Chunk Reduction for Multi-Parameter Persistent Homology

Ulderico Fugacci
Graz University of Technology, Graz, Austria
fugacci@tugraz.at

Michael Kerber
Graz University of Technology, Graz, Austria
kerber@tugraz.at

Abstract

The extension of persistent homology to multi-parameter setups is an algorithmic challenge. Since most computation tasks scale badly with the size of the input complex, an important pre-processing step consists of simplifying the input while maintaining the homological information. We present an algorithm that drastically reduces the size of an input. Our approach is an extension of the chunk algorithm for persistent homology (Bauer et al., Topological Methods in Data Analysis and Visualization III, 2014). We show that our construction produces the smallest multi-filtered chain complex among all the complexes quasi-isomorphic to the input, improving on the guarantees of previous work in the context of discrete Morse theory. Our algorithm also offers an immediate parallelization scheme in shared memory. Already its sequential version compares favorably with existing simplification schemes, as we show by experimental evaluation.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures; Computing methodologies → Shared memory algorithms

Keywords and phrases Multi-parameter persistent homology, Matrix reduction, Chain complexes

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.37

Related Version A full version of this paper is available at `https://arxiv.org/abs/1812.08580`.

Funding Supported by the Austrian Science Fund (FWF) grant number P 29984-N35.

Acknowledgements We thank Sara Scaramuccia for initial discussions on this project, Wojciech Chacholski, Michael Lesnick and Francesco Vaccarino for helpful suggestions, and Federico Iuricich for his help on the experimental comparison with [10]. The datasets used in the experimental evaluation are courtesy of the AIM@SHAPE data repository [9].

1 Introduction

In the last decades, topology-based tools are gaining a more and more relevant role in the analysis and in the extraction of the core information of unorganized, high-dimensional and potentially large datasets. Thanks to its capability of keeping track of the changes in the homological features of a dataset which evolves with respect to a parameter, *persistent homology* has represented a real game-changer in this field. Recently, an extension of persistent homology called *multi-parameter persistent homology* is drawing the attention of a growing number of researchers. In a nutshell, multi-parameter persistence generalizes the classic persistent homology by studying multivariate datasets which are filtered by two or more (independent) scale parameters. Multi-parameter persistent homology of a dataset cannot be captured by complete discrete invariant [5], but this has not prevented the researchers from defining several descriptors based on multi-parameter persistence [6, 12, 15]. Since these descriptors tend to have high algorithmic complexity, it is a natural pre-processing step to pass to a smaller but equivalent representation of the input complex and to invoke any demanding computation on this smaller representation.

Contribution. Inspired by the chunk algorithm [3] for persistent homology, we present a reduction algorithm which for a filtered dataset returns a filtered chain complex having the same multi-parameter persistence but a drastically smaller size. Our approach is based on the observation that even in the absence of a global persistence diagram, local matrix reductions yield pairs of simplices which can be eliminated from the boundary matrix without affecting the homological information. Our approach proceeds in two steps, first identifying such local pairs of simplices, and in a second step manipulating the non-local columns of the boundary matrix such that all indices of locally paired simplices disappear. Both steps permit a parallel implementation with shared memory. We prove that our simplification scheme is optimal in the sense that any chain complex quasi-isomorphic to the input complex must contain at least as many generators as our output complex. Our algorithm yields similar time and space complexity bounds as its one-parameter counterpart. We implemented our algorithm, making use of various techniques that have proven effective for persistence computation, such as the twist reduction [8] and efficient data structure for the columns of a boundary matrix [4]. We experimentally show that our implementation is effective (see next paragraph). For the sake of clarity, our approach is described for the two-parameter case. No constraint prevents the generalization of the proposed method to an arbitrary number of parameters.

Comparison with related work. Our work is motivated by a line of research on complex simplification using discrete Morse theory (DMT) [11, 1, 2, 17]. In these works, the idea is to build a discrete gradient locally and to return the resulting Morse complex on critical simplices as a simplification. In analogy to the one-parameter case, our chunk algorithm is an attempt to realize this simplification scheme using persistent homology instead of DMT. This gives more flexibility, as in DMT, the paired cells are constrained to be incident, while this restriction is not present for persistence pairs. Consequently, we are able to prove optimality of our output size in general, while DMT-based approaches only succeeded to give guarantees for special cases such as multi-filtrations of 3D regular grids and of abstract simplicial 2-complexes [16]. In practice, the theoretical benefit in terms of output size is rather marginal, as our experiments show; however, the timings show that our improved theoretical guarantee comes without performance penalty; on the contrary, our algorithm is always faster than the DMT-based approach presented in [17] on all tested examples. We remark, however, that the DMT-based algorithm returns a complex endowed with a Forman gradient as well as the corresponding discrete Morse complex, which can be of potential use for other application domains than computing persistent homology.

A related question is the computation of a *minimal presentation* of a persistence module induced by a simplicial or general chain complex. Roughly speaking, a presentation consists of a finite set of (graded) generators and relations, and the minimal presentation is one with the minimal possible number of generators and relations. Our algorithm does not yield a minimal presentation, however, since more computations are needed to identify generators of the chain complex as generators or as relations. An algorithm by Lesnick and Wright[1] [13] computes such a minimal presentation through matrix reduction in cubic time, carefully choosing a column order to reduce the number of reduction instances. Their algorithm is used

[1] `https://www.ima.umn.edu/2017-2018/SW8.13-15.18/27428`

in their software package RIVET [14]. Our contribution can serve as a pre-processing step for their algorithm, reducing the size of the matrix through efficient and possibly parallelized computation and invoking their global reduction step on a much smaller chain complex.

