Cubical and statistical topology.

Paweł Dłotko

21/04/2015 Porquerolles.

Two motivations for cubical complexes.

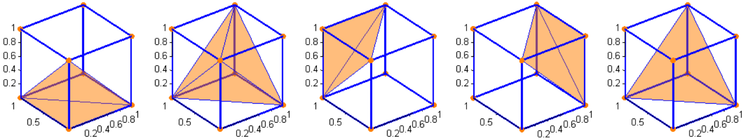
1. Rigorous numerics.
2. Image analysis.

Rigorous numerics.

1. Rigorous interval arithmetic – numbers represented as intervals.
2. Elementary operations implemented in a way, that the true result is always contained in the resulting interval.
3. Elementary functions approximated with Taylor series.
4. Result of a computations of $u(x)$ – an interval that is guaranteed to contain the value of $u(x)$.
5. x do not have to be a point, it can be a cube.

Image analysis.

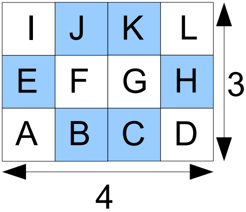
1. In image analysis data often are given as pixels, voxels, 4d voxels etc.
2. Sometime may want to compute some topological information of the image.



Cubical complex.

1. Elementary interval – $[n, n+1]$ (non-degenerated) or $[n, n]$ (degenerated) for $n \in \mathbb{Z}$.
2. Boundary of elementary interval $\partial[n, n+1] = [n+1, n+1] - [n, n]$. $\partial[n, n] = 0$.
3. Elementary cube – product of elementary intervals. $C = I_1 \times \ldots \times I_n$.
4. Boundary of elementary cube, $\partial C = \partial(I_1 \times \ldots \times I_n) = \sum_{i=1}^{n} I_1 \times \ldots \partial I_i \times \ldots \times I_n$.
5. Cubical complex $\mathcal{K}$ – collection of cubes closed under operation of taking subsets.

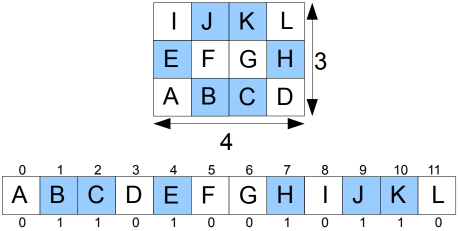
Bitmap, maximal cubes.



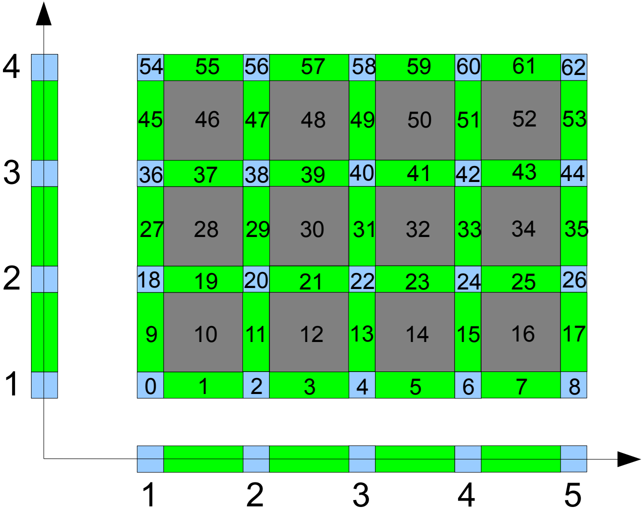
0	1	2	3	4	5	6	7	8	9	10	11
A	B	C	D	E	F	G	H	I	J	K	L
0	1	1	0	1	0	0	1	0	1	1	0

How to compute neighbors?

1. Let us compute neighbors of a vertex F.
2. Its number in bitmap is 5.
3. Bitmap is two dimensional, its wight is 4 and height is 3.
4. Two neighbors are located at $5 - 1 = 4$ and $5 + 1 = 6$ positions. They are E and G.
5. Two others are located in $5 - 4 * \left\lfloor \frac{5}{4} \right\rfloor = 1$ and $5 + 4 * \left\lfloor \frac{5}{4} \right\rfloor = 9$. They are B and J.

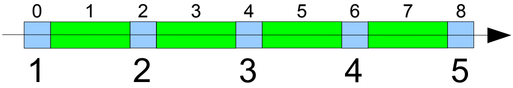


Bitmap, all cubes.



Bitmap, all cubes, dimensions.

1. Size of bitmap in x direction: 9. Size of bitmap in y direction: 7.
2. Let us determine the dimension of elements in positions 36, 49 and 32.
3. $\frac{36}{9} = 4$ reminder 0. $\frac{0}{7} = 0$ reminder 0.
4. $\frac{49}{9} = 5$ reminder 4. $\frac{5}{7} = 0$ reminder 5.
5. $\frac{32}{9} = 3$ reminder 5. $\frac{3}{7} = 0$ reminder 3.
6. Projection to odd coordinate – dim 0, even – dim 1.



Bitmap, all cubes, boundaries.

1. First we need to check in which directions the given cube C project to a non-degenerated interval.
2. And then, use those directions to compute boundary of C.
3. Boundary of elementary cube,
   $\partial C = \partial(I_1 \times \ldots \times I_n) = \sum_{i=1}^{n} I_1 \times \ldots \partial I_i \times \ldots \times I_n$.
4. ∂ of a cube number 29:
   4.1 $\frac{29}{9} = 3$ reminder 2. $\frac{2}{7} = 0$ reminder 2.
   4.2 Degenerated in x direction, non-degenerated in y direction.
   4.3 Boundary in position: $29 - 9 = 20$ and $29 + 9 = 38$ (adding one layer in direction of x).

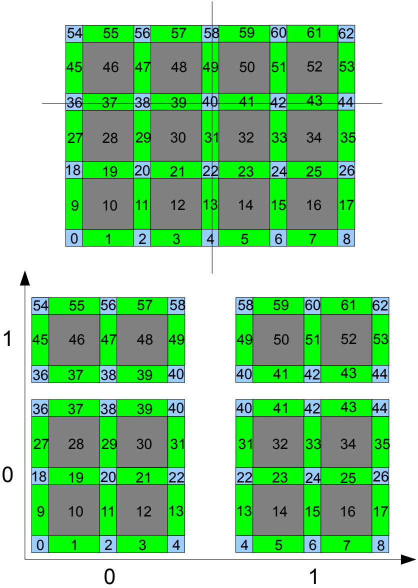
Spacial complexity.

1. For black and white bitmaps – one bite per cube.
2. For a filtered complexes – amount needed to keep filtration value per cube.
3. Boundary is computed from location in the structure.
4. Downsides – great for rectangular regions. Not effective to cover curvy objects.

Divide and Conquer.

1. Sometimes we may want to divide our data into smaller bits.
2. So that the size of intersection is minimized.
3. This is not an easy task for simplicial complexes.
4. There are ways to find minimal cuts based on heath equations.
5. But, this is trivial for bitmaps.

