathrm{v}}})$. In this case, the third argument will be a Gröbner basis for $Z_i(X_{\overline{\mathrm{v}}})$.

Pseudocode for CMP1, which calculates minimal Gröbner bases `imageGB` for $B_{i-1}(X)$ and `kernelGB` for $Z_i(X)$, is given in Algorithm 3. We remark that, for fast implementation, W (cf. line 5) should be implemented as a list or array sorted in lexicographic order. Furthermore, we recommend that each `Poly` be assigned a unique identifier upon creation. This allows insertion into and membership queries for the sets $\mathcal{Z}$ and $\mathcal{B}$ (cf. line 6) to be performed in $O(1)$ time using a hash-based set.

We now prove the correctness of CMP1 and analyze its runtime. We denote the number of i-simplices by n, the number of $(i-1)$-simplices by m, and the persistence dimension by d. We assume that $d \geq 2$.

Lemma 5 `reduceWith` *(Algorithm 1) runs in time $O(m+n)$.*

Proof There are two key observations which validate this claim. First, `allIdxs` (line 3) can be constructed in linear time because we assume that `f.eles` and `g.eles` are sorted. Second, `reduceWith` is linear in the maximum possible size of `f.eles` and `f.LC.eles`. If `f` is an element of $B_{i+1}(X)$, the maximum possible size of `f.eles` is

Algorithm 1 `reduceWith(f, g, c, v)` returns `h = f+(g*c)`

```
1: procedure REDUCEWITH(f, g, c, v)
2:     Let h be a new Poly
3:     Let allIdxs ⟵ sorted({ele.splx_idx | ele in f.eles or g.eles})
4:     for splx_idx in allIdxs do
5:         Let newCoeff ⟵ 0
6:         if splx_idx in f.splx2idx then
7:             newCoeff ⟵ f.eles[f.splx2idx[splx_idx]].coeff
8:         end if
9:         if splx_idx in g.splx2idx then
10:            newCoeff ⟵ newCoeff + c*g.eles[g.splx2idx[splx_idx]].coeff
11:        end if
12:        if newCoeff ≠ 0 then
13:            h.splx2idx[splx_idx] ⟵ h.eles.size()
14:            h.eles.append((splx_idx, newCoeff))
15:        end if
16:    end for
17:    h.grading ⟵ v
18:    if f.LC is not null then
19:        h.LC ⟵ reduceWith(f.LC, g.LC, c, v)
20:    end if
21:    return h
22: end procedure
```

Algorithm 2 Algorithm to add a polynomial to a minimal Gröbner basis for a 1D persistence module

```
1: procedure GB.COMBINEWITH(g, v, KGB (default null))
2:     if g.eles[0].splx_idx is not in this.splx2idx then
3:         this.splx2idx[g.eles[0].splx_idx] ⟵ this.eles.size()
4:         this.eles.append(g)
5:         return
6:     end if
7:     h ⟵ this.eles[this.splx2idx[g.eles[0].splx_idx]]
8:     if g.grading.last < h.grading.last then
9:         this.eles[this.splx2idx[g.eles[0].splx_idx]] ⟵ g
10:        this.combineWith(h, (v.first, h.grading.last), KGB)
11:        return
12:    end if
13:    g ⟵ reduceWith(g, h, -g.eles[0].coeff * h.eles[0].coeff$^{-1}$, v)
14:    if g ≠ 0 then
15:        this.combineWith(g, v, KGB)
16:    end if
17:    if (g = 0) and (KGB is not null) and (g.LC ≠ 0) then
18:        KGB.combineWith(g.LC, v, null)
19:    end if
20: end procedure
```

m and the maximum possible size of `f.LC.eles` is n. If `f` is an element of $Z_i(X)$, the maximum possible size of `f.eles` is n and `f.LC` is `null`.

Lemma 6 *Let $B_{\overline{\mathbf{v}}}$ and $\underline{Z_{\overline{\mathbf{v}}}}$ be instances of a* GB *representing minimal Gröbner bases for submodules M_B of $\overline{B_{i-1}(X)_{\overline{\mathbf{v}}}}$ and M_Z of $\overline{Z_i(X)_{\overline{\mathbf{v}}}}$, respectively.*