2 Background

Homology. Fixed a base field $\mathbb{F}$ and a positive integer d, let us consider, for each $0 \leq k \leq d$, a finite collection of elements denoted as *k-generators* (or, equivalently, generators of dimension k). A finitely generated chain complex $C_* = (C_k, \partial_k)$ over $\mathbb{F}$ is a collection of pairs (C_k, ∂_k) where:

- C_k is the $\mathbb{F}$-vector space spanned by the generators of dimension k,
- $\partial_k : C_k \to C_{k-1}$ is called *boundary* map and it satisfies the property that $\partial_{k-1}\partial_k = 0$.

The elements of C_k are called *k-chains* and, by definition, are $\mathbb{F}$-linear combinations of the generators of dimension k. The *support* of a k-chain is the set of k-generators whose coefficient in the chain is not zero. In the following, we will simply use the term chain complex in place of finitely generated chain complex. Moreover, we will assume that, given a chain complex, an explicit set of generators is provided.

Given a chain complex C_*, we denote as $Z_k := \ker \partial_k$ the space of the k-cycles of C_*, and as $B_k := \text{Im } \partial_{k+1}$ the space of the k-boundaries of C_*. The k^{th} *homology space* of C_* is defined as the vector space $H_k(C_*) := Z_k/B_k$. The rank β_k of the k^{th} homology space of a chain complex C_* is called the k^{th} *Betti number* of C_*.

A *chain map* $f_* : C_* \to D_*$ is a collection of linear maps $f_k : C_k \to D_k$ which commutes with the boundary operators of C_* and of D_*. A simple example of a chain map is the inclusion map, if C_* is a subcomplex of D_*. In general, a chain map f_* induces linear maps $H_k(C_*) \to H_k(D_*)$ for every k.

Chain complexes allow for capturing the combinatorial and the topological structure of a discretized topological space. Given a finite simplicial complex K, the chain complex C_* associated to K is defined by setting C_k as the $\mathbb{F}$-vector space generated by the k-simplices of K and the boundary $\partial_k(c)$ of a k-chain c corresponding to a k-simplex σ as the collection of the $(k-1)$-simplices lying on the geometrical boundary of σ. Consequently, homology of a finite simplicial complex K is defined as the homology of the chain complex C_* associated to K. Intuitively, homology spaces of K reveal the presence of "holes" in the simplicial complex. The non-null elements of each homology space are cycles, which do not represent the boundary of any collection of simplices of K. Specifically, β_0 counts the number of connected components of K, β_1 its tunnels and holes, and β_2 the shells surrounding voids or cavities.

Multi-parameter persistent homology. In the following, we will focus for simplicity on datasets filtered by two independent scale parameters. All definitions and results in this paper can be generalized to more parameters without problems. Let $p = (p_x, p_y), q = (q_x, q_y) \in \mathbb{R}^2$, we will write throughout $p \leq q$ if $p_x \leq q_x$ and $p_y \leq q_y$. Given a chain complex $C_* = (C_k, \partial_k)$, let us assume to have an assignment which, for each generator $g \in C_k$, returns a value $v(g) \in \mathbb{R}^2$ such that, for any generator $g' \in C_{k-1}$, if $\langle \partial_k g, g' \rangle \neq 0$, then $v(g') \leq v(g)$. The assignment v can be extended to a function $v : C_* \to \mathbb{R}^2$ assigning to each chain c the least common upper bound $v(c)$ of the values $v(g)$ of the generators in the support of c. In the following, $v(c)$ will be called the *value* of c.

Given a chain complex $C_* = (C_k, \partial_k)$ endowed with a value function $v : C_* \to \mathbb{R}^2$ and fixed a value $p \in \mathbb{R}^2$ we define $C_*^p = (C_k^p, \partial_k^p)$ as the chain complex for which:

- C_k^p is the space of the k-chains of C_k having value lower than or equal to p,
- ∂_k^p is the restriction of ∂_k to C_k^p.

By the definition of v, for any chain c of dimension k, we have that $v(\partial_k(c)) \leq v(c)$. So, C_*^p is well-defined. For $p, q \in \mathbb{R}^2$ with $p \leq q$, C_*^p is a chain subcomplex of C_*^q. For this reason, we denote the collection C of the chain complexes C_*^p with $p \in \mathbb{R}^2$ as *bifiltered chain complex.*

Given $p, q \in \mathbb{R}^2$ with $p \leq q$, the inclusion map from C_*^p to C_*^q induces a linear map between the corresponding homology spaces $H_k(C_*^p)$ and $H_k(C_*^p)$. The *multi-parameter persistence k^{th} module* $H_k(C)$ of a bifiltered chain complex C is the collection of the homology spaces $H_k(C_*^p)$ with p varying in $\mathbb{R}^2$ along with all the linear maps induced by the inclusion maps.

Let C, D be two bifiltered chain complexes. We call C and D *homology-equivalent* if, for any fixed $k \in \mathbb{N}$, $H_k(C_*^p)$ and $H_k(D_*^p)$ are isomorphic via a map ψ_k^p and, for any $p, q \in \mathbb{R}^2$ with $p \leq q$, the diagram

$$\begin{tikzcd}
H_k(C_*^p) \arrow[r] \arrow[d, "\psi_k^p"] & H_k(C_*^q) \arrow[d, "\psi_k^q"] \\
H_k(D_*^p) \arrow[r] & H_k(D_*^q)
\end{tikzcd}$$

commutes where horizontal maps are induced by inclusion maps. Moreover, we call C and D *quasi-isomorphic* if the isomorphisms of the above diagram are induced by a collection of chain maps $f_*^p : C_*^p \to D_*^p$ satisfying, for any $p \leq q$ and any k, the commutative diagram

$$\begin{tikzcd}
C_k^p \arrow[r] \arrow[d, "f_k^p"] & C_k^q \arrow[d, "f_k^q"] \\
D_k^p \arrow[r] & D_k^q
\end{tikzcd}$$

in which horizontal maps are the inclusion maps. By definition, quasi-isomorphic bifiltered chain complexes are homology-equivalent. The converse, as depicted in Figure 1, does not hold in general. Two chain maps $\alpha_*, \beta_* : C_* \to D_*$ are called *chain-homotopic* if there exists