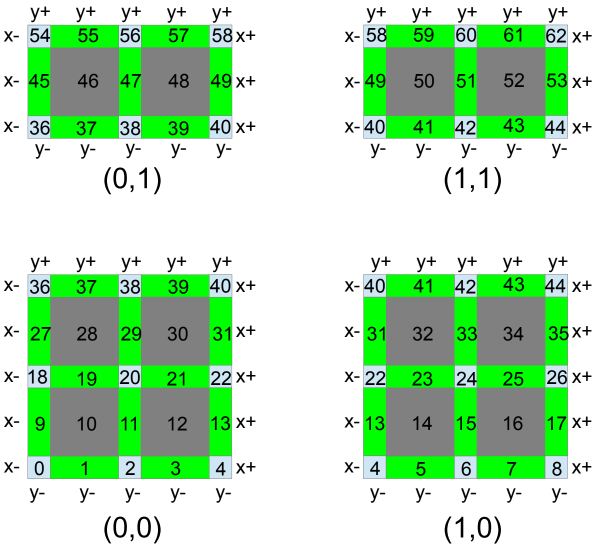
Dividing.



Dividing.

1. Suppose we read the big bitmap.
2. Elements in the sub-bitmaps appear in the order as they would in a bitmap.
3. Data streaming.

Gluing.

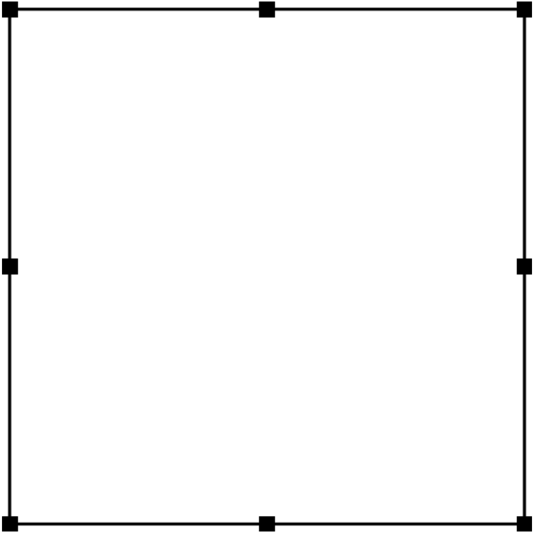


y+ y+ y+ y+ y+
x- 54 55 56 57 58 x+
x- 45 46 47 48 49 x+
x- 36 37 38 39 40 x+
y- y- y- y- y-
(0,1)
y+ y+ y+ y+ y+
x- 58 59 60 61 62 x+
x- 49 50 51 52 53 x+
x- 40 41 42 43 44 x+
y- y- y- y- y-
(1,1)
y+ y+ y+ y+ y+
x- 36 37 38 39 40 x+
x- 27 28 29 30 31 x+
x- 18 19 20 21 22 x+
x- 9 10 11 12 13 x+
x- 0 1 2 3 4 x+
y- y- y- y- y-
(0,0)
y+ y+ y+ y+ y+
x- 40 41 42 43 44 x+
x- 31 32 33 34 35 x+
x- 22 23 24 25 26 x+
x- 13 14 15 16 17 x+
x- 4 5 6 7 8 x+
y- y- y- y- y-
(1,0)

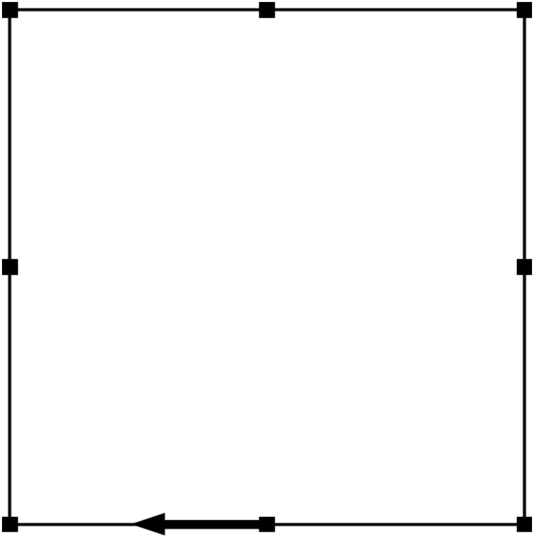
Where now.

1. We have covered basic idea of bitmap data structure.
2. It can be divided and glued back easily.
3. How to take advantage of that to get hierarchical, distributed algorithm to compute persistence?
4. For that we will need Discrete Morse Theory (DMT).