Algorithm 3 [CMP1] Algorithm which calculates a minimal Gröbner basis for $Z_i(X)$ and $B_{i-1}(X)$ given the `Poly` generators `g` of $B_{i-1}(X)$, where each `g` is the boundary of some i-simplex of X.

```
 1: procedure CMP1(Poly generators {g} of $B_{i-1}(X)$)
 2:     Initialize empty sets kernelGB and imageGB (to store Gröbner bases for $Z_i(X)$ and $B_{i-1}(X)$)
 3:     for gradings $\overline{v}$ in $\overline{V^{(i)}}(X)$ in lexicographic order do
 4:         Initialize GBs named $Z_{\overline{v}}$ (for $\overline{Z_i(X)_{\overline{v}}}$) and $B_{\overline{v}}$ (for $\overline{B_{i-1}(X)_{\overline{v}}}$)
 5:         $W \longleftarrow \left\{\overline{w} \,\middle|\, \overline{w} \text{ is an immediate predecessor to } \overline{v} \text{ in } \overline{V^{(i)}(X)}\right\}$
 6:         Let $\mathcal{Z}$ and $\mathcal{B}$ be empty sets of Polys
 7:         for $\overline{w}$ in $W$ in reverse lexicographic order do
 8:             for g in $Z_{\overline{w}}$ such that g is not in $\mathcal{Z}$ do
 9:                 $Z_{\overline{v}}$.combineWith(g, ($\overline{v}$, g.grading.last), null)
10:                 Add g to $\mathcal{Z}$
11:             end for
12:         end for
13:         for $\overline{w}$ in $W$ in reverse lexicographic order do
14:             for g in $B_{\overline{w}}$ such that g is not in $\mathcal{B}$ do
15:                 $B_{\overline{v}}$.combineWith(g, ($\overline{v}$, g.grading.last), $Z_{\overline{v}}$)
16:                 Add g to $\mathcal{B}$
17:             end for
18:         end for
19:         for generator g in $\overline{B_{i-1}(X)_{\overline{v}}}$ do
20:             $B_{\overline{v}}$.combineWith(g, g.grading, $Z_{\overline{v}}$)
21:         end for
22:         for g in {g ∈ $B_{\overline{v}}$ | g.grading.first = $\overline{v}$} do
23:             Add g to imageGB
24:         end for
25:         for g in {g ∈ $Z_{\overline{v}}$ | g.grading.first = $\overline{v}$} do
26:             Add g to kernelGB
27:         end for
28:     end for
29:     return kernelGB and imageGB
30: end procedure
```

A call of the form $B_{\overline{v}}$`.combineWith(g, u,` $Z_{\overline{v}}$`)` *modifies* $B_{\overline{v}}$ *to be a minimal Gröbner basis for the module generated by* M_B *and* `g`*. If the addition of* `g` *into* $B_{\overline{v}}$ *introduces a syzygy* `h` *of the generators of* M_B *(i.e., an element of* $\overline{Z_i(X)_{\overline{v}}}$*), then* $B_{\overline{v}}$`.combineWith(g, u,` $Z_{\overline{v}}$`)` *further modifies* $Z_{\overline{v}}$ *to be a minimal Gröbner basis for the module generated by* M_Z *and* `h`*. Furthermore, calls of this form run in time* $O(n^2 + nm + m^2) = O(n^2 + m^2)$.

Similarly, a call of the form Calling $Z_{\overline{v}}$`.combineWith(g, u, null)` *modifies* $Z_{\overline{v}}$ *to be a minimal Gröbner basis for the module generated by* M_Z *and* `g`*. Calls of this form run in time* $O(n^2)$.

Proof Recall that `combineWith` assumes a priori that $B_{\overline{v}}$ and $Z_{\overline{v}}$ are minimal Gröbner bases. Every call to `combineWith` maintains this minimality. Indeed, calling $B_{\overline{v}}$`.combineWith(g, u,` $Z_{\overline{v}}$`)` reduces `g` with respect to M_B. If the remainder `g′` of this reduction is nonzero, then `g′` is added to $B_{\overline{v}}$. If `g′` is zero, then `g′.LC` (which keeps track of which reductions have been performed in terms of the generators of $B_i(X)$) is combined with $Z_{\overline{v}}$.

Combining `g` with $B_{\overline{\mathtt{v}}}$ requires at most m calls to `reduceWith` (which runs in time $O(m + n)$ (Lemma 5)). Combining `g.LC` with $Z_{\overline{\mathtt{v}}}$ requires at most n calls to `reduceWith` (which runs in time $O(n)$). We conclude that each call of the form $B_{\overline{\mathtt{v}}}$`.combineWith` takes time $O(n^2 + nm + m^2) = O(n^2 + m^2)$.

A similar argument holds for calls of the from $Z_{\overline{\mathtt{v}}}$`.combineWith(g, u, null)`.

Lemma 7 *At the conclusion of CMP1, each $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$) is a minimal Gröbner basis for $\overline{B_{i-1}(X)_{\overline{\mathtt{v}}}}$ (resp. $\overline{Z_i(X)_{\overline{\mathtt{v}}}}$).*

Proof We prove that the loop beginning in line 3 of CMP1 inductively constructs minimal Gröbner bases for $\overline{B_{i-1}(X)_{\overline{\mathtt{v}}}}$ (resp. $\overline{Z_i(X)_{\overline{\mathtt{v}}}}$).