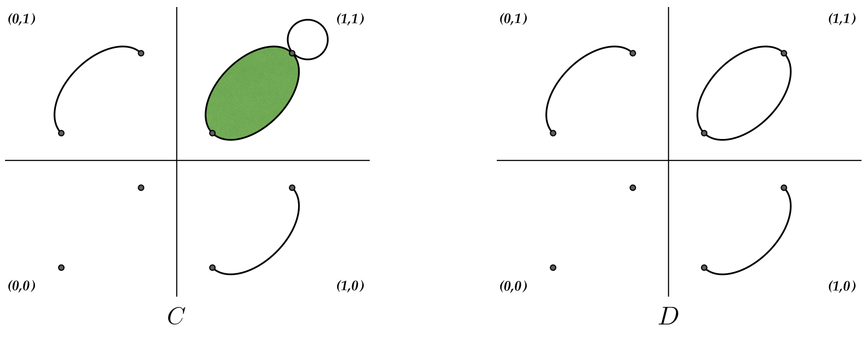

Figure 1 The two depicted bifiltered chain complexes are homology-equivalent but not quasi-isomorphic.

a collection of maps $\phi_k : C_k \to D_{k+1}$ such that, for any k, $\partial_{k+1}^D \phi_k + \phi_{k-1} \partial_k^C = \alpha_k - \beta_k$. Two bifiltered chain complexes C and D are called *homotopy-equivalent* if, there exist two collections of chain maps $f_*^p : C_*^p \to D_*^p$ and $g_*^p : D_*^p \to C_*^p$ such that, for any $p \in \mathbb{R}^2$,

f_*^p and g_*^p are homotopy-inverse one with respect to the other (i.e., $g_*^p f_*^p$ and $f_*^p g_*^p$ are chain-homotopic to $\mathrm{id}_{C_*^p}$ and $\mathrm{id}_{D_*^p}$, respectively) and they satisfy, for any $p \leq q$, the following commutative diagram

$$\begin{tikzcd}
C_k^p \arrow[r] \arrow[d, shift right, "f_k^p"'] & C_k^q \arrow[d, shift right, "f_k^q"'] \\
D_k^p \arrow[r] \arrow[u, shift right, "g_k^p"'] & D_k^q \arrow[u, shift right, "g_k^q"']
\end{tikzcd}$$

in which horizontal maps are the inclusion maps. Homotopy-equivalent bifiltered chain complexes are necessarily quasi-isomorphic.

Representation of bifiltered chain complexes. We assume that the bifiltered chain complex in input is provided as a finite sequence of triples of the form

$$(p, k, \Delta)$$

where $p \in \mathbb{R}^2$, k is a positive integer, and Δ is a finite list of pairs (l, λ) with $\lambda \in \mathbb{F}$ and $l \in \mathbb{N}$. In order to retrieve the bifiltered chain complex C represented by this list, each triple (p, k, Δ) has to be interpreted as a k-generator c with value $v(c) = p$ and whose boundary is encoded through Δ. Specifically, by defining as c_l the generator stored in position l of the sequence of triples, the pair (l, λ) in Δ represents that c_l appears in the boundary of c with coefficient λ. In order to ensure that the above-described interpretation of the input sequence actually returns a valid bifiltered chain complex, we require that, if c' is a generator appearing in the boundary of c, then the value v associated to c' is lower than or equal to the one taken by c.

Given a chain complex C_* endowed with a value function v, let us consider an injective function i providing the set of generators of C_* with a total order which is consistent with the partial order induced by the value funtion v. More precisely, given two generators c_1, c_2 of C_*, the *index* function i has to satisfy that $v(c_1) \leq v(c_2)$ implies $i(c_1) \leq i(c_2)$.

We uniquely represent the bifiltered chain complex C as follows. Each generator c of C_* is stored in a data structure that we call the *column* of the generator. A column stores:

- the index $i(c)$,
- the value $v(c)$,
- the dimension k of g,
- the boundary of c.

The boundary is stored as a (possibly empty) container of entries of the form $(i(c'), \lambda)$, where c' is a generator of dimension $k-1$ such that $\lambda := \langle \partial_k(c), c' \rangle \neq 0$. In such a case, we say that c' is in the boundary of c. The boundary of c is then the linear combination induced by this container. The name "column" comes from the idea that the collection of all columns can be visualized as a matrix. If we decorate the columns of the matrix with extra information (index, value, and dimension), we obtain the data structure from above. We will also write k-column if the column represents a generator of dimension k. Due to the fact that a column is nothing but an encoding of a generator, in the following, with a small abuse of notation we will often use interchangeably the two terms. Given a k-column c, we define the *local pivot* of c as the generator in the boundary of c with maximal index such that $v(c') = v(c)$. If no such generator exists, we simply say that c has no local pivot.