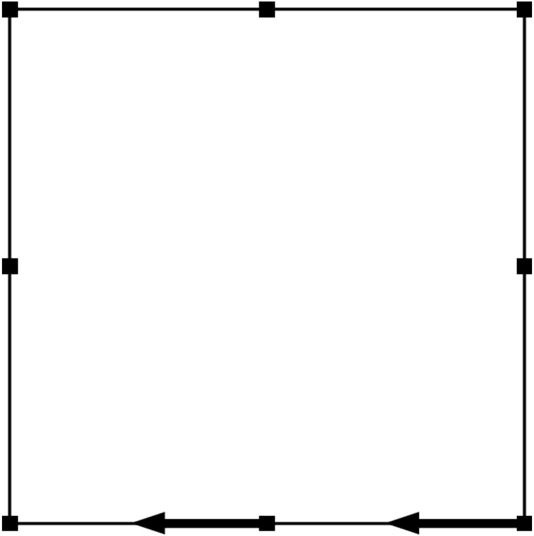
Discrete Morse Theory, Illustration



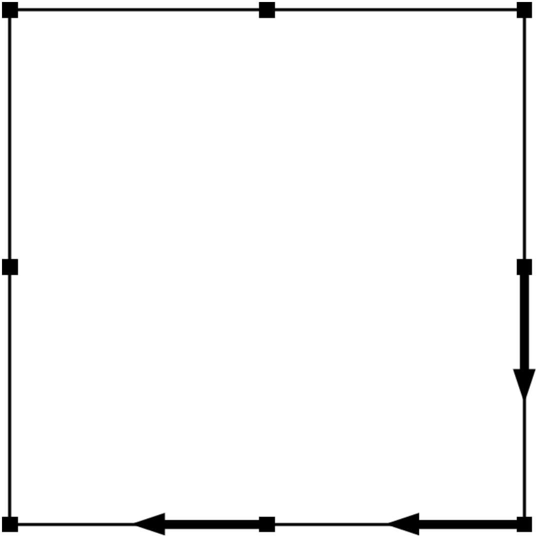
Discrete Morse Theory, Illustration



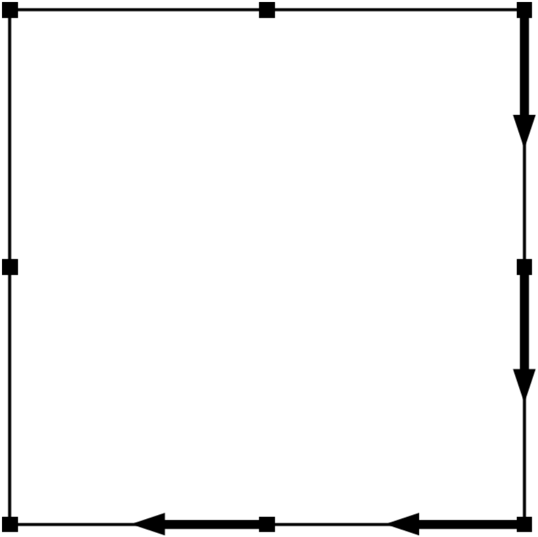
Discrete Morse Theory ,Illustration



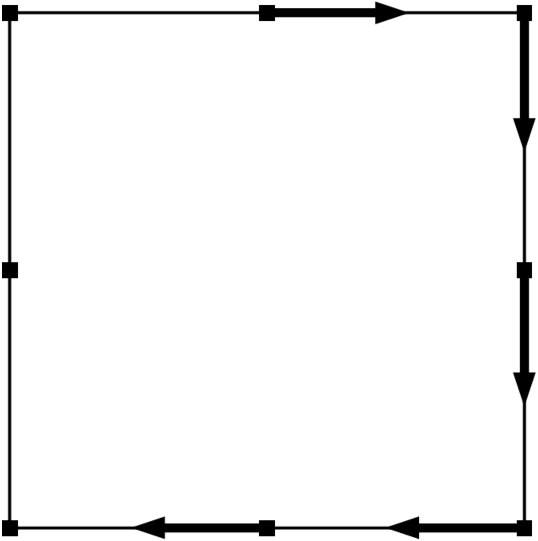
Discrete Morse Theory, Illustration



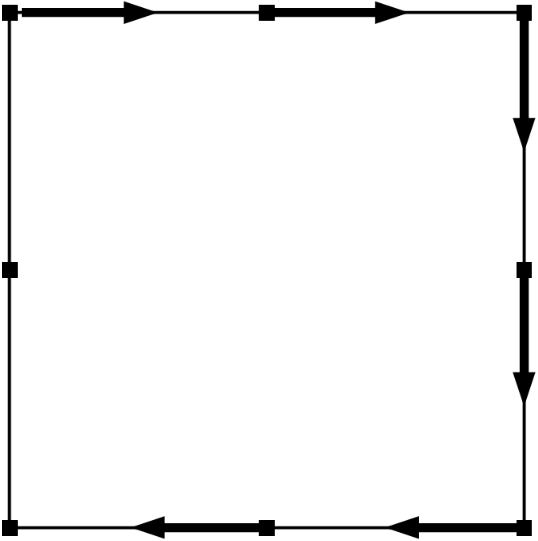
Discrete Morse Theory, Illustration



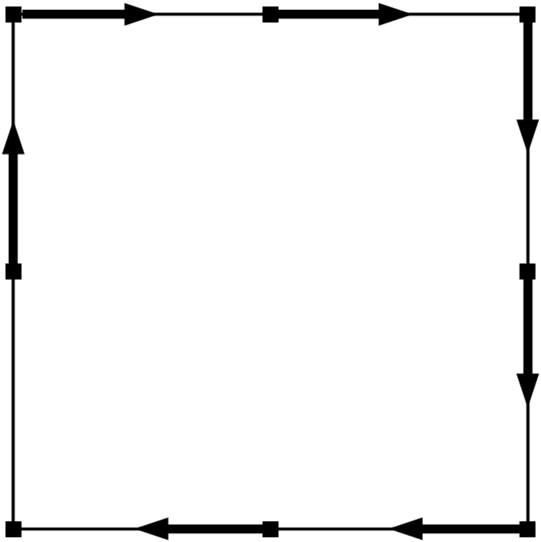
Discrete Morse Theory, Illustration



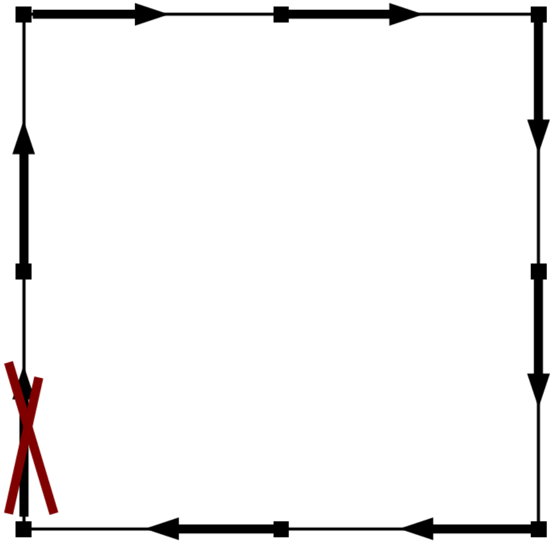
Discrete Morse Theory, Illustration



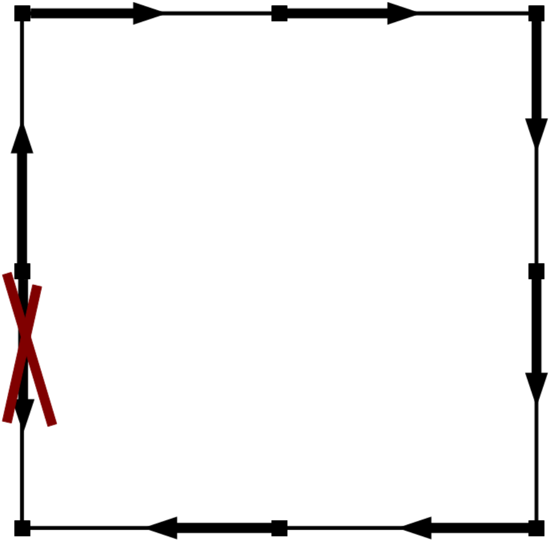
Discrete Morse Theory, Illustration



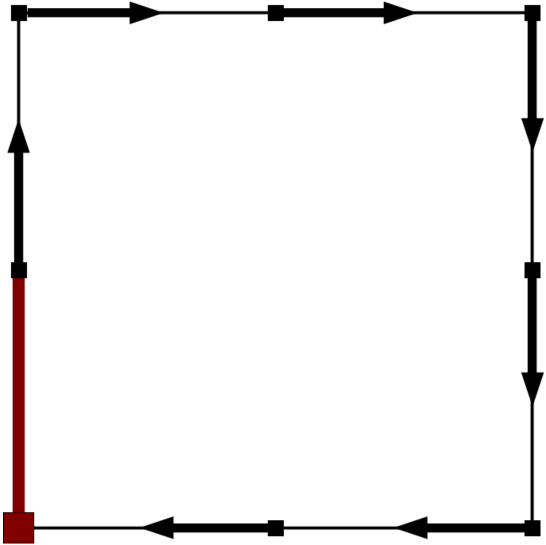
Discrete Morse Theory, Illustration



Discrete Morse Theory, Illustration

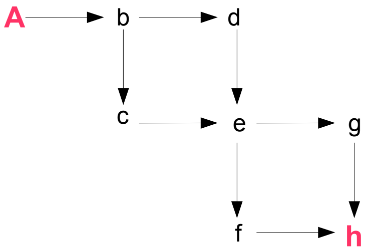


Discrete Morse Theory, Illustration



The Morse complex over $\mathbb{Z}_2$.

