Software for persistent homology: overview and demonstration

Nina Otter

ATI scoping workshop
and
CAT research school
Mathematical Institute, University of Oxford

11 September 2015

\footnote{A roadmap for the computation of persistent homology, N.O, M. Porter, U. Tillmann, P. Grindrod, H. Harrington, 2015}
What is out there?
- **Perseus** [http://www.sas.upenn.edu/~vnanda/perseus/](http://www.sas.upenn.edu/~vnanda/perseus/)
- **JavaPlex** [http://appliedtopology.github.io/javaplex/](http://appliedtopology.github.io/javaplex/)
- **Dionysus** [http://www.mrzv.org/software/dionysus/](http://www.mrzv.org/software/dionysus/)
- **pHom** [http://cran.r-project.org/web/packages/phom/](http://cran.r-project.org/web/packages/phom/)
- **TDA** [http://cran.r-project.org/web/packages/TDA/](http://cran.r-project.org/web/packages/TDA/)
- **GAP persistence** [http://www-circa.mcs.st-and.ac.uk/~mik/persistence/](http://www-circa.mcs.st-and.ac.uk/~mik/persistence/)
- **PHAT** [https://code.google.com/p/phat/](https://code.google.com/p/phat/)
- **DIPHA** [https://code.google.com/p/dipha/](https://code.google.com/p/dipha/)
- **GUDHI** [https://project.inria.fr/gudhi/software/](https://project.inria.fr/gudhi/software/)
- **The persistence landscape toolbox**
  [https://www.math.upenn.edu/~dlotko/persistenceLandscape.html](https://www.math.upenn.edu/~dlotko/persistenceLandscape.html)
- **SimpPers**
- **jHoles** [http://cuda.unicam.it/jHoles](http://cuda.unicam.it/jHoles)
- ...
A brief (biased) history of PH softwares


2005 Plex  V. de Silva, P. Perry, L. Kettner, A. Zomorodian

2011 JavaPlex  A. Tausz, M. Vejdemo-Johansson, H. Adams

2012 Perseus  V. Nanda

2013 PHAT  M. Kerber, J. Reininghaus, U. Bauer, H. Wagner

2014 DIPHA  M. Kerber, J. Reininghaus, U. Bauer

2014 GUDHI  C. Maria, J.-D. Boissonnat, M. Glisse, M. Yvinec
Pipeline for PH computation from point clouds

Data \rightarrow (1) \rightarrow \text{Filtered complex} \rightarrow (2) \rightarrow \text{Barcodes}
## Optimisations

1. From data to a filtered complex

<table>
<thead>
<tr>
<th>Complex $K$</th>
<th>Size of $K$</th>
<th>(A) Approximation</th>
<th>(B) Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Čech</td>
<td>$O(2^n)$</td>
<td>linear size appr. [KS13]</td>
<td>tidy set [Z10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Morse red. [MN13]</td>
</tr>
<tr>
<td>Vietoris–Rips (VR)</td>
<td>$O(2^n)$</td>
<td>linear-size appr. [SH12]</td>
<td>tidy set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(lazy) witness (LW)[dSC14]</td>
<td>Morse red.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GIC [DFW13]</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$n^{O([d/2])}$</td>
<td></td>
<td>tidy set</td>
</tr>
<tr>
<td>($n$ points in $\mathbb{R}^d$)</td>
<td></td>
<td></td>
<td>Morse red.</td>
</tr>
</tbody>
</table>

### Data structures:

- **Simplex tree [Boissonnat, Maria 2012]**

---

[Z10] Zomorodian

[MN] Mischaikow, Nanda 2013
[dSC14] de Silva, Carlsson 2013
Optimisations: implementations

(1) From data to a filtered complex

<table>
<thead>
<tr>
<th>Complex $K$</th>
<th>(A) Approximation</th>
<th>(B) Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Čech</td>
<td>linear size appr.</td>
<td>tidy set</td>
</tr>
<tr>
<td>$\rightarrow$ DIONYSUS</td>
<td>$\rightarrow$ ?</td>
<td>not open source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Morse red.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\rightarrow$ PERSEUS</td>
</tr>
<tr>
<td>Vietoris–Rips (VR)</td>
<td>linear-size appr.</td>
<td>tidy set</td>
</tr>
<tr>
<td>$\rightarrow$ DIONYSUS, PERSEUS, PHAT</td>
<td>$\rightarrow$ ?</td>
<td>Morse red.</td>
</tr>
<tr>
<td>DIPHA, javaPlex, GUDHI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>(lazy) witness (LW)</td>
<td>tidy set</td>
</tr>
<tr>
<td>$\rightarrow$ DIONYSUS</td>
<td></td>
<td>Morse red.</td>
</tr>
<tr>
<td></td>
<td>javaPlex GIC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\rightarrow$ ?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>tidy set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Morse red.</td>
</tr>
</tbody>
</table>