Lines 7 through 21 of CMP1 compute minimal Gröbner bases for $\overline{B_{i-1}(X)_{\overline{\mathtt{v}}}}$ (resp. $\overline{Z_i(X)_{\overline{\mathtt{v}}}}$) in two stages. First, in lines 7 through 18, we combine the generators from all $B_{\overline{\mathtt{w}}}$ (resp. $Z_{\overline{\mathtt{w}}}$), where $\overline{\mathtt{w}}$ runs over all immediate predecessors of $\overline{\mathtt{v}}$ in the partial ordering on $\overline{V^{(i)}}(X)$. Lines 19 through 21 then combine generators that first appear in grading $\overline{\mathtt{v}}$. Minimality is guaranteed by `combineWith` (Lemma 6).

Theorem 4 *CMP1 computes minimal Gröbner bases for $Z_i(X)$ and $B_{i-1}(X)$.*

Proof Throughout the process of constructing $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$), `combineWith` is responsible for maintaining the grading when each generator `g` of $\overline{B_{i-1}(X)_{\overline{\mathtt{v}}}}$ (resp. $\overline{Z_i(X)_{\overline{\mathtt{v}}}}$) was born. Because $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$) is a minimal Gröbner basis (cf. Lemma 7), an element `g` in $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$) has grading $\overline{\mathbf{v}}$ only if the leading term of `g` is not divisible by the leading term of any element of any $\overline{B_{i-1}(X)_{\overline{\mathtt{w}}}}$ (resp. $\overline{Z_i(X)_{\overline{\mathtt{w}}}}$) for any immediate predecessors $\overline{\mathbf{w}}$ of $\overline{\mathbf{v}}$. After we have completely constructed Gröbner bases for $\overline{B_{i-1}(X)_{\overline{\mathtt{v}}}}$ (resp. $\overline{Z_i(X)_{\overline{\mathtt{v}}}}$), we add an element `g` to the minimal Gröbner basis for $B_{i-1}(X)$ (resp. $Z_i(X)$) if and only if `g` is associated with grading $\overline{\mathtt{v}}$ (cf. lines 22 through 27).

Theorem 5 *The runtime of CMP1 is $O(dn^d + n^{d-1}(n^2 + m^2)) \approx O(n^{d+1})$.*

Proof For this proof, we use the symbol `u` to denote an element of $\mathbb{N}^d$.

During our exposition of CMP1, we have so far split gradings into two pieces; that is, we have been representing any d-tuple `u` as a pair `(u.first, u.last)`, where `u.first` $\in \mathbb{N}^{d-1}$ and `u.last` $\in \mathbb{N}$. For this proof, we further split `u.first` into a pair `(u.first.first, u.first.last)`, where `u.first.first` $\in \mathbb{N}^{d-2}$ and `u.first.last` $\in \mathbb{N}$. Note that `u.first.first` will be empty if $d = 2$.

In the remainder of the proof, we will show that for any fixed `u`, the computation of all Gröbner bases for the $B_{\overline{\mathtt{v}}}$ such that $\overline{\mathtt{v}}$`.first = u.first.first` occurs in $O(dn^2 + n(n^2 + m^2))$ time. We will first consider the case where $d = 2$ (i.e., when `u.first.first` is empty) and then show that our argument holds for $d > 2$. Since there exactly n^{d-2} possibilities for `u.first.first` (and we assume that $d \ll n$), it follows that CMP1 runs in time $O(n^{d-1}(n^2 + m^2))$.

If $d = 2$, then $\overline{V^{(i)}}(X) \subseteq \mathbb{N}$ and W (defined on line 5 of CMP1) will contain at most one element throughout the algorithm. Therefore, there is no need to maintain each $B_{\overline{\mathtt{w}}}$ (resp. $Z_{\overline{\mathtt{w}}}$) after having constructed $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$). In this case, lines 7 through 18 of CMP1 are unnecessary; we need only maintain `GBs` B and Z as long as we remember to add the required `Polys` from B and Z to `imageGB` and `kernelGB` (lines 22 through 27) during each iteration of the loop. Hence, when $d = 2$ and when we make the simplification where we ignore lines 7 through 18, `combineWith` is called from CMP1 exactly n times, leading to an overall $O(n(n^2 + m^2))$ runtime.

Fortunately, when $d = 2$, the simplification described in the previous paragraph (which removes lines 7 through 18 from CMP1) does not cause a decrease in the overall complexity of CMP1. Since each $B_{\overline{\mathtt{w}}}$ (resp. $Z_{\overline{\mathtt{w}}}$) is assumed to be a minimal Gröbner basis, each of the calls to `combineWith` in lines 9 and 15 of CMP1 runs in constant time because the conditional in line 2 of `combineWith` will evaluate to `true`. Since the size of $B_{\overline{\mathtt{w}}}$ (resp. $Z_{\overline{\mathtt{w}}}$) is $O(n)$, lines 7 through 18 in CMP1 contribute a negligible $O(n^2)$ increase in the runtime to CMP1. Moreover, this increase is solely due to copying items from $B_{\overline{\mathtt{w}}}$ (resp. $Z_{\overline{\mathtt{w}}}$) into $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$).