- Cells of Morse complex = critical cells of discrete vector field.
- Boundary relation computed by using gradient paths.
- Over $\mathbb{Z}_2$ – $\kappa(A, h)$ =number of gradient paths from A to h mod 2.
- Morse complex (over integers) and the initial complex are homotopically equivalent.
- Homology of a complex and its Morse complex - isomorphic.
- $\kappa(A, h) = 0$.



Why is Morse complex useful?

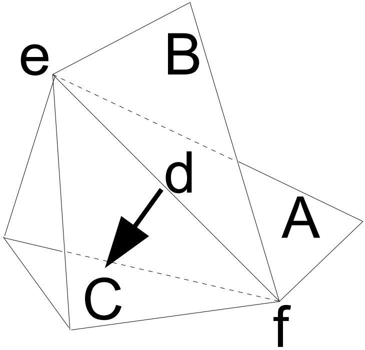
1. ($\mathbb{Z}_2$) homology of the initial complex and the Morse complex are isomorphic.
2. If pairings are made between elements in the same level of filtration, persistent homology is preserved.

How to do it algorithmically?

1. Two strategies: early and late boundary linking.
2. What I call early linking is a version of KMS algorithm.
3. Late linking require transversing DAG-s.

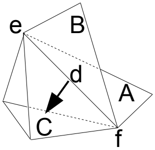
Early linking.

1. Do one pairing, compute boundary, do pairing, compute boundary, ...



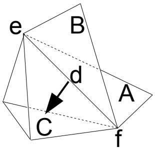
Early linking (over $\mathbb{Z}_2$).

1. Take care of C:
   1.1 $\delta(\partial C)+ = \delta d$.
   1.2 $\partial(\delta C)+ = \delta d$.
2. Take care of d:
   2.1 $\partial(\delta d)+ = \partial C$.
   2.2 $\delta(\partial d)+ = d$
3. Remove C and d from the complex.



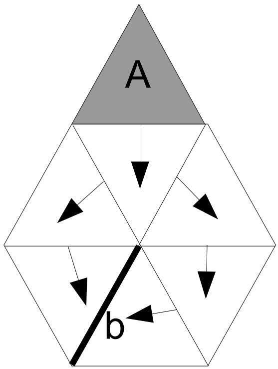
BTW...

1. Can you see matrix reduction here?
2. Suppose d is the lowest one for columns A, B and C (in this order).
3. Then $\partial(\delta C) + = \delta d$ is just standard column reduction.
4. But, when we want DMT, we need to keep track also on coboundaries.
5. This is why just for computing (persistent) homology it do not make sense to use DMT.



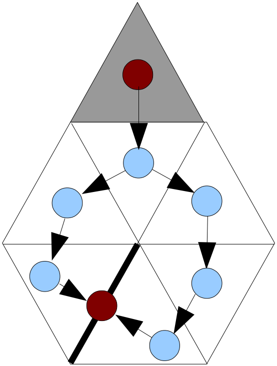
Late linking.

Construct admissible discrete vector field, leave boundary computations for later.



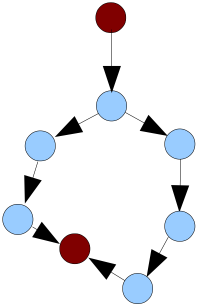
Late linking.

Graph based on pairings.

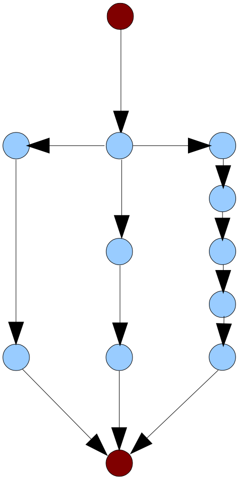


Late linking.

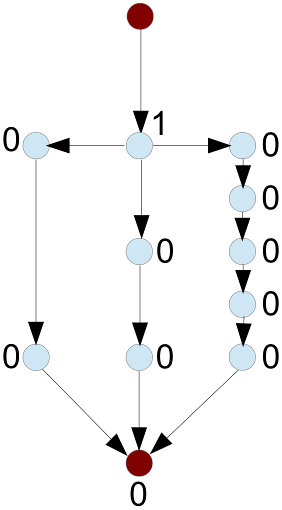
Graph.



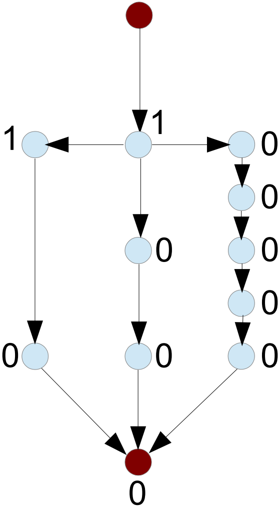
Late linking, boundary.



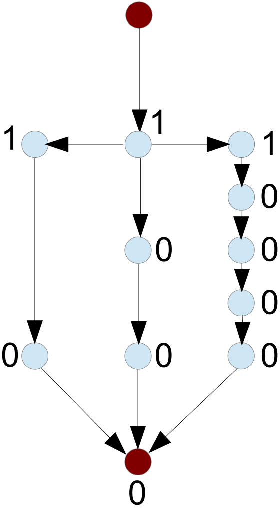
Late linking, boundary.



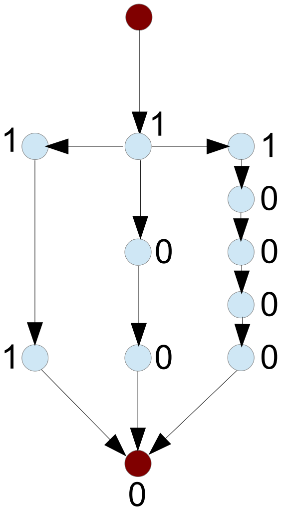
Late linking, boundary.



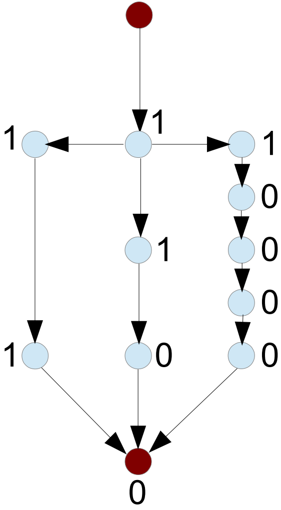
Late linking, boundary.



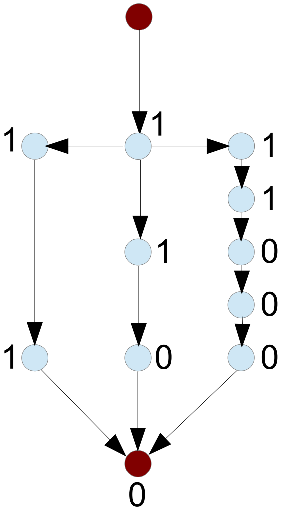
Late linking, boundary.



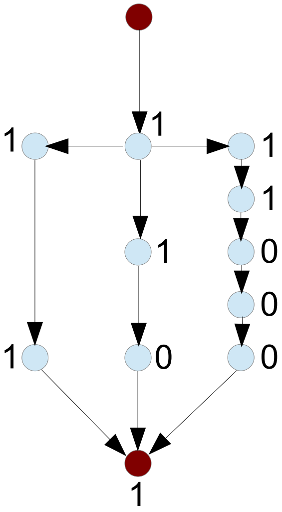
Late linking, boundary.