Data structure:

- **Simplex tree $\rightarrow$ GUDHI**
Optimisations

(2) From a boundary matrix to barcodes

Sequential:
- Dual algorithm [de Silva, Morozov, Vejdemo-Johansson 2011]
- Twist algorithm [Chen, Kerber 2011]

Parallel:
- Spectral sequence algorithm [Edelsbrunner, Harer 2008]
- Chunk algorithm [Bauer, Kerber, Reininghaus 2013]
- Distributed algorithm [Bauer, Kerber, Reininghaus 2014]

Data structure:
- Bit tree pivot column [Bauer, Kerber, Reininghaus, Wagner 2013]
- Compressed annotation matrix [Boissonnat, Dey, Maria 2013]
**Spectral sequence algorithm**

Let $k_j$ denote the number of simplices in subcomplex $K_j$.

Let $B^i_j$ denote the columns numbered $k_{j-1} + 1$ to $k_j$.

Let $B_i$ denote the rows numbered $k_{i-1} + 1$ to $k_i$.

Idea:

- Reduce the matrix in phases: in each phase $r$, reduce columns in $B^i_j$ by adding columns in the blocks from $B^{j-r+1}_j$ to $B^i_j$.

Optimisation:

- The reduction in each block and each phase is independent, and can be executed in parallel.
### Phase $r = 1$:

<table>
<thead>
<tr>
<th></th>
<th>$B^1$</th>
<th>$\cdots$</th>
<th>$\cdots$</th>
<th>$B^j$</th>
<th>$B^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_j$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Phase $r = 2$:

<table>
<thead>
<tr>
<th></th>
<th>$B^1$</th>
<th>$\cdots$</th>
<th>$\cdots$</th>
<th>$B^j$</th>
<th>$B^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\cdots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_j$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Phase $r = 3$:**

<table>
<thead>
<tr>
<th></th>
<th>$B^1$</th>
<th>$\ldots$</th>
<th>$\ldots$</th>
<th>$B^j$</th>
<th>$B^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_j$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Phase $r = k$:**

<table>
<thead>
<tr>
<th></th>
<th>$B^1$</th>
<th>$\ldots$</th>
<th>$\ldots$</th>
<th>$B^j$</th>
<th>$B^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_j$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B_k$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Optimisations: implementations

(2) From a boundary matrix to barcodes

Sequential:
- Dual algorithm $\rightarrow$ Dionysus, javaPlex, GUDHI, PHAT, DIPHA
- Twist algorithm $\rightarrow$ PHAT, DIPHA

Parallel:
- Spectral sequence algorithm $\rightarrow$ PHAT
- Chunk algorithm $\rightarrow$ PHAT
- Distributed algorithm $\rightarrow$ DIPHA

Data structure:
- Bit tree pivot column $\rightarrow$ PHAT, DIPHA
- Compressed annotation matrix $\rightarrow$ GUDHI
<table>
<thead>
<tr>
<th>Software</th>
<th>Precomp.</th>
<th>filt. Compl. ²</th>
<th>Parallel</th>
<th>Visualiz.</th>
<th>PH algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>javaPlex</td>
<td>✓</td>
<td>VR, LW, W, CW</td>
<td>✗</td>
<td>✓</td>
<td>standard, dual zig zag</td>
</tr>
<tr>
<td>Perseus</td>
<td>✓</td>
<td>VR</td>
<td>✗</td>
<td>✓</td>
<td>Morse reductions + standard</td>
</tr>
<tr>
<td>Dionysus</td>
<td>✗</td>
<td>α, VR, Čech</td>
<td>✗</td>
<td>✗</td>
<td>standard, dual zig-zag</td>
</tr>
<tr>
<td>PHAT</td>
<td>✗</td>
<td>✗</td>
<td>✓ shared</td>
<td>✗</td>
<td>standard, dual twist chunk, spectral seq.</td>
</tr>
<tr>
<td>DIPHA</td>
<td>✗</td>
<td>VR, lower star</td>
<td>✓ distr.</td>
<td>✓</td>
<td>dual, twist distributed</td>
</tr>
<tr>
<td>GUDHI</td>
<td>✗</td>
<td>VR</td>
<td>✗</td>
<td>✗</td>
<td>multifield dual</td>
</tr>
</tbody>
</table>