Arguably, the most common case of input data is simplicial complexes. Let K denote a finite simplicial complex. A *bifiltration* is a collection $(K^p)_{p \in \mathbb{R}^2}$ of subcomplexes of K such that $K^p \subseteq K^q$ whenever $p \leq q$. Fixing a simplex $\sigma \in K$, we say that $p \in \mathbb{R}^2$ is *critical* for σ, if $\sigma \in K^p$, but $\sigma \notin K^{p-(\epsilon,0)} \cup K^{p-(0,\epsilon)}$ for every $\epsilon > 0$. The bifiltration is called *h-critical* with $h \geq 1$, if for every $\sigma \in K$, there are at most h critical positions in $\mathbb{R}^2$. For instance, 1-critical means that every simplex in K enters the filtration at a unique minimal value. Such 1-critical bifiltrations can easily be described by a sequence of the form (p, k, Δ) as above by adding one line per simplex which defines its critical position, its dimensions, and its boundary. The case of h-critical bifiltrations can be handled with the following trick (see also [7]). Letting $p_1, \ldots, p_h$ denote the critical positions of σ, sorted by x-coordinate, we introduce h distinct k-generators $c_1, \ldots, c_h$ of the form (p_i, k, Δ), that is, h copies of the same simplex. For two consecutive positions $p_i = (x_i, y_i)$ and $p_{i+1} = (x_{i+1}, y_{i+1})$, we add an additional $(k+1)$-generator at (x_{i+1}, y_i) (which is the smallest point q such that $p_i \leq q$ and $p_{i+1} \leq q$), whose boundary is equal to $c_i - c_{i+1}$.

3 Algorithm

Given a bifiltered chain complex C encoded as a collection of columns, we define the *chunk algorithm* that returns as output a bifiltered chain complex as a collection of columns that is homotopy-equivalent to C and has fewer generators. The algorithm works in three phases:

- *local reduction*;
- *compression*;
- *removal of local columns.*

Phase I: Local reduction. The goal of this phase is to label columns as global, local positive or local negative columns. Initially, all columns are unlabeled. We proceed in decreasing dimensions, from the top-dimensions of C down to 0. For dimension k, the algorithm traverses the k-columns in increasing order with respect to i and performs the following operations on a k-column c. If c is already labeled, do nothing. Otherwise, as long as c has a local pivot and there is a k-column c' with $i(c') < i(c)$ and the same local pivot as c, perform the column addition $c \leftarrow c + \lambda c'$, where λ is chosen such that the local pivot of c disappears. If at the end of this loop, the column c does not have a local pivot, we label the column as global and proceed. Otherwise, we label c as local negative and its local pivot c' as local positive. We call (c', c) a *local pair* in this case. This ends the description of Phase I of the algorithm.

Note that within the local reduction, any column addition of the form $c \leftarrow c + \lambda c'$ implies that $v(c) = v(c')$. Hence, the local reduction operates independently on columns of the same value. We call blocks of columns with the same value *chunks*; hence the name of the algorithm. Operations on one chunk do not affect columns on any other chunk, hence the local reduction phase can be readily invoked in parallel on the chunks of the chain complex.

Finally, note that by proceeding in decreasing dimension, we avoid performing any column additions on local positive columns. That is reminiscent of the *clearing optimization* in the one-parameter version [3, 8].

Phase II: Compression. In the second phase, the algorithm removes local (positive or negative) generators from the boundary of all global columns in the matrix:

For each global k-column c, while the boundary of the column contains a generator that is local positive or local negative, the algorithm picks the local $(k-1)$-generator c' with maximal index.

- if c' is negative, remove c' from the boundary of c;
- if c' is positive, denote c'' as the (unique) local negative k-column with c' as local pivot and perform the column addition $c \leftarrow c + \lambda c''$, where λ is chosen such that c' disappears in the boundary of c.

This ends the description of the compression phase. On termination, all columns in the boundary of a global k-column are global $(k-1)$-columns.

The above process terminates for a column c because the index of the maximal local generator in the boundary of c is strictly decreasing in each step. That is clear for the case that c' is local negative. If c' is local positive, then c' is the generator in the boundary of c'' with the maximal index, so the column addition does not introduce in the boundary of c any generators with a larger index.

Note that the compression of a global column does not affect the result on any other global column. Thus, the phase can be parallelized as well.

Phase III: Removal of local pairs. In this step, the chain complex becomes smaller. The procedure is simple: traverse all columns, and remove all columns labeled as local (positive or negative). Return the remaining (global) columns as resulting chain complex. This finishes the description of the phase and the entire chunk algorithm.

4 Correctness

In this section, we prove that the presented algorithm returns a bifiltered chain complex that is homotopy-equivalent to the input. For that, we define two elementary operations on chain complexes:

Order-preserving column addition. An operation of the form $c \leftarrow c + \lambda c'$ is called *order-preserving* if $v(c') \leq v(c)$. Note that such an operation maintains the property that any generator c'' in the boundary of c satisfies $v(c'') \leq v(c)$, by transitivity of v. We remark that order-preserving column additions are the generalization of left-to-right column additions in the one-parameter case.

Removal of a local pair. Fix a local pair (c_1, c_2), that means, c_1 is a k-column, c_2 is a $(k+1)$-column, $v(c_1) = v(c_2)$ and c_1 is the local pivot of c_2. We call *removal of the local pair* (c_1, c_2) the operation $Del(c_1, c_2)$ which acts on the columns as follows.

- For every $(k+1)$-column c, replace its boundary $\partial_{k+1}(c)$ with $\partial_{k+1}(c) - \lambda^{-1}\mu\partial_{k+1}(c_2)$, where λ and μ are the coefficients of c_1 in $\partial_{k+1}(c_2)$ and in $\partial_{k+1}(c)$, respectively. In particular, after this operation, c_1 disappeared from the boundary of any $(k+1)$-columns.
- For every $(k+2)$-column c, update its boundary by setting the coefficient of c_2 in $\partial_{k+2}(c)$ to 0. Visualizing the chain complex as a matrix, this corresponds to removing the row corresponding to c_2.
- Delete the columns c_1 and c_2.

