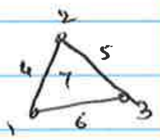
Mar 2

Spectral Sequences Algorithm

→ Consider filtration $\phi = K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n$ with total compatible ordering of simplices $\sigma_1, \sigma_2, \ldots \sigma_n$.
Let M be the $n \times n$ matrix → boundary matrix.
i^{th} column of M represents σ_i, the i^{th} simplex in ordering.

→ Consider sequence of indices $[0 < t_1 < t_2 < \ldots < t_m = n]$ partition the sequence of n simplices in the ordering into m non-overlapping parts s.t. j^{th} part consists of simplices indexed $t_{j-1}+1$ to t_j



$\sigma \to \underbrace{1 \;\; 2 \;\; 3}_{t_1 = 3} \; \underbrace{4 \;\; 5}_{t_2 = 5} \; \underbrace{6 \;\; 7}_{t_3 = 7}$ $\qquad m = 3$

→ This converts the $n \times n$ boundary matrix M into an $m \times m$ block matrix D

$$M = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left[\begin{array}{ccccccc} & & & 1 & & 1 & \\ & & & 1 & 1 & & \\ & & & & 1 & 1 & \\ & & & & & & 1 \\ & & & & & & 1 \\ & & & & & & 1 \\ & & & & & & \end{array}\right] \end{array}$$

$$D = \begin{array}{c|ccc|cc|cc|} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & & & & 1 & & 1 & \\ 2 & & & & 1 & 1 & & \\ 3 & & & & & 1 & 1 & \\ \hline 4 & & & & & & & 1 \\ 5 & & & & & & & 1 \\ \hline 6 & & & & & & & 1 \\ 7 & & & & & & & \\ \hline \end{array}$$

→ This is equivalent to having a filtration with multiple simplices added at each step.

block D_j^i ⇒ Simplices added in i^{th} step that have faces in j^{th} step.

Notation : Superscript : column chunk / block
Subscript : row chunk / block.

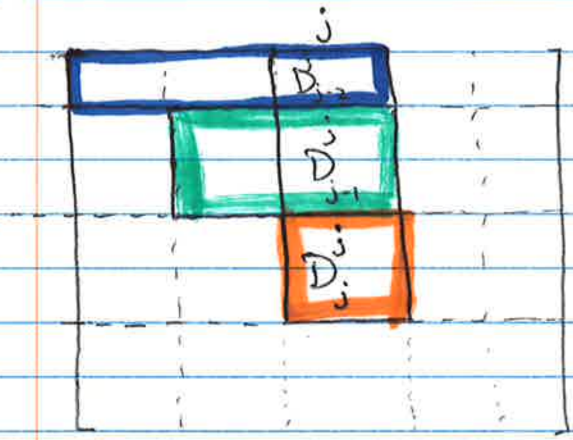
Algorithm: Given $m \times m$ block matrix D

```
• For r = 1 to m
•   For j = r to m
•     For columns l in D^j
        (from left to right)
•       while ( low(l) ∈ D_{j-r+1} and
                  ∃ l' in D^{j-r+1} s.t.
                        low(l') == low(l) )
•          add l' to l (columns)
•       end while
•     end For
•   end for
• end for
```

r indicates "phases" of the algorithm.

$\boxed{\text{In phase 1}}$ we consider for each column chunk, only columns that have pivot in the diagonal block.

Only reduce column l if there is a column l' in the same column chunk with its pivot in diagonal block.



processing j^{th} column chunk.

in phase 1, $r=1$ only consider columns that have pivot in D^j_j

$l' \rightarrow$ columns in D^j with pivot in D_j (on left of col l)

in phase 2, $r=2$. Consider columns l in D^j that have pivot in D^j_{j-1}

$l' \rightarrow$ columns in D^{j-1}, D^j that have pivot in D^{j-1}_{j-1} ~~and~~ and D^j_{j-1}

l' belongs to
- in phase 1
- in phase 2
- in phase 3

→ In the standard reduction algorithm we sweep columns from left to right

→ spectral sequences algorithm → sweeping diagonally.

→ persistence pairs are found in increasing order of persistence

Persistent homology Computation : Key ideas

1. We want to find pairs of simplices (i,j) [persistence pairs]
where $i \rightarrow$ positive simplex, generator of PH class
$j \rightarrow$ negative simplex, destroyer of PH class.

$\therefore$ Given total compatible ordering of simplices in filtration
(a) Construct boundary matrix, (b) apply reduction (c) infer pairs

$\rightarrow$ Pairs of simplices correspond to pairs of columns in boundary M.

2. A simplex σ_i (i^{th} column in BM) can either be +ve or -ve
$\rightarrow$ +ve columns reduce to 0 after reduction
$\rightarrow$ -ve columns have non-zero pivots

\# Strategies to reduce number of computations

1. Clearing! Each column M^j either reduces to 0 or
gives a persistence pair (i,j) wher $i = \text{pivot}(j)$
then i is +ve column and will eventually reduce to 0
$\Rightarrow$ if found pair (i,j) set column i to 0.
$\Rightarrow$ Only works if reduction is applied from right to left.

