

Chunk Reduction for Multi-Parameter Persistent Homology

Ulderico Fugacci Michael Kerber

Computational Geometry Week, Portland, June 19, 2019



FWF

Der Wissenschaftsfonds.



Multi-Parameter Persistent Homology

Multi-parameter persistence describes the *changes in homology* of datasets evolving along *two or more (independent) scale parameters*

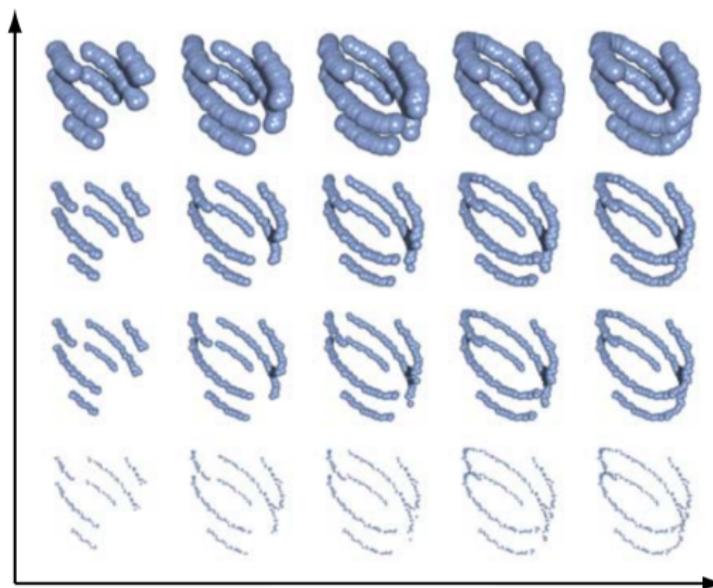


Image courtesy of [Carlsson & Zomorodian 2009]

Multi-Parameter Persistent Homology

Formally:

Given a d -filtered chain complex $C := (C_*^p)_{p \in \mathbb{R}^d}$,

the *multi-parameter persistence k^{th} module* $H_k(C)$ consists of:

- A collection of *homology spaces* $H_k(C_*^p)$ for $p \in \mathbb{R}^d$
- A collection of *linear maps* $\iota_k^{p,q} : H_k(C_*^p) \rightarrow H_k(C_*^q)$ induced by the inclusion maps at k -chain level for $p, q \in \mathbb{R}^d$ and $p \leq q$

Multi-parameter persistent homology provides a *stable and discriminative information* **but...**

Multi-Parameter Persistent Homology

But...

Computations scale badly with the size of the input complex

Our Goal

Design and implement a *simplification process*
drastically reducing the size of the input complex while
preserving its multi-parameter persistent homology

Outline

- *Multi-Chunk Algorithm*
- *Optimality Result*
- *Experimental Evaluation*
- *Conclusions and Future Developments*

Multi-Chunk Algorithm

Inspired by the chunk algorithm for persistent homology [Bauer et al. 2014],

We propose a *reduction algorithm* such that

given a d -filtered chain complex C returns a d -filtered chain complex \bar{C}

- *drastically smaller* than C
- *homology-equivalent* to C

C and \bar{C} are *homology-equivalent* if, for any $k \in \mathbb{N}$ and any $p, q \in \mathbb{R}^d$ with $p \leq q$,

$H_k(C_*^p)$ and $H_k(\bar{C}_*^p)$ are *isomorphic* via a map φ_k^p and each diagram

$$\begin{array}{ccc} H_k(C_*^p) & \longrightarrow & H_k(C_*^q) \\ \downarrow \varphi_k^p & & \downarrow \varphi_k^q \\ H_k(\bar{C}_*^p) & \longrightarrow & H_k(\bar{C}_*^q) \end{array}$$

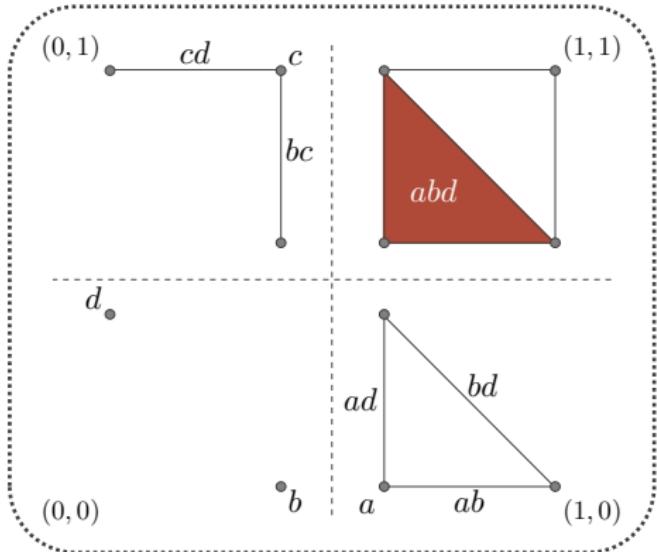
commutes where horizontal maps are induced by inclusion maps

