Pruning vineyards

Updating barcodes and representative cycles by removing (and inserting) simplices

`arxiv:2312.03925`
accompanying paper

Barbara Giunti
SUNY Albany

Jānis Lazovskis
University of Latvia

`bitbucket:phat-sirup`
accompanying code

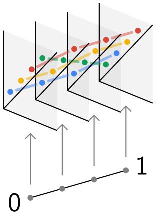
October 14, 2025 / ComPer 2025: Workshop on Computational Topology
Slides online at: `jlazovskis.com/talks`

Funding for JL: Activity 1.1.1.9 "Post-doctoral Research" of the Specific Objective 1.1.1 "Strengthening research and innovative capacities and introduction of advanced technologies in the common R&D system" of the European Union's Cohesion Policy Programme for 2021-2027 research application No 1.1.1.9/LZP/1/24/125 "Efficient topological signatures for representation learning in medical imaging"

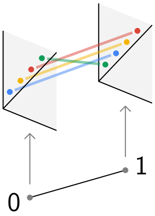
Līdzfinansē Eiropas Savienība
2027 Nacionālais attīstības plāns
Latvijas Zinātnes padome
LATVIJAS UNIVERSITĀTE
PostDoc Latvia

Vineyards

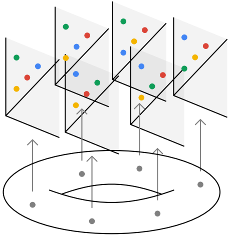
Vines and **vineyards** were introduced to capture a change in the persistence diagram coming from a change in the underlying data. Has since been generalized in different ways.



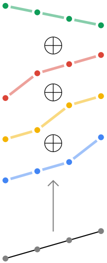
vines and vineyards:
Cohen–Steiner, Edelsbrunner, Morozov (2006)



moves: Busaryev, Dey, Wang (2010)
move schedules:
Piekenbrock, Perea (2024)



persistence diagram bundles:
Hickok (2022)



vineyard modules:
Turner (2023)

These approaches are about **swapping** the order of simplices in a filtration.
What about other possible changes, like **inserting** or **removing** simplices?

Dynamic persistence

Motivation:

- Potential to reuse costly computations
- Suited to non-static data (changing resolution, zooming in, moving around)
- Get a better understanding of class representatives

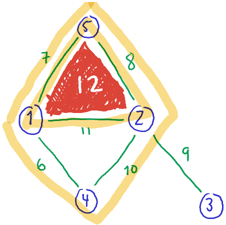
Computational reality: Given a simplicial complex with n simplices:

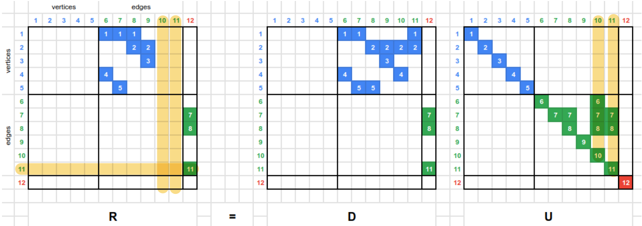
- $O(n^3)$ complexity to compute PH with standard algorithm (mat.mul. by Morozov, Skraba)
- Massive speedups with clever code, homological observations, dimensional restrictions
- Most **dynamic data** is read into / out of the computer as a sequence of static data points (changes inferred, not given directly). Makes sense to rerun everything with new data.

But still: Persistence cycles (and so persistence diagrams) are hard to track this way.
Would be great if: There was a PH implementation providing an **efficient**, **user-friendly** interface with **access to PD** at intermediate steps.

Example

Consider a simplex-wise filtration on a simplicial complex. Consider 1-cycles and their reps.





Inserting: How are 1-classes affected if edge $\{4, 3\}$ is insterted at position 10?

- One non-trivial 1-class appears. Is representative $\{4, 2, 3\}$ or $\{4, 1, 5, 2, 3\}$?

Removing: How are 1-classes affected if edge 8 is removed?

- One trivial 1-class disappears. Is other non-trivial representative different or the same?

Inserting simplices: easy

Similarly to the **standard barcode algorithm**, add matching pivot columns left-to-right.