Note that all column additions performed in the first step are order-preserving because the pair (c_1, c_2) is local, that is, $v(c_1) = v(c_2)$.

We will show at the end of this section that both elementary operations leave the homotopy type the bifiltered chain complex unchanged.

▶ **Theorem 1.** *Let $\bar{C}$ denote the bifiltered chain complex computed by the chunk algorithm from the previous section on an input bifiltered chain complex C. Then, $\bar{C}$ and C are homotopy-equivalent.*

Proof. The idea is to express the chunk algorithm by a sequence of order-preserving column additions and removals of local pairs. Because every column addition in Phase I is between columns of the same value, all column additions are order-preserving. Hence, after Phase I, the chain complex is equivalent to the input.

In Phase II, note that all column additions performed are order-preserving. Indeed, if c' is in the boundary of column c, then $v(c') \leq v(c)$ holds. If c' is local positive, it triggers a column addition of the form $c \leftarrow c + \lambda c''$ with its local negative counterpart c''. Since $v(c') = v(c'')$, $v(c'') \leq v(c)$ as well.

A further manipulation in Phase II is the removal of local negative columns from the boundary of global columns. These removals cannot be directly expressed in terms of the two elementary operations from above. Instead, we define a slight variation of our algorithm: in Phase II, when we encounter a local negative c', we do nothing. In other words, the compression only removes the local positive generators from the boundary c, and keeps local negative and global generators. In Phase III, instead of removing local columns, we perform a removal of a local pair (c_1, c_2) whenever we encounter a local negative column c_2 with local pivot c_1. We call that algorithm the *modified chunk algorithm.* Note that this modified algorithm is a sequence of order-preserving column additions, followed by a sequence of local pair removals, and thus produces a chain complex that is equivalent to the input C.

We argue next that the chunk algorithm and the modified chunk algorithm yield the same output. Since both versions eventually remove all local columns, it suffices to show that they yield the same global columns. Fix an index of a global column, and let c denote the column of that index returned by the original chunk algorithm. Let c^* denote the column of the same index produced by the modified algorithm after the modified Phase II. The difference of c^* and c lies in the presence of local negative generators in the boundary of c^* which have been removed in c. The modified Phase III affects c^* in the following way: when a local pair (c_1, c_2) is removed, the local negative c_2 is, if it is present, removed from the boundary of c^*. There is no column addition during the modified Phase III involving c^* because all local positive columns have been eliminated. Hence, the effect of the modified Phase III on c^* is that all local negative columns are removed from its boundary which turns c^* to be equal to c at the end of the algorithm. Hence, the output of both algorithms is the same, proving the theorem. ◀

We proceed with the proofs that both elementary operations yield homotopy-equivalent bifiltered chain complexes.

Order-preserving column addition. Given two k-columns c_1, c_2 such that $v(c_1) \leq v(c_2)$ and a scalar value $\lambda \in \mathbb{F}$, the algorithm we propose allows for adding λ copies of the boundary of column c_1 to the boundary of column c_2. In this section, we formalize that in terms of modifications of chain complexes and we prove that this operation does not affect multi-parameter persistent homology.

Given a chain complex $C_* = (C_l, \partial_l)$ endowed with a value function $v : C_* \to \mathbb{R}^2$, let us consider two generators c_1, c_2 among the ones of the space of the k-chains C_k such that $\langle c_1, c_2 \rangle = 0$ and $v(c_1) \leq v(c_2)$. Chosen a scalar value $\lambda \in \mathbb{F}$, let us define $\bar{C}_* = (\bar{C}_l, \bar{\partial}_l)$ by setting:

- $\bar{C}_l = C_l$,
- for any $c \in \bar{C}_l$,

$$\bar{\partial}_l(c) = \begin{cases} \partial_k(c) + \lambda\langle c, c_2\rangle \partial_k(c_1) & \text{if } l = k, \\ \partial_{k+1}(c) - \lambda\langle \partial_{k+1}(c), c_2\rangle c_1 & \text{if } l = k+1, \\ \partial_l(c) & \text{otherwise.} \end{cases}$$

As formally proven in the full version of the paper, $\bar{C}_*$ is a chain complex.

Let us define the maps $f_* : C_* \to \bar{C}_*$, $g_* : \bar{C}_* \to C_*$ as follows:

- for any $c \in C_l$,

$$f_l(c) = \begin{cases} c - \lambda\langle c, c_2\rangle c_1 & \text{if } l = k, \\ c & \text{otherwise;} \end{cases}$$

- for any $c \in \bar{C}_l$,

$$g_l(c) = \begin{cases} c + \lambda\langle c, c_2\rangle c_1 & \text{if } l = k, \\ c & \text{otherwise.} \end{cases}$$

The just defined maps enable to prove the following result (see the full version of the paper for detailed proofs).

▶ **Proposition 2.** *C_* and $\bar{C}_*$ are isomorphic via the chain map f_* and its inverse g_*.*

Since the function v is a valid value function for $\bar{C}_*$ inducing a bifiltered chain complex $\bar{C}$ and, for any $p \in \mathbb{R}^2$, the restrictions $f_l^p : C_l^p \to \bar{C}_l^p$ and $g_l^p : \bar{C}_l^p \to C_l^p$ of the maps f_l and g_l, respectively, are well-defined (see the full version of the paper), we are ready to prove the equivalence between the two bifiltered chain complexes C and $\bar{C}$.