Multi-Chunk Algorithm

Initialization:

Given a d -filtered simplicial complex,

each k -simplex σ can be represented by a k -column encoding:



- Boundary of σ
- Dimension k
- Filtration value $v(\sigma)$
- Index $i(\sigma)$

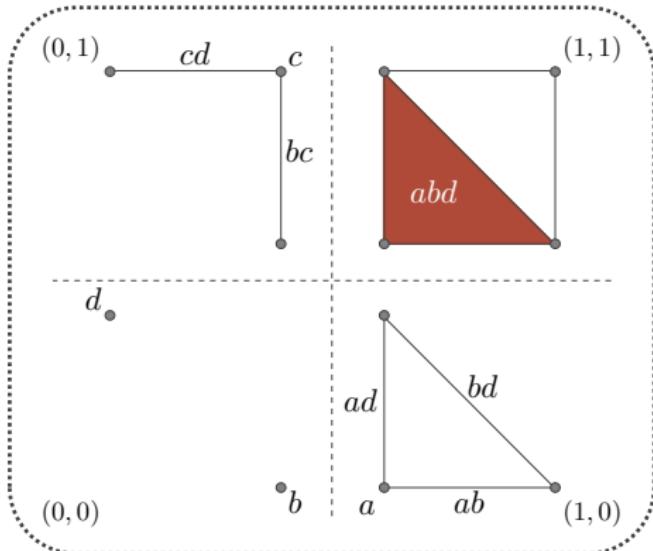
where i is a *total order* on the simplices compatible with the filtering function v

Multi-Chunk Algorithm

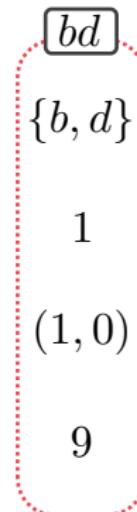
Initialization:

Given a d -filtered simplicial complex,

each k -simplex σ can be represented by a k -column encoding:



- Boundary of σ
- Dimension k
- Filtration value $v(\sigma)$
- Index $i(\sigma)$



where i is a *total order* on the simplices compatible with the filtering function v

Multi-Chunk Algorithm

i		(0, 0)		(0, 1)		(1, 0)				(1, 1)	
	σ	1	2	3	4	5	6	7	8	9	10
(0, 0)	1	b				1			1		1
	2	d					1			1	1
	3	c				1	1				
(0, 1)	4	bc									
	5	cd									
	6	a							1	1	
	7	ab									1
	8	ad									1
	9	bd									1
(1, 0)	10	abd									

Multi-Chunk Algorithm

Multi-chunk algorithm offers an immediate *parallelization* scheme in shared memory and it consists of *three phases*:

I - Local Reduction

II - Compression

III - Removal of Local Pairs

Local Pivot:

A column σ has a *local pivot* iff the element τ with *maximal index* in the boundary of σ is such that $v(\tau)=v(\sigma)$

Multi-Chunk Algorithm

i		(0, 0)		(0, 1)		(1, 0)				(1, 1)	
	σ	1	2	3	4	5	6	7	8	9	10
(0, 0)	1	b				1			1		1
	2	d					1			1	1
	3	c				1	1				
(0, 1)	4	bc									
	5	cd									
	6	a							1	1	
	7	ab									1
	8	ad									1
	9	bd									1
(1, 0)	10	abd									

Multi-Chunk Algorithm

Phase I: Local Reduction

Proceed in *decreasing dimension k* and traverse the k -columns in *increasing order w.r.t. i*

*Goal: Label the columns as **local** or **global***

Given an *unlabeled* k -column σ , while

- σ has a local pivot and
- there is a k -column σ' with the same local pivot and $i(\sigma) < i(\sigma')$

perform the *column addition* $\sigma \leftarrow \sigma + \lambda\sigma'$ (λ is s.t. the local pivot of σ disappears)