Algorithm 1: SBA - Standard Barcode Algorithm [ELZ00]

Input: Boundary matrix D, identity matrix I
Output: Reduced matrix R, operations matrix U

$R \leftarrow D$
$U \leftarrow \mathrm{I}$
for $j = 1, \ldots, n$
  while *there exists* $j' < j$ *for which* $\mathrm{piv}\,(R\,[j']) = \mathrm{piv}\,(R\,[j]) \neq 0$
    add $R\,[j']$ to $R\,[j]$
    add $U\,[j']$ to $U\,[j]$
return R, V

Algorithm 1: Inserting a simplex

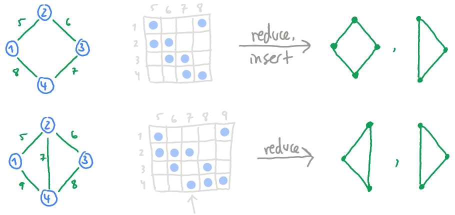
Input: Reduced matrix R, operations matrix U, list of simplices to insert L
Output: Updated reduced matrix P, updated operations matrix V

$P \leftarrow R$
$V \leftarrow U$
for $\sigma_j \in L$
  insert row and column at index j in R, with nonzero values at $R[i, j]$, $\sigma_i \in \partial\sigma_j$
  insert row and column at index j in U, with one nonzero value at $U[j, j]$
  $r \leftarrow \mathrm{piv}\,(\sigma_j)$
  while *there exists* $k \neq j$ *with* $R[r, j] = R[r, k] \neq 0$
    if $k < j$
      add column k to column j in P and in V
      $r \leftarrow \mathrm{piv}\,(P\,[j])$
    else if $j < k$
      add column j to column k in P and in V
      $r \leftarrow \mathrm{piv}\,(P\,[k])$
return P, V

Complexity: for one insertion is $O(n^2)$, vs $O((n+1)^3)$ for computing from scratch.

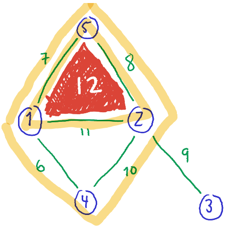
Careful: The representative cycles do not match up with computation from scratch.

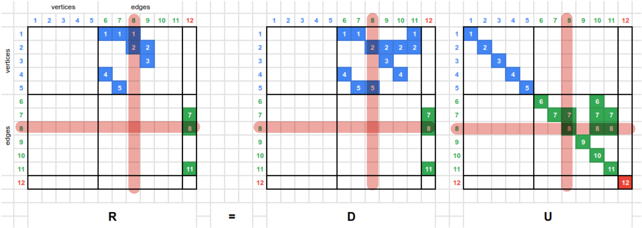
This is still work in progress.



Removing simplices: motivating example

Back to example: To remove simplex 8 (after 12), “undo” the column ops it was involved in.



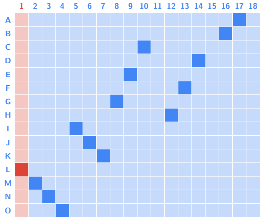


1. Check row 8 in matrix U to see in which representative cycles it appears.
   - ▶ Add column 8 to columns 10, 11 in R, U.
2. Ensure that R is reduced.
   - ▶ Add column 10 to column 11 in R, U.

Observation: Column 8 was “added twice” to column 11, cancelling itself out in the end. Only one column operation (8 to 10) was necessary.

Removing simplices: in general

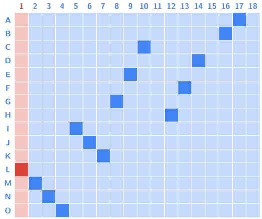
- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).

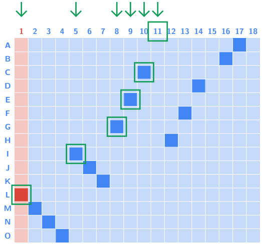


Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).

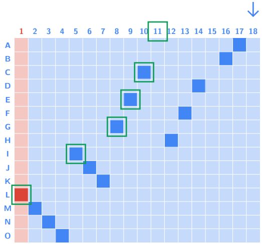


Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



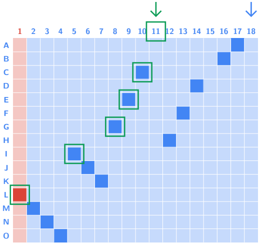
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 18 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



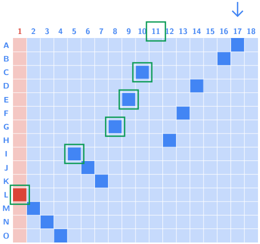
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 18 is reduced: add column 11