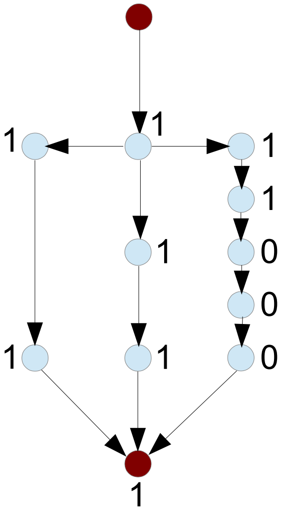
Late linking, boundary.



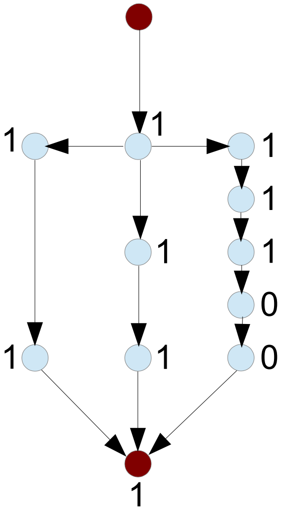
Late linking, boundary.



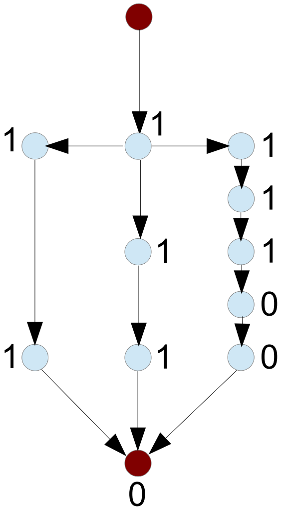
Late linking, boundary.



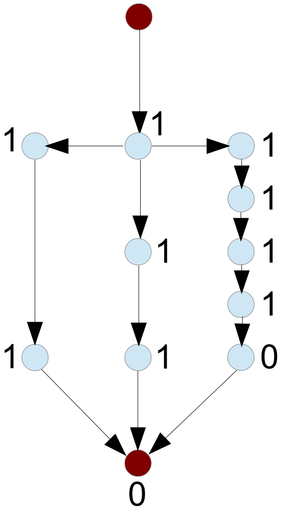
Late linking, boundary.



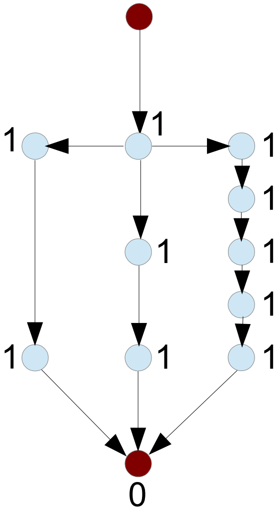
Late linking, boundary.



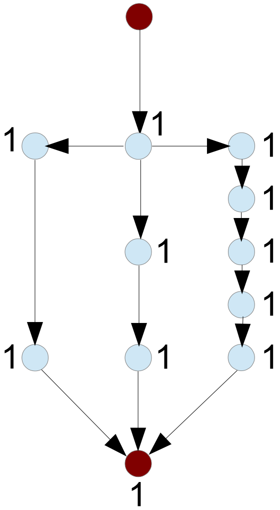
Late linking, boundary.



Late linking, boundary.



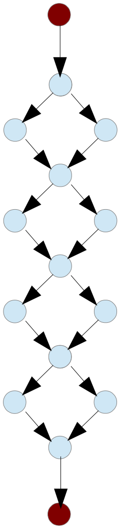
Late linking, boundary.



Late linking, boundary.

1. Note that I am not marking vertices as visited.
2. This BFS algorithm will terminate because we have DAG at the input.
3. Pessimistic exponential complexity.

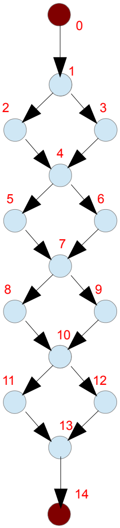
Late linking, pessimistic case.



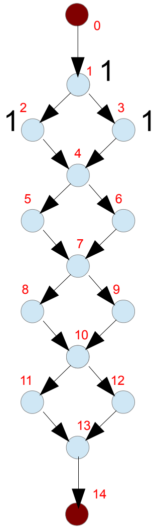
Late linking, boundary, dynamic programming.

1. Run a topsort on a graph.
2. Assign to each vertex its position from topsort.
3. Number of paths to from node s to the node $p =$ sum of number of paths to nodes $pred_1, \ldots, pred_n$, which have outgoing edge to p and are predecessors of p in the topsort order.

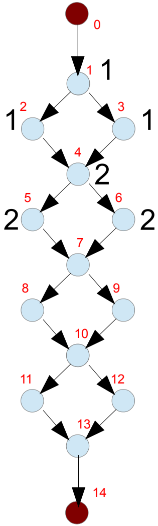
Late linking, topsort.



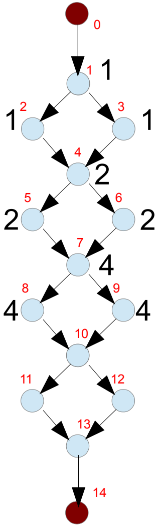
Late linking, number of paths.



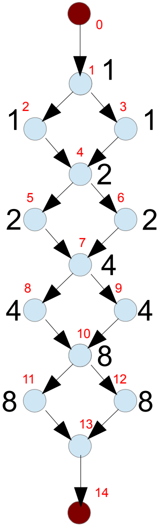
Late linking, number of paths.



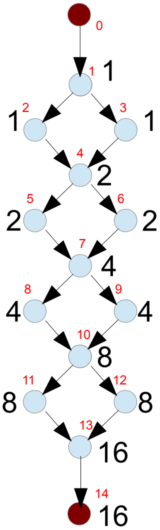
Late linking, number of paths.



Late linking, number of paths.



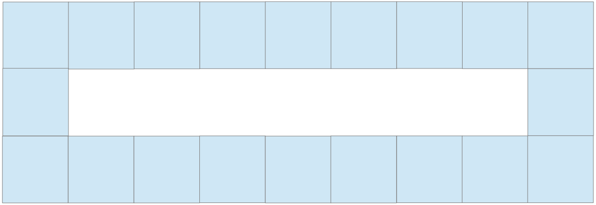
Late linking, number of paths.



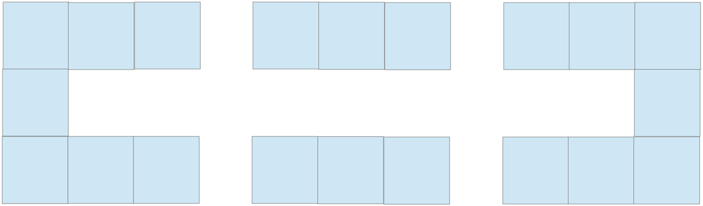
Multi-level distributed persistence, idea.

1. Divide the complex.
2. Construct a discrete Morse complex on subdivided pieces,
3. so that paths used to compute boundary do not go out from that piece and
4. we get globally correct discrete Morse complex.
5. (example for homology).