If, at the end of the loop, the column σ *does not have a local pivot*

- σ is labeled as **global**

otherwise,

- σ is labeled as **local negative** and its local pivot as **local positive**

Multi-Chunk Algorithm

i		(0, 0)		(0, 1)		(1, 0)				(1, 1)	
	σ	1	2	3	4	5	6	7	8	9	10
(0, 0)	1	b				1			1		1
	2	d					1			1	1
	3	c				1	1				
(0, 1)	4	bc									
	5	cd									
	6	a							1	1	
	7	ab									1
	8	ad									1
	9	bd									1
(1, 0)	10	abd									

Multi-Chunk Algorithm

i		(0, 0)	(0, 1)	(1, 0)	(1, 1)						
	σ	1	2	3	4	5	6	7	8	9	10
(0, 0)	1	b			1			1		1	
	2	d				1			1	1	
	3	c			1	1					
(0, 1)	4	bc									
	5	cd									
	6	a						1	1		
	7	ab								1	
	8	ad								1	
	9	bd								1	
(1, 0)	10	abd									

Multi-Chunk Algorithm

i		(0, 0)		(0, 1)		(1, 0)				(1, 1)	
	σ	1	2	3	4	5	6	7	8	9	10
(0, 0)	1	b				1			1		1
	2	d					1			1	1
	3	c				1	1				
(0, 1)	4	bc									
	5	cd									
	6	a							1	1	
	7	ab									1
	8	ad									1
	9	bd									1
(1, 1)	10	abd									

Multi-Chunk Algorithm

i		(0, 0)		(0, 1)		(1, 0)				(1, 1)	
	σ	1	2	3	4	5	6	7	8	9	10
(0, 0)	1	b				1	1		1		1
	2	d					1			1	1
	3	c				1					
(0, 1)	4	bc									
	5	cd									
	6	a							1	1	
	7	ab									1
	8	ad									1
	9	bd									1
(1, 1)	10	abd									

Multi-Chunk Algorithm

i		(0, 0)	(0, 1)	(1, 0)	(1, 1)					
σ	b	d	c	bc	cd	a	ab	ad	bd	abd
(0, 0)	1	b		1	1		1	1	1	
	2	d			1			1	1	
	3	c		1						
(0, 1)	4	bc								
	5	cd								
	6	a					1			
	7	ab						1		
(1, 0)	8	ad							1	
	9	bd							1	
(1, 1)	10	abd								

Multi-Chunk Algorithm

Phase II: Compression

*Goal: Simplify the boundary of the **global** columns*

Given a *global* k -column σ , while

- the boundary of σ contains local positive or negative elements

pick the local $(k-1)$ -column τ in the boundary of σ with maximal index i

If τ is **local negative**,

- remove τ from the boundary of σ

If τ is **local positive**

- perform the column addition $\sigma \leftarrow \sigma + \lambda\sigma'$

where σ' is s.t. (τ, σ) is a local pair and λ is such that τ disappears

Multi-Chunk Algorithm

i		(0, 0)	(0, 1)	(1, 0)	(1, 1)						
	σ	1 <i>b</i>	2 <i>d</i>	3 <i>c</i>	4 <i>bc</i>	5 <i>cd</i>	6 <i>a</i>	7 <i>ab</i>	8 <i>ad</i>	9 <i>bd</i>	10 <i>abd</i>
(0, 0)	1	<i>b</i>			1	1		1	1	1	
	2	<i>d</i>				1			1	1	
	3	<i>c</i>			1						
(0, 1)	4	<i>bc</i>									
	5	<i>cd</i>									
	6	<i>a</i>						1			
	7	<i>ab</i>							1		
	8	<i>ad</i>							1		
	9	<i>bd</i>								1	
(1, 0)	10	<i>abd</i>									

Multi-Chunk Algorithm

i		(0, 0)	(0, 1)	(1, 0)	(1, 1)					
σ	b	d	c	bc	cd	a	ab	ad	bd	abd
(0, 0)	b			1	1		1	1	1	
2	d				1			1	1	
3	c			1						
(0, 1)	bc									
5	cd									
6	a							1		
7	ab									
(1, 0)	ad								1	
9	bd									1
(1, 1)	abd									

Multi-Chunk Algorithm

Phase III: Removal of Local Pairs

Traverse the columns and remove all the columns labeled as **local positive** or **local negative**

Theorem

Remaining columns generate a d -filtered chain complex having the **same multi-parameter persistent homology** as the input (they are homotopy-equivalent)