▶ **Theorem 3.** *C and $\bar{C}$ are homotopy-equivalent.*

Proof. The previous results enables us to claim that, for any $p, q \in \mathbb{R}^2$ with $p \leq q$, the following diagram (in which horizontal maps are the inclusion maps)

$$\begin{tikzcd} C_l^p \arrow[r] \arrow[d, "f_l^p"] & C_l^q \arrow[d, "f_l^q"] \\ \bar{C}_l^p \arrow[r] & \bar{C}_l^q \end{tikzcd}$$

commutes and the maps f_l^p, f_l^q are isomorphisms. ◀

Removal of a local pair. Given a local pair (c_1, c_2) of columns, the algorithm we propose allows for deleting them from the boundary matrix. In this section, we formalize that in terms of modifications of chain complexes and we prove that this operation does not affect multi-parameter persistent homology.

Given a chain complex $C_* = (C_l, \partial_l)$ endowed with a value function $v : C_* \to \mathbb{R}^2$, let us consider two generators c_1, c_2 among the ones of the space of the k-chains C_k and of the space of the $(k+1)$-chains C_{k+1}, respectively, such that $\lambda := \langle \partial_{k+1}(c_2), c_1\rangle \neq 0$ and $v(c_1) = v(c_2)$. Let us define $\bar{C}_* = (\bar{C}_l, \bar{\partial}_l)$ by setting:

- the space of the l-chains $\bar{C}_l$ as

$$\bar{C}_l = \begin{cases} \{c \in C_k \mid \langle c, c_1\rangle = 0\} & \text{if } l = k, \\ \{c \in C_{k+1} \mid \langle c, c_2\rangle = 0\} & \text{if } l = k+1, \\ C_l & \text{otherwise;} \end{cases}$$

- for any $c \in \bar{C}_l$,

$$\bar{\partial}_l(c) = \begin{cases} \partial_{k+1}(c) - \lambda^{-1}\langle \partial_{k+1}(c), c_1\rangle \partial_{k+1}(c_2) & \text{if } l = k+1, \\ \partial_{k+2}(c) - \langle \partial_{k+2}(c), c_2\rangle c_2 & \text{if } l = k+2, \\ \partial_l(c) & \text{otherwise.} \end{cases}$$

As formally proven in the full version of the paper, $\bar{C}_*$ is a chain complex.

Let us define the maps $r_* : C_* \to \bar{C}_*$, $s_* : \bar{C}_* \to C_*$ as follows:

- for any $c \in C_l$,

$$r_l(c) = \begin{cases} c - \lambda^{-1}\langle c, c_1\rangle \partial_{k+1}(c_2) & \text{if } l = k, \\ c - \langle c, c_2\rangle c_2 & \text{if } l = k+1, \\ c & \text{otherwise;} \end{cases}$$

- for any $c \in \bar{C}_l$,

$$s_l(c) = \begin{cases} c - \lambda^{-1}\langle \partial_{k+1}(c), c_1\rangle c_2 & \text{if } l = k+1, \\ c & \text{otherwise;} \end{cases}$$

The just defined maps enable to prove the following result (see the full version of the paper for detailed proofs).

▶ **Proposition 4.** *The maps r_* and s_* are chain maps which are homotopy-inverse one with respect to the other.*

This latter combined with the fact that the function v is a valid value function for $\bar{C}_*$ inducing a bifiltered chain complex $\bar{C}$ and, for any $p \in \mathbb{R}^2$, the restrictions $r_l^p : C_l^p \to \bar{C}_l^p$, $s_l^p : \bar{C}_l^p \to C_l^p$ of the maps r_l and s_l, as well as the restrictions of the maps ensuring that r_* and s_* are homotopy-inverse, are well-defined (see the full version of the paper), guarantees the homotopy-equivalence between the two bifiltered chain complexes C and $\bar{C}$.

▶ **Theorem 5.** *C and $\bar{C}$ are homotopy-equivalent.*

5 Optimality and complexity

Optimality. Let D be a bifiltered chain complex. For $p \in \mathbb{R}^2$, we define

$$D_*^{<p} := \sum_{q<p} D_*^q,$$

where the sum of chain complexes, analogously to the sum of the vector spaces, is the chain complex spanned by the union of the generators of the summands. Moreover, let η_k^p be the homology map in dimension k induced by the inclusion of $D_*^{<p}$ into D_*^p. We denote the number of variations in the k^{th} homology space occurred at value p as

$$\delta_k^p(D) := \dim \ker \eta_{k-1}^p + \dim \operatorname{coker} \eta_k^p,$$

and the number of k-generators added at value p in D as

$$\gamma_k^p(D) := \dim D_k^p - \dim D_k^{<p}.$$

▶ **Theorem 6.** *Let $\bar{C}$ be the bifiltered chain complex obtained by applying the chunk algorithm to the bifiltered chain complex C. We have that $\delta_k^p(C) = \gamma_k^p(\bar{C})$.*

The detailed proof is given in the full version of the paper and can be summarized as follows. Every global column with value p either destroys a homology class of $H(C^{<p})$, or it creates a new homology class in $H(C^p)$, which is not destroyed by any other column of value p. Hence, each global column contributes a generator to $\ker \iota_{k-1}^p$ or to $\operatorname{coker} \iota_k^p$, where ι_l^p is

the map between the l^{th} homology spaces induced by the inclusion of $C_*^{<p}$ into C_*^p. Local columns do not contribute to either of these two spaces. The result follows from the fact that the number of global columns at value p is precisely the number of generators added at $\bar{C}$.

The next statement shows that our construction is optimal in the sense that any bifiltered chain complex D that is quasi-isomorphic to C must have at least as many generators as $\bar{C}$.