²VR=Vietoris Rips complex, W=witness complex, LW=lazy witness complex, CW=CW complex, WRCF=weight rank clique filtration
We study the software from four different points of view:

1. Performance measured in CPU and real time
2. Memory usage
3. Maximum size of simplicial complex allowed by the software
4. User-friendliness: phases of computation of PH supported by software
Data

**Synthetic data**
- Points sampled from Klein bottle
- Random Vietoris Rips complexes

**Real world data**
- Genomic sequence of HIV virus
- Points sampled from 3D scans of Stanford dragon
- C. Elegans neuronal network
- Human genome network
Machines

- Cluster: 1728 (180*16) cores of 2.0GHz
  RAM: 64 GiB x80 nodes, 128 GiB x 4 nodes

---

Advanced Research Computing (ARC), Oxford
Table: Performance measured in wall-time and CPU seconds for the computation of PH with the Vietoris Rips complex.
### Table: Memory usage in GB

For **JavaPlex**, we indicate the value of the maximum heap size that was sufficient to perform the computation. For **DIPHA**, we indicate the maximum memory used by a single core.

<table>
<thead>
<tr>
<th>Data set</th>
<th>C. elegans</th>
<th>Klein</th>
<th>HIV</th>
<th>Dragon 1</th>
<th>Dragon 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of complex</td>
<td>$4.4 \times 10^6$</td>
<td>$1.1 \times 10^7$</td>
<td>$2.1 \times 10^8$</td>
<td>$1.7 \times 10^8$</td>
<td>$1.3 \times 10^9$</td>
</tr>
<tr>
<td><strong>JavaPlex st</strong></td>
<td>$&lt; 5$</td>
<td>$&lt; 15$</td>
<td>$&gt; 120$</td>
<td>$&gt; 120$</td>
<td>$&gt; 120$</td>
</tr>
<tr>
<td><strong>Dionysus st</strong></td>
<td>1.3</td>
<td>11.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>DIPHA st</strong></td>
<td>0.1</td>
<td>0.2</td>
<td>2.7</td>
<td>2.4</td>
<td>4.9</td>
</tr>
<tr>
<td><strong>Perseus</strong></td>
<td>5.1</td>
<td>12.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Dionysus d</strong></td>
<td>0.5</td>
<td>1.1</td>
<td>-</td>
<td>16.8</td>
<td>-</td>
</tr>
<tr>
<td><strong>DIPHA d</strong></td>
<td>0.1</td>
<td>0.2</td>
<td>1.8</td>
<td>1.8</td>
<td>13.8</td>
</tr>
<tr>
<td><strong>GUDHI</strong></td>
<td>0.2</td>
<td>0.6</td>
<td>9.9</td>
<td>9.2</td>
<td>64.5</td>
</tr>
</tbody>
</table>
### Results: maximal size of simplicial complex

<table>
<thead>
<tr>
<th>Software</th>
<th><strong>JAVA PLEX</strong></th>
<th><strong>DIONYSUS</strong></th>
<th><strong>DIPHA</strong></th>
<th><strong>PERSEUS</strong></th>
<th><strong>GUDHI</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>max. size</td>
<td>$1 \cdot 10^7$</td>
<td>$1.1 \cdot 10^7$</td>
<td>$3.2 \cdot 10^8$</td>
<td>$1.3 \cdot 10^9$</td>
<td>$1.3 \cdot 10^9$</td>
</tr>
</tbody>
</table>

**Table:** Maximal size of simplicial complex supported by the software.
Table: Roughly, one can divide the pipeline for the computation of PH into five steps: (1) installation of the software, (2) computation of the complex from input data, (3) computation of the boundary matrix, (4) computation of barcodes, and (5) visualisation of outputs. For each software package, we indicate which of these steps are supported.
Interlude: Demonstration
Demonstration

Using points sampled from the Klein bottle we will:

- Compute PH using different filtered simplicial complexes
- Plot results as barcodes and persistence diagrams
- Compare results computing bottleneck distance
The Klein bottle

Homology of image of immersion in $\mathbb{R}^3$ and embedding in $\mathbb{R}^4$:

\[
\begin{align*}
H_0(K) &= F_2 \\
H_1(K) &= F_2 \oplus F_2 \\
H_2(K) &= F_2
\end{align*}
\]
Given a point cloud $X \subset \mathbb{R}^d$

- the Čech complex at scale $\epsilon$ is
  \[ C(\epsilon) = \{ \sigma \subset X \mid \bigcap_{x \in \sigma} B_x(\epsilon) \neq \emptyset \} \]

- the Vietoris Rips complex at scale $\epsilon$ is
  \[ V(\epsilon) = \{ \sigma \subset X \mid d(x, y) \leq 2\epsilon \text{ for all } x, y \in \sigma \} \]

- the $\alpha$-complex at scale $\epsilon$ is
  \[ \alpha(\epsilon) = \{ \sigma \subset X \mid \bigcap_{x \in \sigma} B_x(\epsilon) \cap V_x \neq \emptyset \} \]

For all $\epsilon > 0$ we have: $\alpha(\epsilon) \subset C(\epsilon) \subset V(\epsilon)$. 
Lazy witness complex

Given a point cloud $X$, choose a set of landmarks $L \subset X$. Let $\nu \in \mathbb{N}$.

For $x \in X$ let $m(x)$ be the distance from $x$ to its $\nu$ closest landmark point.

The lazy witness complex $LW_\nu(\epsilon)$ at scale $\epsilon$:

- vertices are $L$
- edge $\{x_0, x_1\}$ is in $LW_\nu(\epsilon)$ if there exists a witness $z \in Z$ such that $\max\{d(x_0, z), d(x_1, z)\} \leq t + m(z)$. 
With Dionysus:

```
./cech-complex < input-file > output-file
```

input-file:

```
embedding-dimension max-dimension
x_{00} x_{01} \ldots
x_{10} x_{11} \ldots
\ldots
```

\[ |X| = 100 \subset \mathbb{R}^3 \]
max-dimension 3 ✓

\[ |X| = 400 \subset \mathbb{R}^3 \]
max-dimension 2 ✓
max-dimension 3 ×
With **Dionysus**:

```
./rips-pairwise -s max-dimension -m max-distance \
-d output-file input-file
```

**input-file:**

\[ X \]
\[ |X| = 100 \subset \mathbb{R}^3 \]
max-dimension 3 ✓

\[ |X| = 400 \subset \mathbb{R}^3 \]
max-dimension 2 ✓
max-dimension 3 ×
With **PERSEUS**: 

```
./perseus distmat input-file output-file
```

**input-file:**

```
number-points
\( \epsilon_0 \) \( \epsilon \)-increment number-steps max-dimension
\( \epsilon_0 \) \( d_{00} \) \( d_{01} \) ...
\( d_{10} \) \( d_{11} \) ...
...
```

\(|X| = 100 \subset \mathbb{R}^3\) 

max-dimension 3 ✓

\(|X| = 400 \subset \mathbb{R}^3\) 

max-dimension 2 ✓
max-dimension 3 ✗
With GUDHI:

```bash
./rips_persistence input-file -r max-distance -d max-dimension -p prime -o output-file input-file
```

input-file:

```
x00 x01 ...
x10 x11 ...
```

\[ |X| = 100 \subset \mathbb{R}^3 \] \(\checkmark\)

max-dimension 3 \(\checkmark\)

\[ |X| = 400 \subset \mathbb{R}^3 \] \(\checkmark\)

max-dimension 2 \(\checkmark\)

max-dimension 3 \(\checkmark\)
With DIPHA:

./dipha [options] input-file output-file

input-file in binary format:

|X| = 100 ⊂ \mathbb{R}^3
max-dimension 3 ✓

|X| = 400 ⊂ \mathbb{R}^3
max-dimension 2 ✓
max-dimension 3 ✓
With **JAVAPLEX** in **MATLAB**:

\[
vietoris_rips_javaplex(input\text{-}file, \text{max-dimension}, \ldots, \text{max-filtration\text{-}value}, \text{number\text{-}steps}, \text{plot\text{-}name})
\]