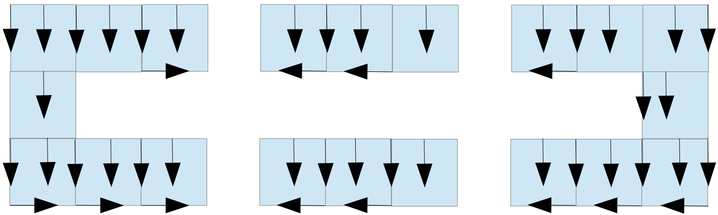
Multi-level distributed persistence.



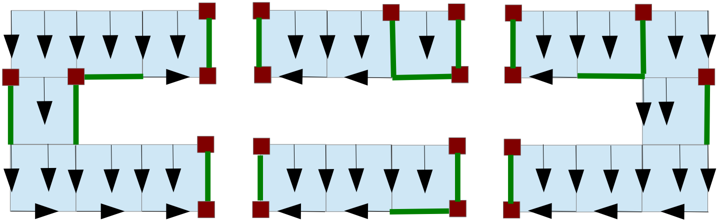
Multi-level distributed persistence.



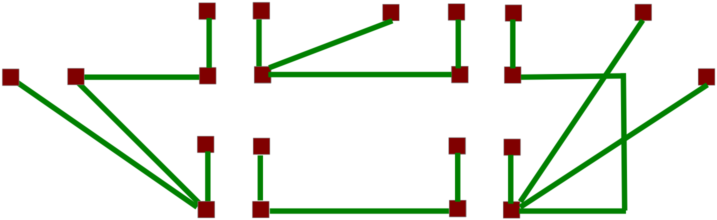
Multi-level distributed persistence.



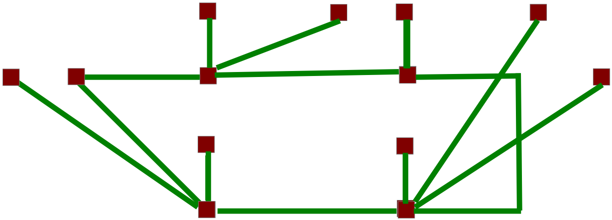
Multi-level distributed persistence.



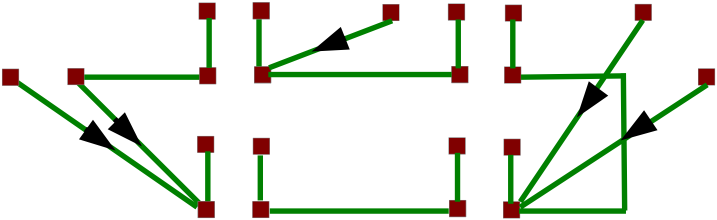
Multi-level distributed persistence.



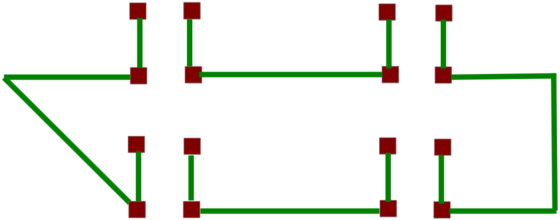
Multi-level distributed persistence.



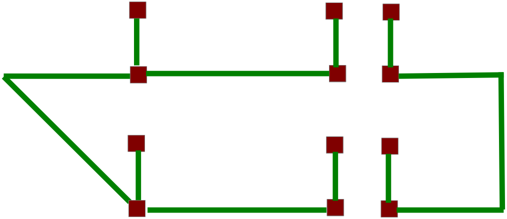
Multi-level distributed persistence.



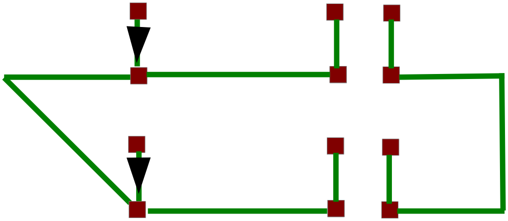
Multi-level distributed persistence.



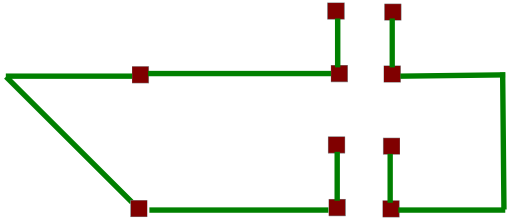
Multi-level distributed persistence.



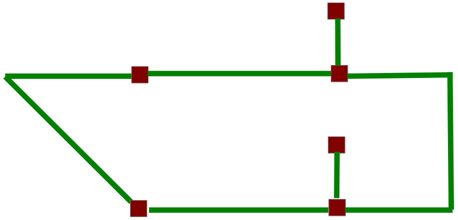
Multi-level distributed persistence.



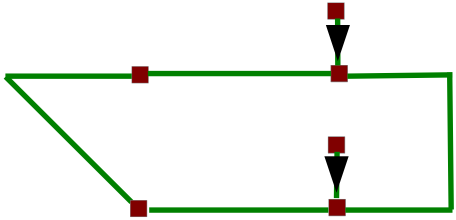
Multi-level distributed persistence.



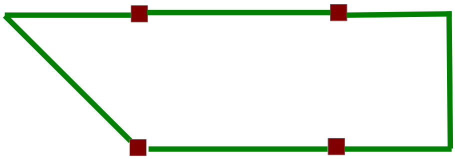
Multi-level distributed persistence.



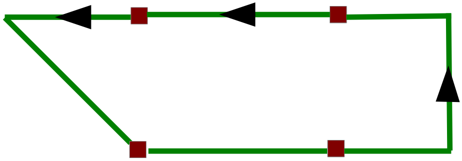
Multi-level distributed persistence.



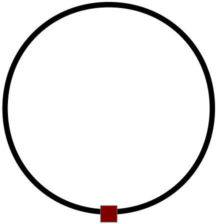
Multi-level distributed persistence.



Multi-level distributed persistence.

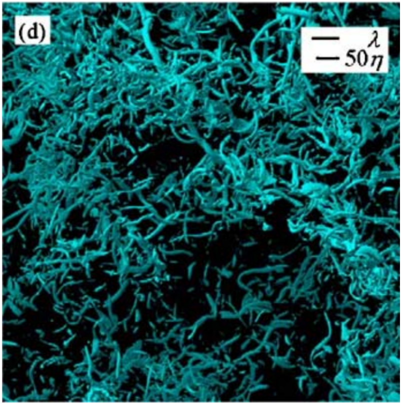


Multi-level distributed persistence.

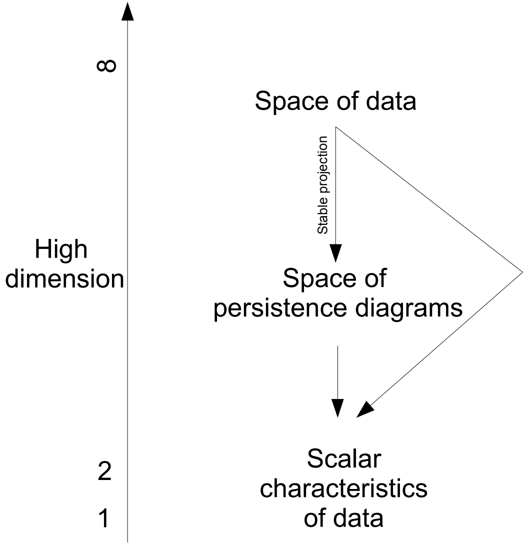


Applications.