▶ **Theorem 7.** *Any bifiltered chain complex D quasi-isomorphic to C has to add at least $\delta_k^p(C)$ k-generators at value p. I.e., $\delta_k^p(C) \leq \gamma_k^p(D)$.*

The detailed proof is given in the full version of the paper. To summarize it, it is not too hard to see that, for any bifiltered chain complex D,

$$\delta_k^p(D) \leq \gamma_k^p(D)$$

holds. Moreover, the quasi-isomorphism of C and D implies that $\dim\ker \eta_{k-1}^p = \dim\ker \iota_{k-1}^p$ and $\dim\operatorname{coker} \eta_k^p = \dim\operatorname{coker} \iota_k^p$. So,

$$\delta_k^p(C) = \delta_k^p(D)$$

which implies that claim. The equality of the dimensions is formally verified using the Mayer-Vietoris sequence and the 5-lemma to establish an isomorphism from $H_k(C_*^{<p})$ to $H_k(D_*^{<p})$ that commutes with the isomorphism at value p.

Complexity. In order to properly express the time and the space complexity of the proposed algorithm, let us introduce the following parameters. Given a bifiltered chain complex C, we denote as n the number of generators of C, as m the number of chunks (i.e., the number of different values assumed by v), as ℓ the maximal size of a chunk, and as g the number of global columns. Moreover, we assume the maximal size of the support of the boundary of the generators of C as a constant. The latter condition is always ensured for bifiltered simplicial complex of fixed dimension.

▶ **Theorem 8.** *The chunk algorithm has time complexity $O(m\ell^3 \log \ell + g\ell n \log \ell)$ and space complexity $O(n\ell + g^2)$.*

Proof. Due to the similarity between the two algorithms, the analysis of complexity of the proposed algorithm is analogous to the first two steps of the one-parameter chunk algorithm [3]. The additional factor of $\log \ell$ comes from our choice of using priority queues as column type and could be removed by using list representations as in [3].[2]

On the space complexity, during the Phase I, the generators in the boundary of any column can be at most $O(\ell)$. So, $O(n\ell)$ is a bound on the accumulated size of all columns after Phase I. During Phase II, the boundary of any global column can consist of up to n generators, but reduces to g generators at the end of the compression of the column because all local entries have been removed. Hence, the final chain complex has at most g entries in each of its g columns, and requires $O(g^2)$ space. Hence, the total space complexity is $O(n\ell + n + g^2)$, where the second summand is redundant.[3] ◀

[2] We remark, however, that this would result in a performance penalty in practice. See [4].
[3] We remark that this bound only holds for the sequential version of the algorithm. In a parallelized version, it can happen that several compressed columns achieve a size of $O(n)$ at the same time.

6 Implementation and experimental results

We briefly introduce the developed implementation of the chunk algorithm and we evaluate its performance. The current C++ implementation of the chunk algorithm consists of approximately 350 lines of code. It takes as input a finite bifiltered chain complex expressed accordingly to the representation described in Section 2 and encodes each column as a `std::priority_queue`. Even if the chunk algorithm permits a parallelization in shared memory, this first implementation does not exploit this potential. Our experiments have been performed on a configuration Intel Xeon CPU E5-1650 v3 at 3.50 GHz with 64 GB of RAM.

For testing the implemented chunk algorithm, we have compared its performances with the simplification process based on discrete Morse theory proposed in [17] (DMT-based algorithm) whose implementation is publicly available [10]. We remark once more that the DMT-based approach yields a somewhat richer output than solely a simplified chain complex with the same homotopy; however, we only compare the size of the resulting structure (in terms of the number of generators per chain group) in this experimental comparison.

In our experiments, we have considered both synthetic and real datasets represented as simplicial complexes. The latter ones are courtesy of the AIM@SHAPE data repository [9]. Most of the datasets are of dimension 2 or 3 and are embedded in a 3D environment. For our experiments, we have adopted as filtering functions the ones obtained by extending to all the simplices of the complex the x and the y coordinates assigned to the vertices of the datasets. Table 1 displays the achieved results.

The data sets are given in off file format, that means, as a list of triangles specified by boundary vertices. In order to apply our algorithm, we first have to convert the data into a boundary matrix representation. Hence, we enlist in Table 1 the *preparation time* to create the boundary matrix and the *simplification time* to perform our chunk algorithm. In contrast, the DMT-based approach avoids the initial construction of the boundary matrix but transforms the input into a different data structure before starting its simplification step. We also list the running times of these separate steps in Table 1. In both cases, we do not list the time for reading the input file into memory and writing the output structure on disk.

The column *Size* in Table 1 collects the sizes (in terms of the number of simplices/columns) of the bifiltered chain complexes in input and in output. The compression factor achieved by both algorithms varies between 9 and 23. As average, the reduced chain complex in output is approximately 13 times smaller than the original one. Despite of the theoretical advantage that our approach yields an optimal size in all situations, the size of the output returned by the two algorithms is nearly the same in all listed examples. A difference can be noticed just for datasets including shapes that can be considered as "pathological". For instance, an example of such a dataset is the conification of a dunce hat.

The column *Time* shows the computation times of both algorithms. In all tested examples, the chunk algorithm takes a small fraction of time with respect to the DMT-based approach. We also see that the creation of the boundary matrix (preparation step) takes more time than the chunk algorithm (simplification step) by a factor of 2 to 3, but even these two steps combined are faster than the simplification step of the DMT-based approach. It is worth investigating whether the boundary matrix creation becomes a more severe bottleneck for other datasets (e.g., in higher dimension), where the DMT-based approach might have advantages by not creating the boundary matrix explicitly.

The column *Memory Usage* shows the memory consumption of both approaches. The chunk algorithm does not need to encode auxiliary structures like the discrete Morse gradient stored in the DMT-based approach resulting in a slightly less amount of required space.