We now consider the case when $d > 2$. The argument is similar; we wish to split the computation into $O(n^{d-2})$ two-dimensional slices (by two-dimensional "slice", we refer to all gradings `u` of $V^{(i)}(X)$ which share the same value for `u.first.first`). We propose that computing the Gröbner bases for each two-dimensional slice can be accomplished in $O(n(n^2 + m^2))$ time. If our goal was simply to compute Gröbner bases for $B_{i-1}(X)$ and $Z_i(X)$ in the proposed runtime, we could simply repeat the two-dimensional algorithm (analyzed in the previous two paragraphs) $O(n^{d-2})$ times. However, we wish to compute *minimal* Gröbner bases. Consequently, we must insert a `Poly` `g` of $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$) into `imageGB` (resp. `kernelGB`) if and only if `g` is reduced with respect to all $B_{\overline{\mathtt{w}}}$ (resp. $Z_{\overline{\mathtt{w}}}$) such that $\overline{\mathtt{w}} < \overline{\mathtt{v}}$. We accomplish this via lines 7 through 18 of CMP1.

Note that, for each $\overline{\mathtt{v}}$, the first iteration of lines 8 though 11 and in lines 14 through 17 will only *copy* elements from the relevant $B_{\overline{\mathtt{w}}}$ (resp. $Z_{\overline{\mathtt{w}}}$) into $B_{\overline{\mathtt{v}}}$ (resp. $Z_{\overline{\mathtt{v}}}$); no reductions will occur. Note further that if there is a predecessor $\overline{\mathtt{w}}$ of $\overline{\mathtt{v}}$ in the same two-dimensional slice, then this $\overline{\mathtt{w}}$ is lexicographically last among the immediate predecessors of $\overline{\mathtt{v}}$.

The key to calculating minimal Gröbner bases while maintaining a modest runtime lies in calling `combineWith` on a `Poly` only if that `Poly` has not yet been processed into $\mathtt{B}_{\overline{\mathtt{v}}}$. For each generator `g` of $B_{i-1}(X)$, we would like to ensure that `g` only causes one call to `combineWith` from CMP1 per two-dimensional slice. To achieve this, lines 7 and 13 traverse W in *reverse* lexicographic order. As a result of this ordering, for each $\overline{\mathtt{v}}$, during the first iteration of lines 8 through 11 and 14 through 17, $\overline{\mathtt{w}}$ is in the same two-dimensional slice as $\overline{\mathtt{v}}$. Because of lines 8, 10, 14, and 16, subsequent iterations of lines 8 through 11 and 14 through 17 (for the same $\overline{\mathtt{v}}$) pass to `combineWith` `Poly` generators `g` (potentially partially reduced in previously processed two-dimensional slices) only if `g.grading.first.last` = $\overline{\mathtt{v}}$`.last`. Indeed, if `g.grading.first.last` < $\overline{\mathtt{v}}$`.last`, then `g` would have already been added to $\mathcal{Z}$ or $\mathcal{B}$.

Consequently, for each two-dimensional slice, calls to `combineWith` contribute $O(n(n^2+m^2))$ to the algorithm's time complexity. Moreover, lines 8 and 14 contribute $O(dn)$ to the time complexity of CMP1 for each grading $\overline{\mathtt{v}}$, since lines 8 and 14 must filter $O(n)$ `Poly`s from each of $O(d)$ `GB`s.

Thus, our algorithm runs in time

$$O(n^{d-2}(dn^2 + n(n^2 + m^2))) = O(dn^d + n^{d-1}(n^2 + m^2)).$$

Moreover, we do not need to consider all m $(i-1)$-simplices; we need only consider those simplices which occur as the face of some i-simplex. There are at most $(i+1)n$ of these. In most applied topology contexts, i and d are taken to be small; by assuming that $i = O(1)$ and $d = O(n)$, we obtain an approximate runtime bound of $O(n^{d+1})$ for CMP1.

Remark 10 The proof of Theorem 5 actually provides a slightly tighter complex-dependent bound on the runtime of CMP1. If $V^{(i)}_{1:d-2}(X)$ denotes the set obtained by projecting each element of $V^{(i)}(X)$ onto its first $d-2$ coordinates, then our proof shows that the runtime of CMP1 when computing $B_{i-1}(X)$ and $Z_i(X)$ is

$$O\left(\left|V^{(i)}_{1:d-2}(X)\right|\left(dn^2 + n(n^2+m^2)\right)\right).$$

Remark 11 It is possible to parallelize CMP1! Although we specified in line 3 that $\overline{V^{(i)}}(X)$ was to be traversed in increasing lexicographic order, it is possible to execute lines 4 through 27 of CMP1 for any grading $\overline{\mathbf{v}} \in \overline{V^{(i)}}(X)$ provided that these lines have already been executed for all gradings $\overline{\mathbf{w}} \in \overline{V^{(i)}}(X)$ such that $\overline{\mathbf{w}}$ is an immediate predecessor of $\overline{\mathbf{v}}$.