2. Compression: if found persistence pair $(i,j) \Rightarrow j$ is -ve col.
$\therefore$ j can not be a pivot (+ve column) for any other col.
$\Rightarrow$ set row j to 0 because there is no pivot in
the j^{th} row.

i.e. if found column M^j with $i = \text{pivot}(j)$ [pair (i,j)]

then Clearing $\Rightarrow$ set col i to 0 [requires right to left]

Compression $\Rightarrow$ set row j to 0 [requires left to right]

Reduction in chunks

→ Construct the $m \times m$ block matrix D from boundary matrix M.

Idea: (a) Perform few steps of spectral sequence algorithm (2-phases)
→ right to left reduction within blocks / chunks so that clearing optimization can be applied.

→ Only finds local pairs (i, j) where i and j are from the same or adjacent chunks.
→ if col j has pivot i s.t. i belongs to the same chunk as j or to chunk adjacent to j then j is called local column.
all other columns are global columns.

(b) Next, independently compress all columns globally.
(Given local pairs (i, j) from (a) set rows j to 0)

(c) After (a) & (b), we are left with a submatrix of global rows and columns → reduce using clearing.

Algorithms:

Compress (R, k)

```
for non-zero entry index l of R_k
    if l is paired
        if l is inactive
            R^k_l <- 0
        else
            j <- L[l]
            R^k <- R^k + R^j
        end if
    end if
end for.
```

Mark_Active_Entries (R)

```
For each unpaired col R^k
    Mark_Column (R, k)
```

Mark_Column (R, k)

```
For each non-zero entry index i of R^k
    if i is unpaired
        mark k as active
    if i is +ve
        j <- L[i]
        if j ≠ k and Mark_Column(R, j)
            mark k as active
mark k as inactive
```

```
Local_Reduction (M, t_1, t_2, ..., t_m)

R ← M ; L ← [0, 0, ..., 0]
for δ = d, d-1, ..., 0
  for r = 1, 2
    for b = r, ..., m
      for j = t_{b-1}+1, ..., t_b
          with dim σ_j = δ
        if j is not paired
          while (R^j ≠ 0 and L[pivot(R^j_δ)] ≠ 0
                  and pivot(R^j) > t_{b-r})
            R^j ← R^j + R^{L[pivot(R^j)]}
          if R^j ≠ 0
            i ← pivot(R^j)
            if i > t_{b-r}
              L[i] ← j
              R^i ← 0
              mark (i, j) paired
```

[Right to left in decreasing order of simplex dimension]

this loop can be paralleliged [local reduction of column chunks in spectral sequence phases is independent]

$\longrightarrow$ reduction step

$\longrightarrow$ clearing step.

⇒ M is the boundary matrix.
t_i define block boundaries in block matrix D [implicit]

⇒ Right to left reduction within each ~~bto~~ column chunk is applied in decreasing order of simplex dimensions

Why ? ⇒ ⓐ Reduce simplex before it's faces
ⓑ Reduce all d-dimensional simplices before (d-1)-dimensional simplices within the block.

Persistence in Chunks : Given boundary matrix M and block boundary indices $t_0, t_1, \dots t_m$

- $(R, L) \leftarrow$ Local_Reduction $(M, t_0, t_1, \cdots t_m)$ } (can be paralleliged)
- Mark_Active_Entries (R)
- for $\delta = d, d-1, \dots, 0$

    for $j = 1, \dots, n$ and $\dim(\sigma_j) = \delta$
        if col j is not paired
            Compress (R, j)
    } Global column Compression (can be paralleliged)

    for $j = 1, \dots, n$ and $\dim(\sigma_j) = \delta$
        while $R^j \neq 0$ and $L[\text{pivot}(R^j)] \neq 0$
            $R^j \leftarrow R^j + R^{L[\text{pivot}(R^j)]}$ $\longrightarrow$ Reducing Column
        if $R^j \neq 0$
            $i \leftarrow \text{pivot}(R^j)$
            $L[i] = j$
            $R^i = 0$ $\longrightarrow$ Clearing
            pair (i, j) marking
    } Reducing submatrix of global columns and rows. [Should have size much smaller than M]

Run time: Let g be the number of global columns after step 1 ($g \ll n$ ideally.)

→ with $\overbrace{\left(\frac{n}{\log n}\right)}^{=m}$ chunks, each of size $\log n$

run time : $O\left(n \log^2 n + gn \log n + g^3\right)$

→ with $\overbrace{(\sqrt{n})}^{=m}$ chunks, each of size $\sqrt{n}$

runtime : $O\left(n^2 + g n \sqrt{n} + g^3\right)$

⇒ much better than the $O(n^3)$ runtime of standard algorithm.