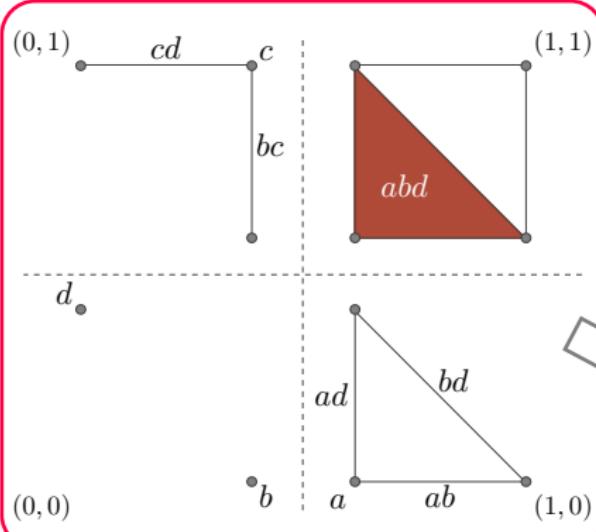
Multi-Chunk Algorithm

i		(0, 0)		(0, 1)		(1, 0)		(1, 1)			
	σ	1	2	3	4	5	6	7	8	9	10
(0, 0)	1	b				1		1	1	1	
	2	d					1		1	1	
	3	c									
(0, 1)	4	bc									
	5	cd									
	6	a									
	7	ab									
(1, 0)	8	ad								1	
	9	bd								1	
(1, 1)	10	abd									

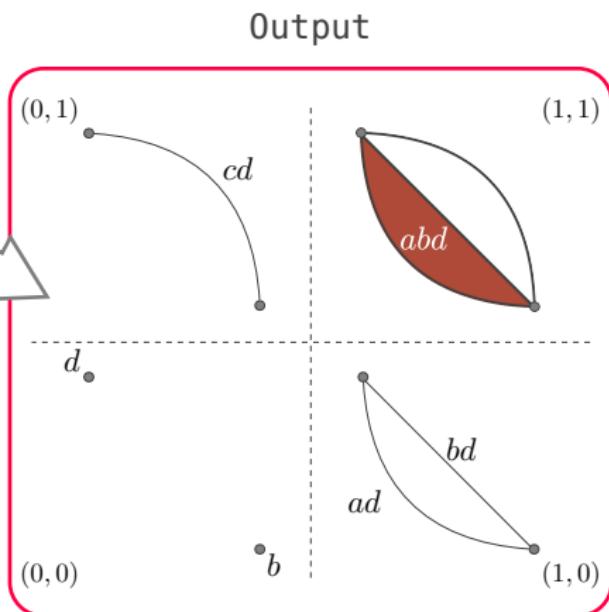
Dashed red lines indicate vertical connections between columns:

- Column 1 connects to Column 3.
- Column 2 connects to Column 4.
- Column 3 connects to Column 5.
- Column 4 connects to Column 6.
- Column 5 connects to Column 7.
- Column 6 connects to Column 8.
- Column 7 connects to Column 9.
- Column 8 connects to Column 10.

Multi-Chunk Algorithm



Input



Output

Multi-Chunk Algorithm

Remarks

- In Phase I, to *proceed in decreasing dimension* avoids performing any column additions on local positive columns
- Algorithm operates *independently* on columns of the same value (called *chunks*)

Proposition

Multi-chunk algorithm has *time complexity* $O(m\ell^3 \log \ell + gnl \log \ell)$ and *space complexity* $O(n\ell + g^2)$ where:

- n is the size of the input complex
- m is the number of chunks
- ℓ is the maximal size of a chunk
- g is the number of global columns

Optimality Result

- $C_*^{<p} := \sum_{q < p} C_*^q$
- $\gamma_k^p(C) := \dim C_k^p - \dim C_k^{<p}$
- $\delta_k^p(C) := \dim(\ker \iota_{k-1}^p) + \dim(\text{coker } \iota_k^p)$

Proposition

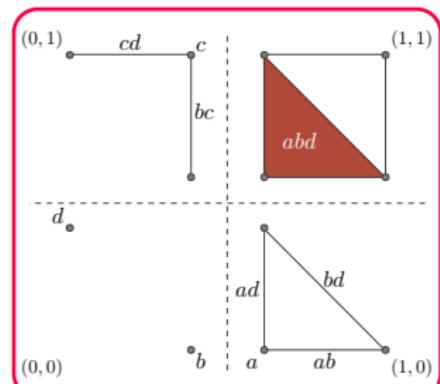
For any d -filtered chain complex D quasi-isomorphic to the input complex C ,

$$\gamma_k^p(D) \geq \delta_k^p(C)$$

Theorem

Let \bar{C} be the d -filtered chain complex obtained as output, then

$$\gamma_k^p(\bar{C}) = \delta_k^p(C)$$



$$\gamma_1^{(1,0)}(C) = 3 > 2 = \delta_1^{(1,0)}(C)$$

Optimality Result

- $C_*^{<p} := \sum_{q < p} C_*^q$
- $\gamma_k^p(C) := \dim C_k^p - \dim C_k^{<p}$
- $\delta_k^p(C) := \dim(\ker \iota_{k-1}^p) + \dim(\text{coker } \iota_k^p)$