Table 1 Results and performances obtained by the chunk and the DMT-based algorithms. The columns from left to right indicate: the name of the dataset (*Dataset*), the number of cells before and after the simplification algorithms (*Size*), the time (*Time*), expressed in seconds, needed to perform the preparation step (*Prep.*) and the simplification step (*Simpl.*), the maximum peak of memory, expressed in GB, required by the two algorithms (*Memory Usage*).

Dataset	Size		Time (sec.)				Memory Usage (GB)	
			Chunk		DMT			
	Input	Output	Prep.	Simpl.	Prep.	Simpl.	Chunk	DMT
Eros	2.9 M	202 K	1.7	0.8	2.7	15.8	0.36	0.46
Donna	3.0 M	217 K	1.8	0.8	2.8	16.9	0.38	0.48
Chinese Dragon	3.9 M	321 K	2.5	1.1	3.9	22.3	0.52	0.64
Circular Box	4.2 M	365 K	2.9	1.2	4.3	24.0	0.68	0.68
Ramesses	5.0 M	407 K	3.4	1.3	5.5	29.4	0.68	0.81
Pensatore	6.0 M	369 K	3.8	1.6	6.8	34.3	0.76	0.97
Raptor	6.0 M	260 K	4.4	1.7	5.4	32.3	0.73	0.93
Neptune	12.0 M	893 K	8.4	4.4	14.9	69.2	1.52	1.94
Cube 1	590 K	67 K	0.5	0.3	0.7	3.2	0.09	0.10
Cube 2	2.4 M	264 K	1.8	1.1	2.6	13.1	0.35	0.40
Cube 3	9.4 M	1.0 M	7.6	4.8	11.0	53.1	1.37	1.58
Cube 4	37.7 M	4.2 M	31.9	19.4	44.9	216.0	5.50	6.32

Depending on the dataset, the chunk approach requires between 0.09 and 5.50 GB of memory and it is, on average, 1.2 times more compact than the DMT-based one.

In summary, the chunk algorithm satisfies a stronger optimality condition than [17], but still is an order of magnitude faster and uses comparable memory.

7 Conclusion

We have presented a pre-processing procedure for improving the computation of multi-parameter persistent homology and we have provided theoretical and experimental evidence of its effectiveness. In the future, we want to further improve the proposed strategy as follows. First, we would like to develop a parallel implementation with shared memory of the chunk algorithm and compare its performances with the ones obtained by the current version. Similarly to [17], we also want to evaluate the impact of the chunk algorithm for the computation of the persistence module and of the persistence space.

Our optimality criterion is formulated for the class of chain complexes quasi-isomorphic to the input. It is possible to produce counterexamples showing that this statement cannot be generalized to the class of the chain complexes homology-equivalent to the input. Nevertheless, we want to further investigate the optimality properties satisfied by our algorithm and compare our algorithm with the minimal presentation algorithm for persistence modules from RIVET.

References

1 M. Allili, T. Kaczynski, and C. Landi. Reducing complexes in multidimensional persistent homology theory. *Journal of Symbolic Computation*, 78:61–75, 2017. doi:10.1016/j.jsc.2015.11.020.

2 M. Allili, T. Kaczynski, C. Landi, and F. Masoni. Acyclic partial matchings for multidimensional persistence: algorithm and combinatorial interpretation. *Journal of Mathematical Imaging and Vision*, pages 1–19, 2018.

3 U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: computing persistent homology in chunks. In *Topological methods in data analysis and visualization III*, pages 103–117. Springer, 2014.

4 U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. Phat - persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017.

5 G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.

6 A. Cerri and C. Landi. The persistence space in multidimensional persistent homology. In R. Gonzalez-Diaz, M.-J. Jimenez, and B. Medrano, editors, *Discrete Geometry for Computer Imagery*, pages 180–191. Springer Berlin Heidelberg, 2013.

7 W. Chachólski, M. Scolamiero, and F. Vaccarino. Combinatorial presentation of multidimensional persistent homology. *Journal of Pure and Applied Algebra*, 221(5):1055–1075, 2017.

8 C. Chen and M. Kerber. Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry*, volume 11, pages 197–200, 2011.

9 Digital Shape WorkBench. AIM@SHAPE project, 2006. URL: `http://visionair.ge.imati.cnr.it`.

10 F. Iuricich. FG-multi, 2018. URL: `http://github.com/IuricichF/fg_multi`.

11 F. Iuricich, S. Scaramuccia, C. Landi, and L. De Floriani. A discrete Morse-based approach to multivariate data analysis. In *SIGGRAPH ASIA 2016 Symposium on Visualization*, pages 5–12. ACM, 2016.

12 K. P. Knudson. A refinement of multi-dimensional persistence. *Homology, Homotopy and Applications*, 10(1):259 – 281, 2008.

13 M. Lesnick and M. Wright. Computing minimal presentations of bipersistence modules in cubic time. In preparation.

14 M. Lesnick and M. Wright. Interactive visualization of 2-D persistence modules. *arXiv preprint*, 2015. `arXiv:1512.00180`.

15 E. Miller. Data structures for real multiparameter persistence modules. *arXiv preprint*, 2017. `arXiv:1709.08155`.

16 S. Scaramuccia. *Computational and theoretical issues of multiparameter persistent homology for data analysis.* PhD thesis, University of Genova, Italy, 2018. URL: `http://hdl.handle.net/11567/929143`.

17 S. Scaramuccia, F. Iuricich, L. De Floriani, and C. Landi. Computing multiparameter persistent homology through a discrete Morse-based approach. *arXiv preprint*, 2018. `arXiv:1811.05396`.