By taking full advantage of this fact in a setting with limitless parallelization, the runtime of our algorithm becomes $O(dn^2(d+n^2+m^2))$ for $d > 2$. Indeed, the runtime can be bounded above by the product of (a) the runtime of the computation of one iteration of lines 4 through 27 in CMP1, and (b) the length of a minimal sequence $\overline{\mathbf{v}}_0, \overline{\mathbf{v}}_1, \ldots, \overline{\mathbf{v}}_\ell$ from the minimum element of $\overline{V^{(i)}}(X)$ to the maximum element of $\overline{V^{(i)}}(X)$, where each $\overline{\mathbf{v}}_{r+1}$ is an immediate successor to $\overline{\mathbf{v}}_r$ in the partial order on $\overline{V^{(i)}}(X)$. Theorem 5 shows that one iteration of lines 4 through 27 has runtime $O(dn + n(n^2+m^2))$, and the maximum possible length of such a minimal sequence has $\ell = (d-1)(n-1)+1 = O(dn)$ (this bound is tight when $|V^{(i)}_j(X)| = n$ for all j).

Remark 12 Although we have managed to achieve a respectable upper bound on the runtime of the computation of a Gröbner basis for $B_i(X)$ and $Z_i(X)$, our algorithm as stated above will quite possibly execute a number of redundant computations (in lines 7 through 21 of CMP1). We would like a way to ensure that if two polynomials have been reduced in one two-dimensional slice, then these two polynomials will never again be reduced in another two-dimensional slice. We leave as open the problem of constructing an algorithmic solution to this problem.

We mention that this problem is also encountered in the original version of Buchberger's algorithm, where one does not wish to compute a reduction to any S-polynomial if one has prior information that this reduction will be 0. Faugère's F5 algorithm [11] provides a good solution to this problem in the case when one is trying to calculate a Gröbner basis of an ideal of a polynomial ring. It would be highly beneficial to adopt these techniques to the persistence setting in a way that does not greatly increase the time or space complexity of our algorithm.

Remark 13 Given Gröbner bases G_B for $B_i(X)$ and G_Z for $Z_i(X)$, we can calculate the rank $\rho_{\mathbf{u},\mathbf{v}}(H_i(X))$ for any $\mathbf{u}, \mathbf{v} \in \mathbb{N}^d$ using the algorithm given in Section 6.3 of [4]. A cursory analysis of the algorithm shows that the runtime of calculating $\rho_{\mathbf{u},\mathbf{v}}(H_i(X))$

is well approximated by the runtime of column reducing the matrix whose columns correspond to the elements of G_B and G_Z. The runtime of such an operation is $O\left(n\left(|G_B| + |G_Z|\right)^2\right)$.

7 Empirical Evaluation of Runtime

In the previous section we have described an algorithm for computing multidimensional persistence and provided an upper bound on its runtime. In this section, we randomly construct and compute the persistent homology of d-filtered simplicial complexes.

In [4], empirical measures of runtime were calculated from simulated boundary matrices. In an attempt to better emulate the boundary matrix of a collection of i-simplices, each column in these simulated boundary matrices contained the same number of nonzero entries. Our experiments, motivated by the following proposition, differ slightly in that we construct boundary matrices from actual d-filtered abstract simplicial complexes.

Proposition 1 *There exists a matrix over $\mathbb{F}_2$ such that each column contains the same number of nonzero entries that does not arise as the boundary matrix from any abstract simplicial complex.*

Proof Consider the following matrix:

$$M = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Assume towards a contradiction that $M = \partial_2(X)$ for some abstract simplicial complex X. Since the rows of M must be indexed by the 1-simplices of X, we may assume without loss of generalization that the first row of M represents the 1-simplex $[0, 1]$, that the second row of M represents the 1-simplex $[0, 2]$, and that the third row of M represents $[1, 2]$. Because these are the only three edges that may exist among an abstract simplicial complex with three vertices, the existence of a fourth row in M implies the existence of a fourth vertex [3]. But then the second column of M represents a 2-simplex which contains 4 vertices, which is impossible.

Remark 14 Proposition 1 presents a number of questions which we leave as open problems. For example:

1. For a given dimension i, a number n of i-simplices, and a number m of $i - 1$ simplices, what percentage of all $m \times n$ matrices with exactly $i + 1$ nonzero entries in each column arise as the boundary matrices of a simplicial complex?
2. Does the time complexity of computing persistent homology change when considering only matrices which arise as boundary matrices of some simplicial complex?
3. Does the assumption that a matrix M arises as the boundary matrix of a simplicial complex provide any information about the size of a Gröbner basis G for the image or boundary of M?