1. Fluid dynamics, tracking high vorticity regions.



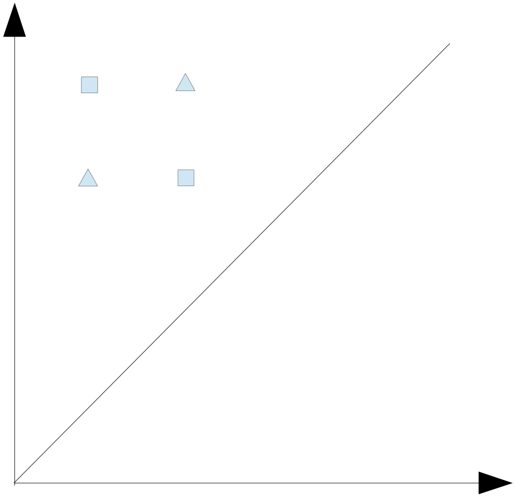
Dimension reduction.



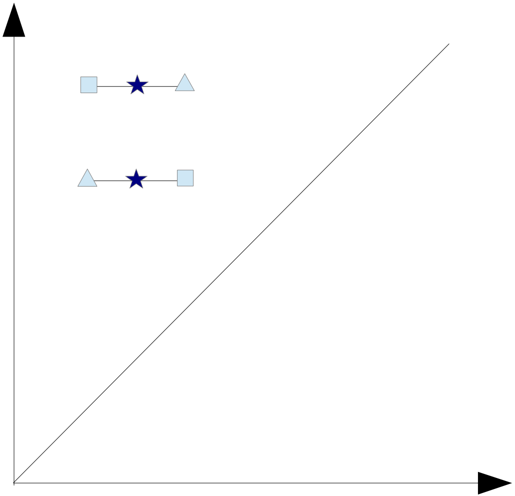
Dimension reduction.

1. Persistent homology is a stable dimension reduction technique that is useful in many applications.
2. What about doing statistics in the space of diagrams?
3. Problem: so far one were not able to do statistics on persistence.
4. Let us start with *mean*.

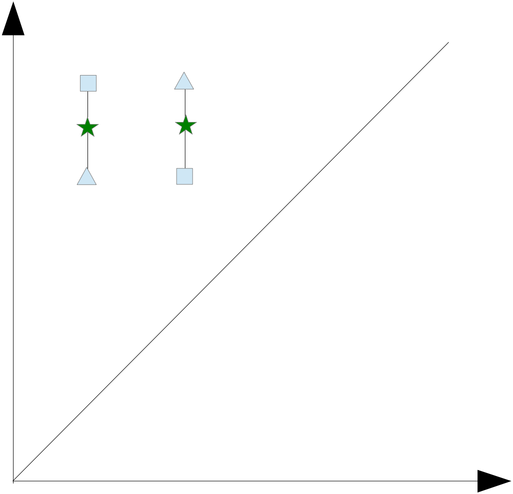
Problem with Frechet mean.



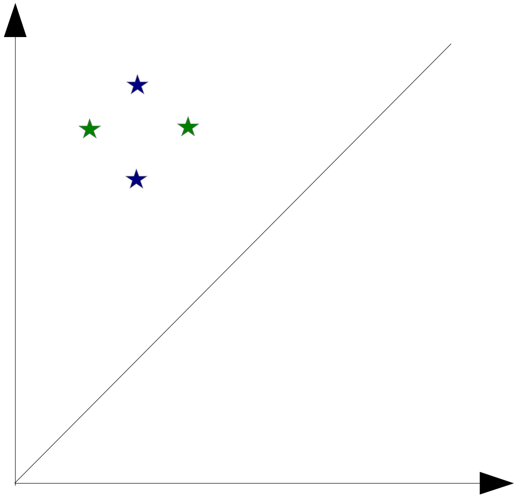
Problem with Frechet mean.



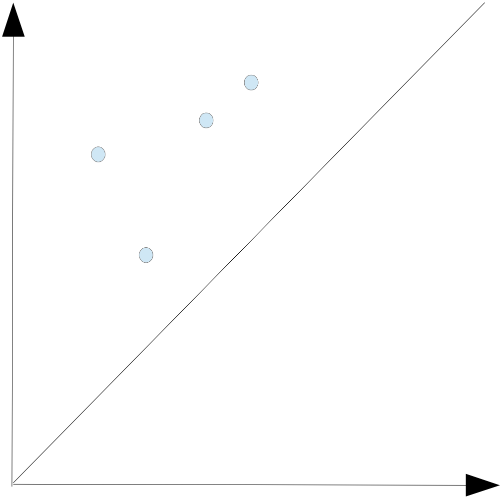
Problem with Frechet mean.



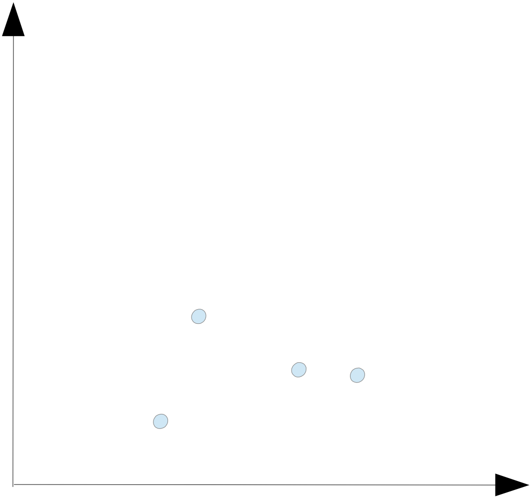
Problem with Frechet mean.



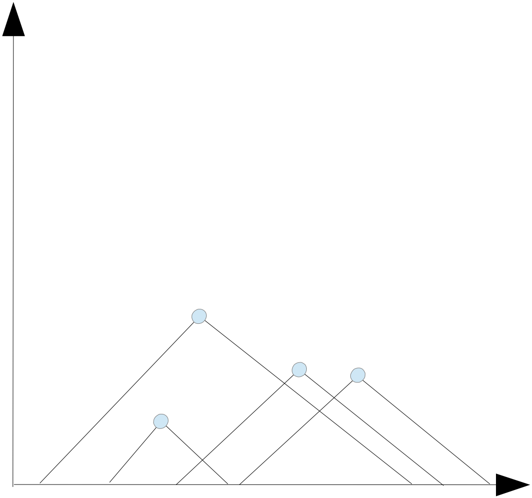
Persistence landscapes.