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



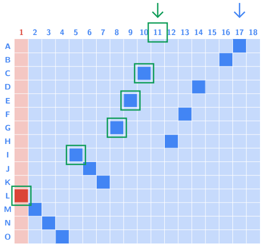
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 17 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



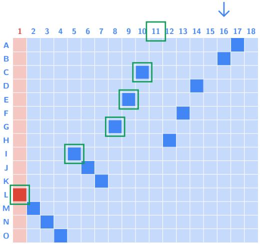
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 17 is reduced: add column 11

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



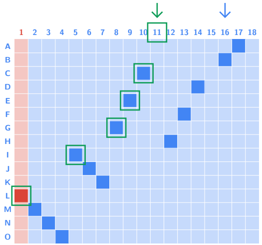
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 16 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



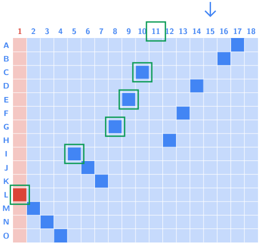
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 16 is reduced: add column 11

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



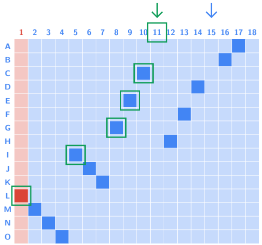
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 15 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



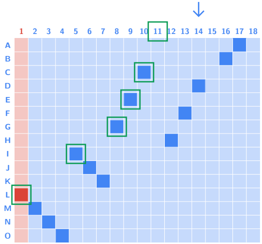
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 15 is reduced: add column 11

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



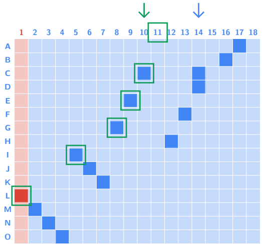
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 14 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



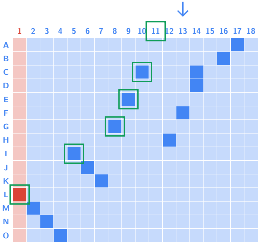
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 14 is reduced: add column 10

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



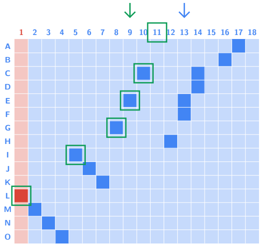
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 13 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



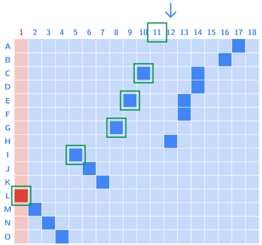
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 13 is reduced: add column 9

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



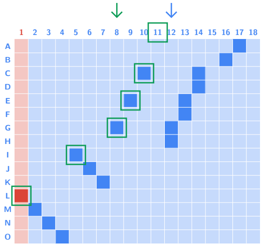
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 12 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



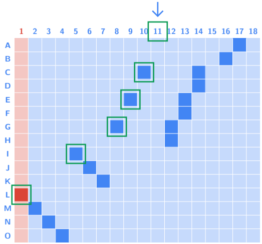
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 12 is reduced: add column 8

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



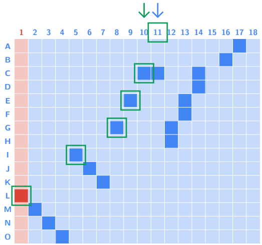
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 11 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



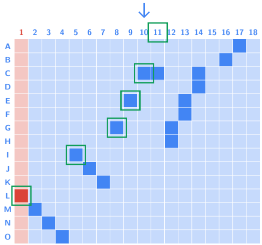
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 11 is reduced: add column 10

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



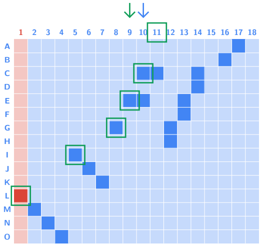
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 10 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



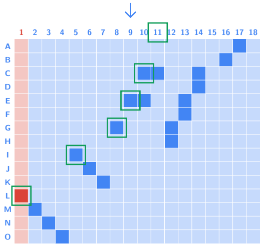
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 10 is reduced: add column 9

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



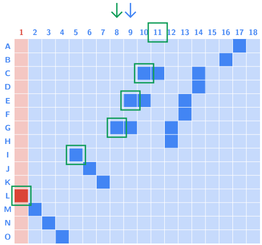
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 9 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