Thus, for our experiments, we randomly generate an i-dimensional abstract simplicial complex X to test the computation of $GB(Z_i(X))$ and $GB(B_i(X))$ according to the following procedure. First, we specify the values of six variables:

- the dimension i of the abstract simplicial complex X
- the number $|V|$ of vertices of X
- the number $|S|$ of i-simplices of X
- the persistence dimension d of X
- the number of distinct persistence coordinates $|P|$ per persistence dimension (notated above as $\left|V_j^{(i)}(X)\right|$)
- the number $|T|$ of birth times per i-simplex

We randomly create a set $\mathcal{X}$ of cardinality $|S|$, each element of which is a subset of $\{0, 1, \ldots, |V| - 1\}$ of cardinality $i + 1$. For each element of $\mathcal{X}$, we assign $|T|$ d-tuples, each entry of which is a natural number between 0 and $|P| - 1$ inclusive. The elements of $\mathcal{X}$ represent a collection of $|S|$ i-simplexes, and the d-tuples associated to each simplex represent the birth times of the simplex. We then run algorithm CMP1 against the complex and measure runtime.

We first test the effect of changing the dimension i of the complex. Figure 2 shows the runtime while varying i from 2 to 8. We see a modest runtime for smaller complex dimensions. However, once the dimension of the complex of the dimension of the complex reaches 6, the runtime increases significantly.

We next test the effect of varying the number of vertices between 25 and 5000. For this experiment, we fix the number of simplexes at 500, the dimension of the complex at 3, the persistence dimension at 4, the number of distinct persistence coordinates in each (persistence) dimension at 50, and the number of birth times per simplex at 1. With these settings, the runtime of CMP1 varies between 5.75 seconds and 6.25 seconds, and the runtime for this experiment appears to be independent of the number of vertices.

For our third test, we study the effect of changing the number of simplexes in the top dimension. Figure 3 shows the runtime while varying the number $|S|$ of simplexes of top dimension between 100 and 10,000. Note that the linear relationship between runtime and the number of simplexes of top dimension in no way contradicts the $O(n^5)$ runtime suggested by Theorem 5. In this experiment, we fix the number of birth times per persistence dimension at 50; in contrast, Theorem 5 implicitly assumes that the number of birth times per persistence dimension is $\Theta(|S|)$.

Next, we study the relationship between persistence dimension and runtime. Figure 4 shows the runtime while varying the persistence dimension between 1 and 5. We see from this graph that, as suggested by Theorem 5, persistence dimension has a very significant effect on runtime.

For our next experiment, we plot the runtime of CMP1 against $|P|$, the number of distinct persistence coordinates per persistence dimension and present the results in Figure 5. We see a modest runtime when the number of unique persistence coordinates per dimension stays below 100. We thus recommend that users of multidimensional persistence bin persistence coordinates to maintain a modest runtime.

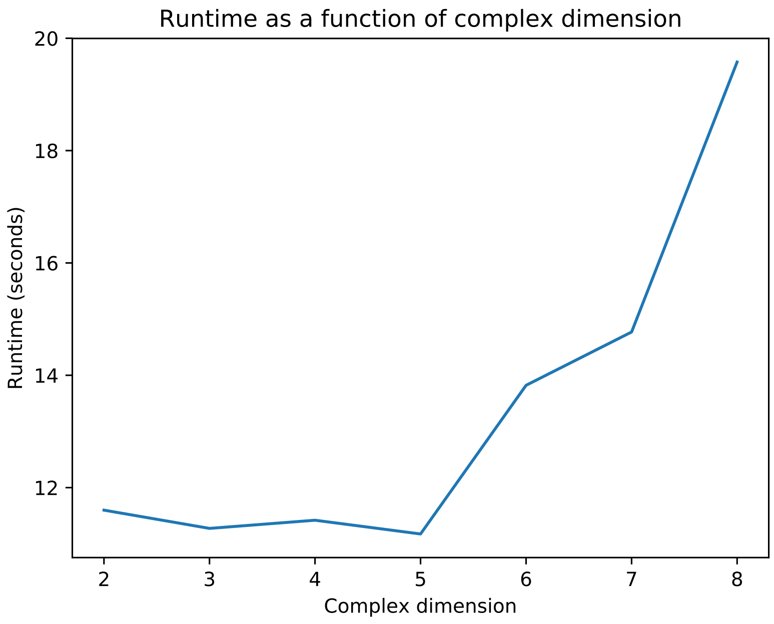

Fig. 2 The runtime of CMP1 as a function of the complex dimension i. For this experiment, we fix the number of vertices and the number of i-simplices at 1000. Furthermore, the persistence dimension is fixed at 4, the number of distinct persistence coordinates in each (persistence) dimension is fixed at 50, and the number of birth times per simplex is fixed at 1.

Finally, we plot the runtime of CMP1 against the number $|T|$ of birth times per simplex. Figure 6 suggests that there is a roughly linear relationship between the number of birth times per simplex and the runtime of CMP1.

