## Persistent homology Computation : Software packages

① TDA - R package ( https://cran.r-project.org/package=TDA )

uses two underlying C++ libraries :

ⓐ Dionysus : ( www.mrzv.org/software/dionysus )

→ Not well documented. Have to work through some examples provided.

→ Has python bindings available.

ⓑ GUDHI : ( gudhi.gforge.inria.fr )

② PHAT : https://github.com/blazs/phat
https://bitbucket.org/phat-code/phat

C++ source. can be embedded in project OR can be used as command line utility.

→ Requires user to specify filtration / boundary matrix

③ DiPHA : distributed version of PHAT
https://github.com/DIPHA/dipha

→ works with distance matrices
→ distributed computation using MPI (required to build DiPHA)
→ can be used as command line utility.

## Reduction Algorithm

Left to right persistence

$R \leftarrow \partial$ ; $L \leftarrow [0, 0, \cdots, 0]$

for $j = 1, \cdots, n$

  while $R^j \neq 0$ and $L[\text{low}(R^j)] \neq 0$

    $R^j \leftarrow R^j + R^{L[\text{low}(R^j)]}$

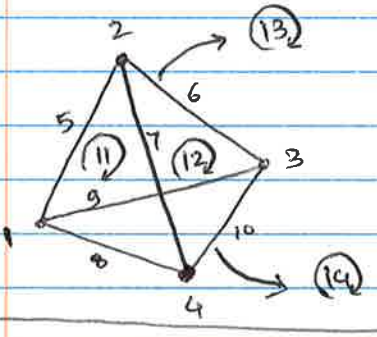
  if $R^j \neq 0$

    $L[\text{low}(R^j)] = j$

For columns in order from left to right, if a column on left has the same low(·) / pivot then add it to the column on right.

$R^j \Rightarrow j^{th}$ column of $R$

$\partial \Rightarrow$ boundary matrix




| | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | X | X | | | | | 14 + 12 + 13 + 11 |
| 2 | (1) | 1 | 1 | X | X | X | | | | |
| 3 | | (1) | | | X | X | | | | |
| 4 | | | (1) | X | | X | | | | |
| 5 | | | | 8 + 7 + 5 | 9 + 6 + 5 | 10 + 7 + 6 | 1 | | 1 | X |
| 6 | | | | | | | | 1 | 1 | X |
| 7 | | | | | | | 1 | 1 | | X |
| 8 | | | | | | | (1) | | | X |
| 9 | | | | | | | | | (1) | X |
| 10 | | | | | | | | (1) | | X |

dim 0 → (2,5) (3,6) (4,7) (1,∞)

dim 1 → (8,11) (9,13) (10,12)

dim 2 → (14,∞)

2, 3, 4 → +ve 0-simplices

5, 6, 7 → −ve 1-simplices

8, 9, 10 → +ve 1-simplices

14 → +ve 2-simplex

11, 12, 13 → −ve 2-simplices

**Observation:**

(1) Each p-dimensional simplex $\sigma_i$ either increases $\beta_p$ by 1 or decreases $\beta_{p-1}$ by 1

eg. an edge either reduces # Connected components OR creates a loop (non-bounding cycle ∵ the faces haven't showed up in filtration yet)

(*) Boundary of simplex must appear before the simplex in filtration.

→ +ve simplex : one that adds to $\beta_p$

→ −ve simplex : one that reduces $\beta_{p-1}$

Observations:

→ All columns corresponding to +ve simplices (+ve columns) are eventually reduced to 0

→ +ve simplex : creates a non-bounding cycle ⇒ boundary is 0
∴ sum of columns in boundary matrix must be 0.

i.e. The +ve column can be represented as linear combination of columns to its left ( ~~[illegible]~~ )

→ Columns that have a non-zero pivot form the basis
∵ they can't be represented as linear combination of columns to their left in boundary matrix
⇒ These are called as essential columns.

→ Most of the computations done during reduction are on columns that reduce to 0 : on +ve, non-essential columns
if we already know that a column is +ve, we can set it to 0 without performing any computations.

# Decreasing dimension persistence

$R \leftarrow \partial$ ; $L \leftarrow [0, 0, \cdots, 0]$

for $\delta = d, d-1, \cdots, 0$

  for $j = 1, \cdots, n$ with $\dim(R^j) = \delta$

    [ Reduce $R^j$ ]

  if $R^j \neq 0$

    $L[\text{low}(R^j)] \leftarrow j$

    $R[\text{low}(R^j)] \leftarrow 0$

if $i = \text{low}(R^j)$ then $(i, j)$ form a persistence pair. $R^i$ is +ve column, $R^j$ is -ve column.

The idea is to kill / clear all non-essential +ve columns without performing reduction.

→ Requires reducing higher dim. columns before lower dim. ones

⇒ This is called the "twist" algorithm
→ default algorithm in PHAT

Right to left execution ($\delta = d, d-1, \cdots 0$)