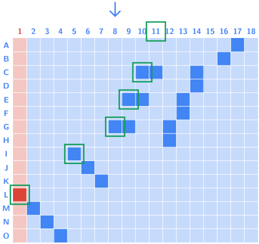
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 9 is reduced: add column 8

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



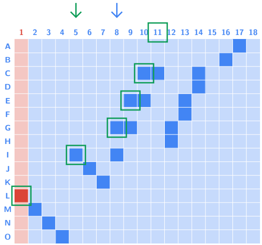
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 8 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



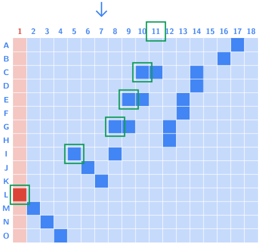
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 8 is reduced: add column 5

Removing simplices: in general

- Each “affected column” needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



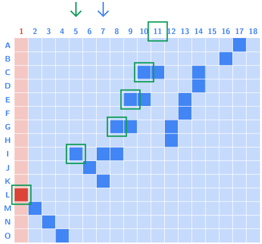
Consider the submatrix of R that contains only the columns “affected” by the removal of column 1.

Step 1: Identify the “left-most highest” pivots.

Step 2: Ensure column 7 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



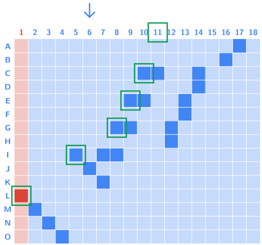
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 7 is reduced: add column 5

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



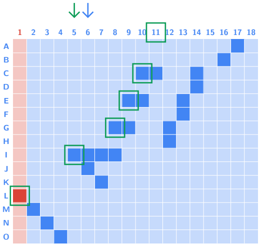
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 6 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



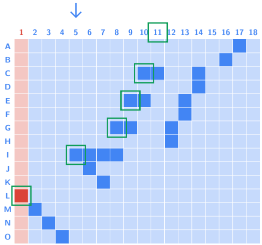
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 6 is reduced: add column 5

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



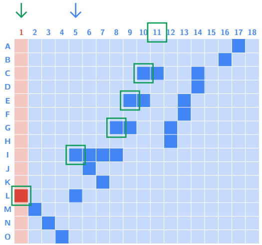
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 5 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



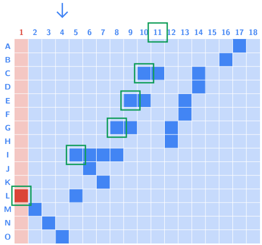
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 5 is reduced: add column 1

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



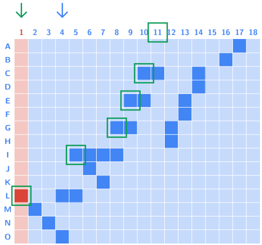
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 4 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



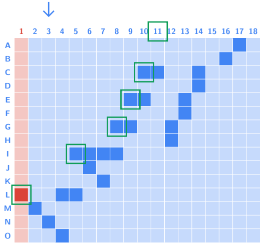
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 4 is reduced: add column 1

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



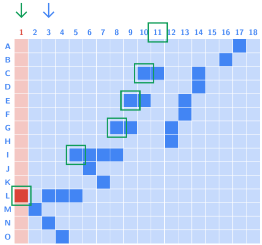
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 3 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



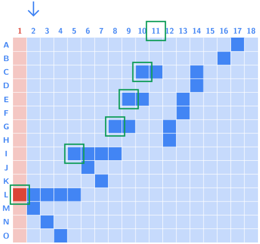
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 3 is reduced: add column 1

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



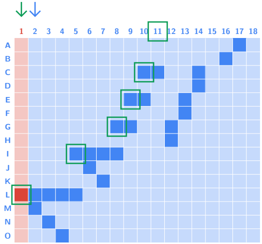
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 2 is reduced:

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



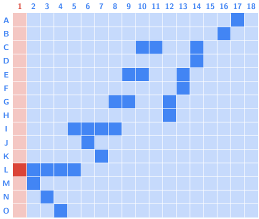
Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure column 2 is reduced: add column 1

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

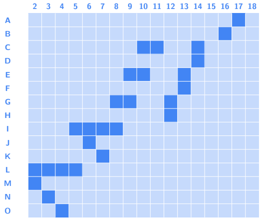
Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure all columns are reduced

Step 3: Drop column 1

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

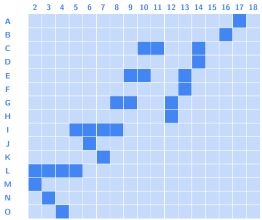
Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure all columns are reduced

Step 3: Drop column 1

Removing simplices: in general

- Each "affected column" needs only one column addition to get it to where it should be.
- The column which should be added to it is determined by the left-most column with a pivot above its pivot (or below, if it itself is a left-most pivot column).