References

1. Carlsson, G.: Topology and data. Bulletin of the American Mathematical Society **46**, 255–308 (2009)
2. Carlsson, G.: Topological pattern recognition for point cloud data. Acta Numerica **23**, 289–368 (2014)
3. Carlsson, G., Ishkhanov, T., De Silva, V., Zomorodian, A.: On the local behavior of spaces of natural images. International journal of computer vision **76**(1), 1–12 (2008)
4. Carlsson, G., Singh, G., Zomorodian, A.: Computing multidimensional persistence. Journal of Computational Geometry **1**(1), 72–100 (2010)
5. Carlsson, G., Zomorodian, A.: The theory of multidimensional persistence. Discrete & Computational Geometry **42**(1), 71–93 (2009)
6. Cox, D.A., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2007)
7. De Silva, V., Carlsson, G.: Topological estimation using witness complexes. Proc. Sympos. Point-Based Graphics pp. 157–166 (2004)
8. Dummit, D.S., Foote, R.M.: Abstract algebra, vol. 1984. Wiley Hoboken (2004)
9. Emmett, K.J., Rabadan, R.: Characterizing scales of genetic recombination and antibiotic resistance in pathogenic bacteria using topological data analysis. In: Brain Informatics and Health, pp. 540–551. Springer (2014)

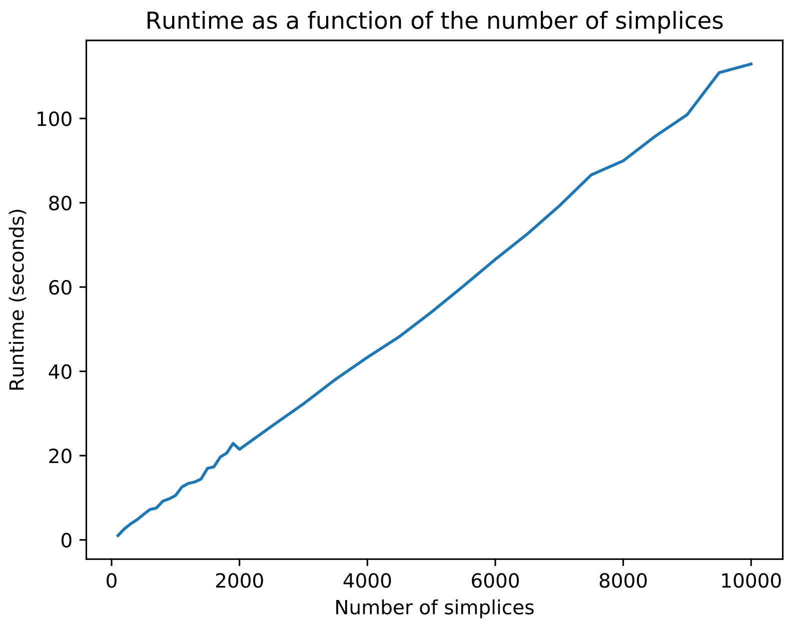

Fig. 3 The runtime of CMP1 as a function of the number $|S|$ of simplices of dimension i. For this experiment, we fix the number of vertices at 1000, the dimension of the complex at 3, the persistence dimension is fixed at 4, the number of distinct persistence coordinates per persistence dimension is fixed at 50, and the number of birth times per simplex is fixed at 1. The runtime profile assuming 250 and 500 vertices is almost identical (although not shown here).

10. Faugère, J.C.: A new efficient algorithm for computing gröbner bases (f4). Journal of pure and applied algebra **139**(1), 61–88 (1999)
11. Faugère, J.C.: A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In: International Symposium on Symbolic and Algebraic Computation Symposium-ISSAC 2002 (2002)
12. Hatcher, A.: Algebraic topology. Cambridge University Press, Cambridge, New York (2002)
13. Singh, G., Memoli, F., Ishkhanov, T., Sapiro, G., Carlsson, G., Ringach, D.L.: Topological analysis of population activity in visual cortex. Journal of vision **8**(8), 11 (2008)
14. Zomorodian, A., Carlsson, G.: Computing persistent homology. Discrete & Computational Geometry **33**(2), 249–274 (2005)