Proposition

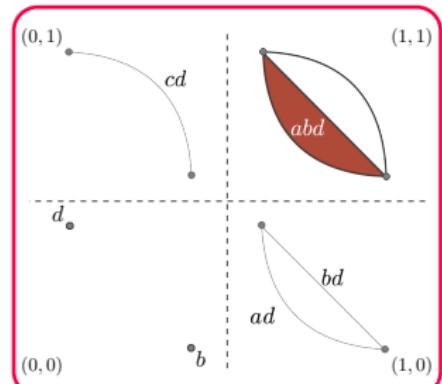
For any d -filtered chain complex D quasi-isomorphic to the input complex C ,

$$\gamma_k^p(D) \geq \delta_k^p(C)$$

Theorem

Let \bar{C} be the d -filtered chain complex obtained as output, then

$$\gamma_k^p(\bar{C}) = \delta_k^p(C)$$



$$\gamma_1^{(1,0)}(\bar{C}) = 2 = \delta_1^{(1,0)}(C)$$

Experimental Evaluation

We have experimentally compared the performances of our approach with the simplification process based on *discrete Morse theory* proposed in [Scaramuccia et al. 2018]

Dataset	Size		Time (sec.)				Memory Usage (GB)	
	Input	Output	Chunk		DMT		Chunk	DMT
			Prep.	Simpl.	Prep.	Simpl.		
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

Conclusions and Future Developments

In Summary:

We have presented a *pre-processing procedure* for improving the computation of multi-parameter persistent homology and we have provided *theoretical and experimental evidences of its effectiveness*

Future Directions:

- Develop a *parallel implementation* with shared memory of the multi-chunk algorithm
- Evaluate the impact of the chunk algorithm for the computation of the **persistence module** and of the *persistence space*
- Compare the proposed strategy with the *minimal presentation algorithm* for persistence modules from RIVET [Lesnick & Wright 2019]

Thank you

*Ulderico Fugacci
TU Graz, Institute of Geometry*

C and \bar{C} are *homology-equivalent* if, for any $k \in \mathbb{N}$ and any $p, q \in \mathbb{R}^d$ with $p \leq q$,

$H_k(C_*^p)$ and $H_k(\bar{C}_*^p)$ are *isomorphic* via a map φ_k^p and each diagram

$$\begin{array}{ccc} H_k(C_*^p) & \longrightarrow & H_k(C_*^q) \\ \downarrow \varphi_k^p & & \downarrow \varphi_k^q \\ H_k(\bar{C}_*^p) & \longrightarrow & H_k(\bar{C}_*^q) \end{array}$$

commutes where horizontal maps are induced by inclusion maps

C and \bar{C} are *quasi-isomorphic* if the isomorphisms of the above diagram are induced by a collection of chain maps f_*^p such that each diagram

$$\begin{array}{ccc} C_*^p & \longrightarrow & C_*^q \\ \downarrow f_k^p & & \downarrow f_k^q \\ \bar{C}_*^p & \longrightarrow & \bar{C}_*^q \end{array}$$

commutes where horizontal maps are the inclusion maps

C and \bar{C} are *homotopy-equivalent* if there exist two collections of chain maps f_*^p and g_*^p such that they are *homotopy-inverse* (i.e., $g_*^p f_*^p \simeq \text{id}_{C_*^p}$ and $f_*^p g_*^p \simeq \text{id}_{\bar{C}_*^p}$) and each diagram

$$\begin{array}{ccc} C_*^p & \longrightarrow & C_*^q \\ f_k^p \downarrow \uparrow g_k^p & & f_k^q \downarrow \uparrow g_k^q \\ \bar{C}_*^p & \longrightarrow & \bar{C}_*^q \end{array}$$

commutes where horizontal maps are the inclusion maps



*Filtered chain complexes which are
homology-equivalent but not quasi-isomorphic*