Consider the submatrix of R that contains only the columns "affected" by the removal of column 1.

Step 1: Identify the "left-most highest" pivots.

Step 2: Ensure all columns are reduced

Step 3: Drop column 1

Each removal performs m column additions in R, U (ensuring updated rep. cycles)

Complexity: for one removal is $O(mn)$, vs $O((n-1)^3)$ for computing from scratch.

SiRUP: Simplicial Removal Update Procedure

- Proof of correctness shows equivalence with "naive" method, on sets A, B.
- (Strong) induction on these "affected" simplices.

Algorithm 2: SiRUP- Simplicial Removal Update Procedure

Input: Reduced matrix R, operations matrix U, list of simplices to be removed L
Output: Updated reduced matrix P, updated operations matrix V

$P \leftarrow R$
$V \leftarrow U$
for $\sigma_j \in L$
  $A = \{i \ : \ V[j,i] \neq 0, \ i \neq j\} = \{A_1, A_2, \dots\}$ in increasing order
  $B = \{A_0 = j\} \cup \{A_i \in A \ : \ \mathrm{piv}(P[A_i]) < \mathrm{piv}(P[A_t]) \ \forall \ t < i\}$ in increasing order
  for $i \in A$ *in decreasing order*
    $k \leftarrow$ the smallest index in B such that $\mathrm{piv}(P[B_k]) \leqslant \mathrm{piv}(P[i])$
    if $i = B_k$
      $k \leftarrow k - 1$
    add column B_k to column i in P and in V
  remove $P[j]$, $V[j]$, and row j in P and in V
return P, V

Implementation: On top of PHAT software, publicly available

- Bottleneck is accessing rows to get "affected" simplices
- Tested and timed, see arxiv - time vs. space tradeoff (need matrix U)

Usage

Input boundary matrix and output persistence pairs are in PHAT format:

input.bmat

```
0
0
0
0
0
1 0 3
1 0 4
1 1 4
1 1 2
1 1 3
1 0 1
2 6 7 10
```

updates.txt

```
remove 11 7
insert 8 2 3
```

```
./phat-sirup --ascii --standard \
  --bmat_in input.bmat
  --bmat_out input_reduced.bmat
  --opmat_out input.opmat
  --pairs_out input.pairs

./phat-sirup --ascii --standard --dont_reduce \
  --bmat_in input_reduced.bmat
  --opmat_in input.opmat
  --updates updates.txt
  --pairs_out output.pairs
```

Goal: Impelement removal, insertion, swapping (see also Dionysus).

Availability: Main (remove) and dev (remove, insert) branches on BitBucket.

Work in progress!!

Thank you for your attention

- Bauer, Kerber, Reininghaus, Wagner. *PHAT - Persistent Homology Algorithms Toolbox.* Journal of Symbolic Computation, 2016.
- Busaryev, Dey, Wang. *Tracking a Generator by Persistence.* Computing and Combinatorics, 2010.
- Cohen–Steiner, Edelsbrunner, Morozov. *Vines and vineyards by updating persistence in linear time.* Symposium of Computational Geometry, 2006.
- Edelsbrunner, Letscher, Zomorodian. *Topological persistence and simplification.* Symposium on Foundations of Computer Science, 2000.
- Giunti, Lazovskis. *Pruning vineyards: updating barcodes by removing simplices.* arxiv, 2025.
- Giunti, Lazovskis. *PHAT-SiRUP.* BitBucket, 2025.
- Hickok. *Persistence Diagram Bundles: A Multidimensional Generalization of Vineyards.* arxiv, 2022.
- Morozov, Skraba. *Persistent (Co)Homology in Matrix Multiplication Time.* arxiv, 2024.
- Morozov. *Dionysus, a C++ library for computing persistent homology.* 2007.
- Piekenbrock, Perea. *Move schedules: fast persistence computations in coarse dynamic settings.* Journal of Applied and Computational Topology, 2024.
- Turner. *Representing Vineyard Modules.* arxiv, 2023.