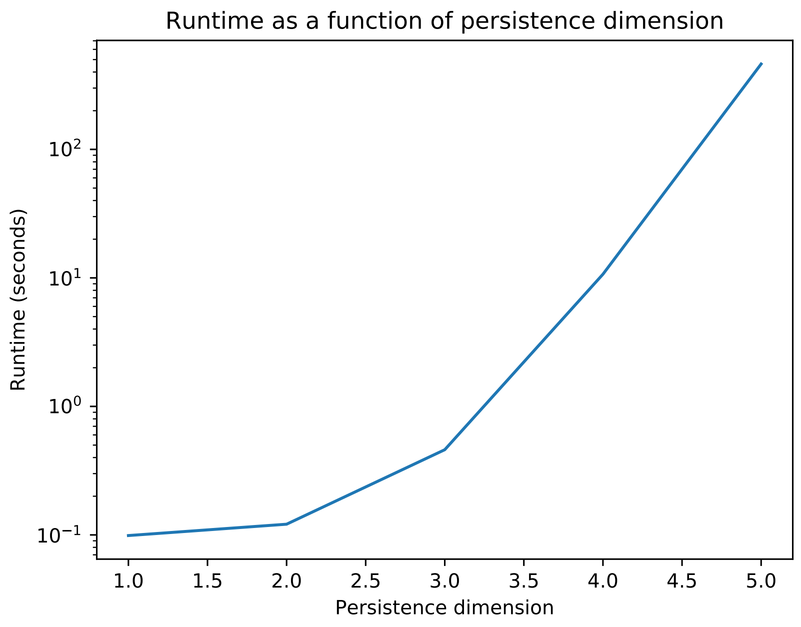

Fig. 4 The runtime of CMP1 as a function of persistence dimension. For this experiment, we fix the number of vertices and simplexes of top dimension at 1000, the dimension of the complex at 3, the number of distinct persistence coordinates per persistence dimension at 50, and the number of birth times per simplex at 1.

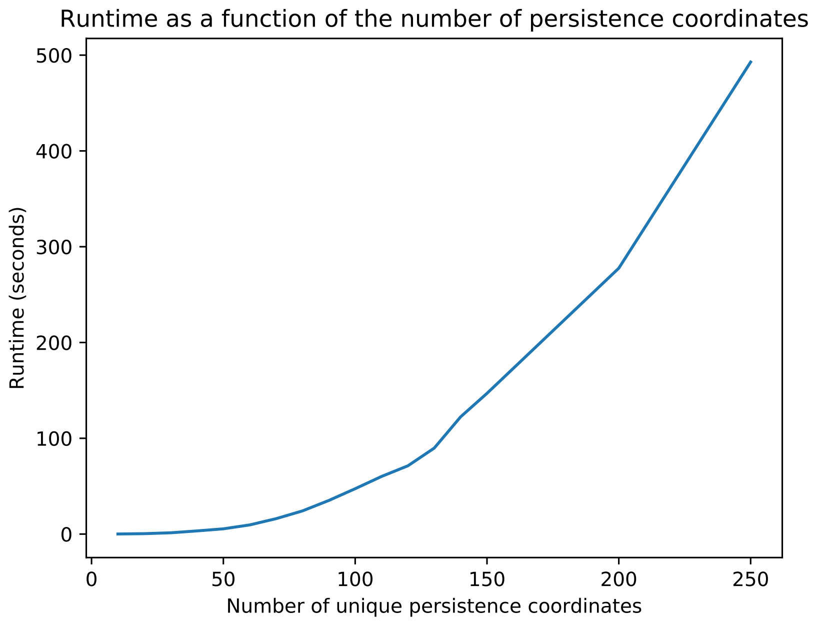

Fig. 5 The runtime of CMP1 as a function of the number of distinct persistence coordinates per persistence dimension. For this experiment, we fix the number of vertices and simplexes of top dimension at 500, the dimension of the complex at 3, the persistence dimension at 4, and the number of birth times per simplex at 1.

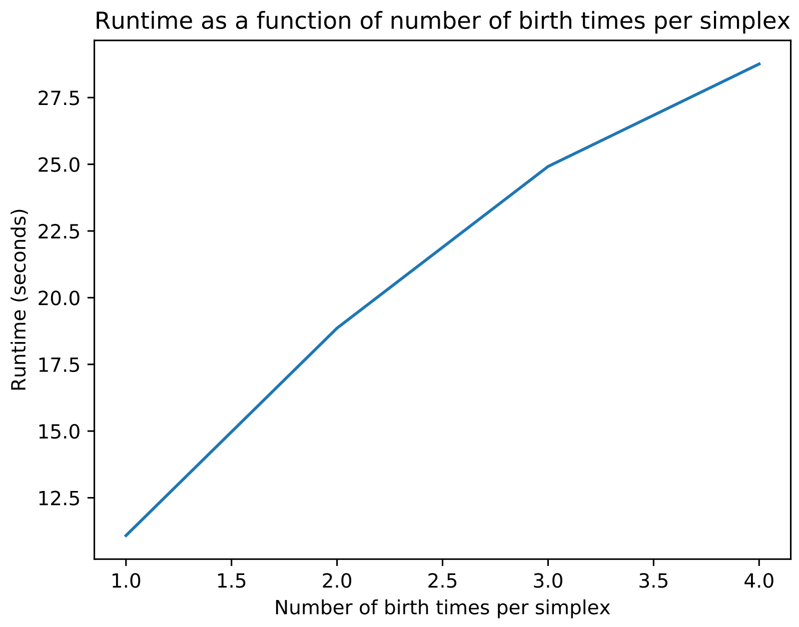

Fig. 6 The runtime of CMP1 as a function of the number of birth times per simplex in top dimension. For this experiment, we fix the number of vertices and simplexes of top dimension at 1000, the dimension of the complex at 3, the persistence dimension at 4, and the number of distinct persistence coordinates per persistence dimension at 50.