**input-file:**

\[
\begin{align*}
X_00 & \quad X_01 & \ldots \\
X_{10} & \quad X_{11} & \ldots \\
\ldots
\end{align*}
\]

\[
|X| = 100 \subset \mathbb{R}^3 \quad \text{max-dimension } 3 \checkmark
\]

\[
|X| = 400 \subset \mathbb{R}^3
\]

\[
\text{max-dimension } 2 \checkmark
\]

\[
\text{max-dimension } 3 \times
\]
Computations: $\alpha$

With **Dionysus** from command line:

```
./alphashapes3d-cohomology input-file output-file
```

input-file:

```
x_00 x_01 ...
x_10 x_11 ...
...  
```

$|X| = 100 \subset \mathbb{R}^3$

max-dimension 3 ✓  

$|X| = 400 \subset \mathbb{R}^3$

max-dimension 3 ✓
Computations: lazy witness

with **JAvaPlex** in **MATLAB**:

```
lazy_witness_javaPlex(input-file, max_dimension,...
num_landmark_points, type, nu, filtration_steps, plot_name)
```

input-file:

```
x_00 x_01 ...
x_10 x_11 ...
```

\[ |X| = 100 \subset \mathbb{R}^3 \; L = 50 \]
max-dimension 3 ✔

\[ |X| = 400 \subset \mathbb{R}^3 \; L = 200 \]
max-dimension 2 ✔
max-dimension 3 ✗

\[ |X| = 100 \subset \mathbb{R}^3 \; L = 50 \]
max-dimension 3 ✔
Barcodes for $|X| = 100$

lazy Klein bottle (dimension 0)

lazy Klein bottle (dimension 1)

lazy Klein bottle (dimension 2)
Barcodes for $|X| = 100$
Other barcodes and bottleneck distance: exercise!

Scripts will be uploaded to http://people.maths.ox.ac.uk/otter/ this weekend. In the meantime you can install the libraries..
Other barcodes and bottleneck distance: exercise!

- Scripts will be uploaded to http://people.maths.ox.ac.uk/otter/ this weekend.
Other barcodes and bottleneck distance: exercise!

- Scripts will be uploaded to http://people.maths.ox.ac.uk/otter/ this weekend.
- In the meantime you can install the libraries.
Conclusions

Challenges:

- User-friendly interfaces.
- Creation of a computational topology library.
- Definition and construction of benchmarking datasets.
- Uniformization of input type across different implementations.
- Focus on step (1) of pipeline: implementation and construction of simpler simplicial complexes.

Stream Processing: new techniques are needed to compute PH for streams of data.
Conclusions

Challenges:

- User-friendly interfaces.
- Creation of a computational topology library.
- Definition and construction of benchmarking datasets.
- Uniformization of input type across different implementations.
- Focus on step (1) of pipeline: implementation and construction of simpler simplicial complexes.
- Stream Processing: new techniques are needed to compute PH for streams of data.
Software

- **Perseus** [http://www.sas.upenn.edu/~vnanda/perseus/](http://www.sas.upenn.edu/~vnanda/perseus/)
- **JavaPlex** [http://appliedtopology.github.io/javaplex/](http://appliedtopology.github.io/javaplex/)
- **Dionysus** [http://www.mrzv.org/software/dionysus/](http://www.mrzv.org/software/dionysus/)
- **pHom** [http://cran.r-project.org/web/packages/phom/](http://cran.r-project.org/web/packages/phom/)
- **TDA** [http://cran.r-project.org/web/packages/TDA/](http://cran.r-project.org/web/packages/TDA/)
- **GAP persistence** [http://www-circa.mcs.st-and.ac.uk/~mik/persistence/](http://www-circa.mcs.st-and.ac.uk/~mik/persistence/)
- **PHAT** [https://code.google.com/p/phat/](https://code.google.com/p/phat/)
- **DIPHA** [https://code.google.com/p/dipha/](https://code.google.com/p/dipha/)
- **GUDHI** [https://project.inria.fr/gudhi/software/](https://project.inria.fr/gudhi/software/)
- **The persistence landscape toolbox**
  [https://www.math.upenn.edu/~dlotko/persistenceLandscape.html](https://www.math.upenn.edu/~dlotko/persistenceLandscape.html)
- **SimpPers**
- **jHoles** [http://cuda.unicam.it/jHoles](http://cuda.unicam.it/jHoles)
- **...**