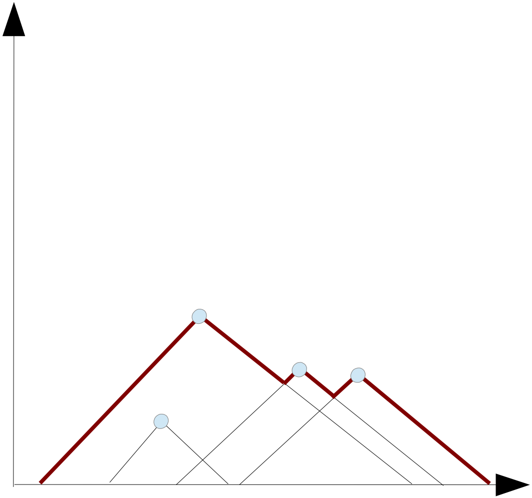
Persistence landscapes.



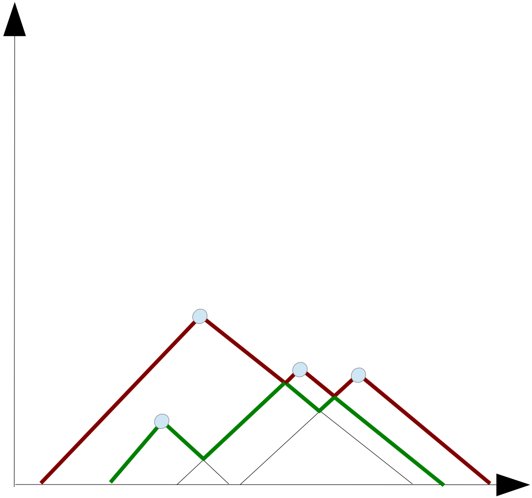
Persistence landscapes.



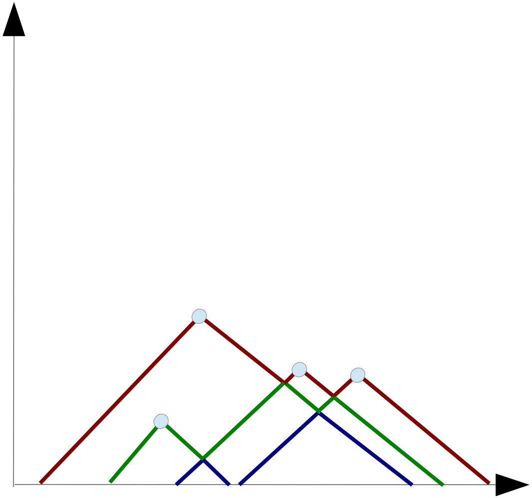
Persistence landscapes.



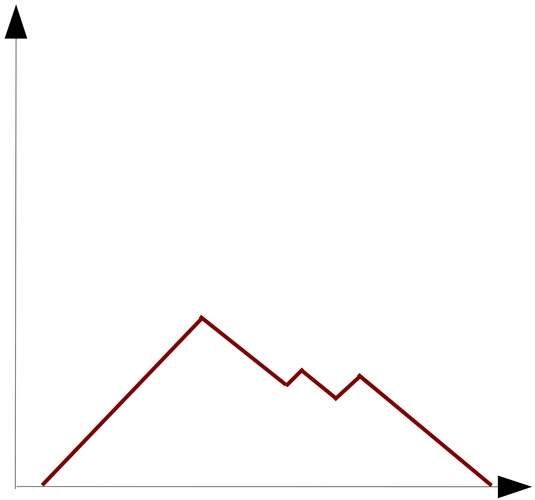
Persistence landscapes.



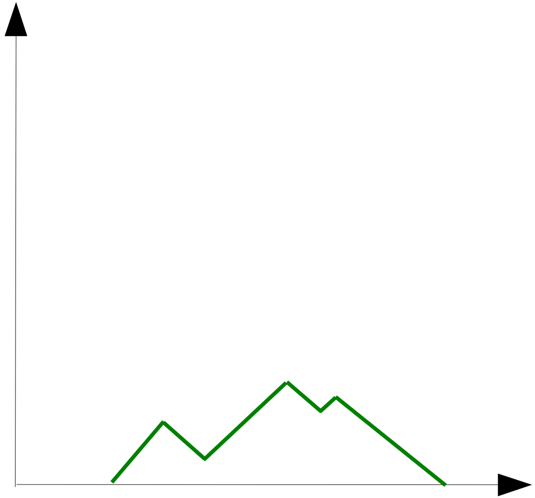
Persistence landscapes.



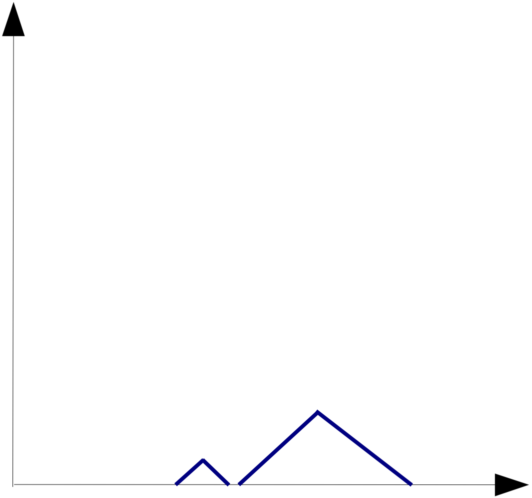
Persistence landscapes λ_1.



Persistence landscapes λ_2.



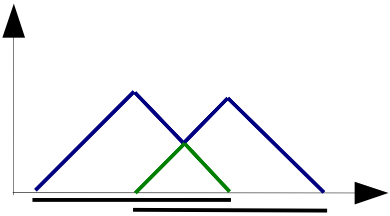
Persistence landscapes λ_3.



Formal definition.

1. The persistence landscape of a multiset of persistence barcodes $\{(b_i, d_i)\}_{i=1}^n$ is a set of functions $\lambda_k : R \to R$ such that $\lambda_k(x) =$ k-th largest value of $\{f(b_i, d_i)(x)\}_{i=1}^n$, where
2.

$$f_{(b,d)} = \begin{cases} 0 & \text{if } x \notin (b, d) \\ x - b & \text{if } x \in (b, \frac{b+d}{2}] \\ -x + d & \text{if } x \in (\frac{b+d}{2}, d) \end{cases} \tag{1}$$



Persistence landscapes.

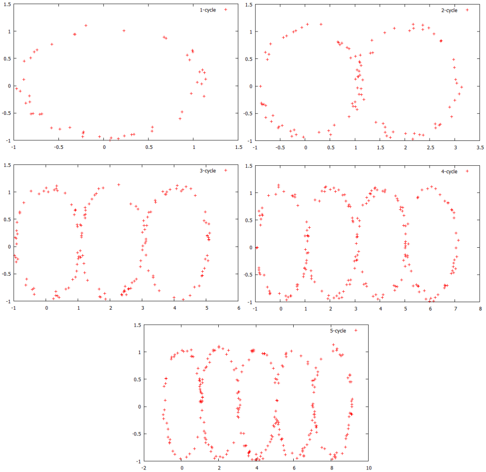
1. $1 - 1$ representation of persistence.
2. Vector space operations on functions $+, -$, multiplication by scalar well defined.
3. Average of two functions f, g in function space is just $\frac{f+g}{2}$.
4. Standard L^p norms and distances well defined.
5. PL-functions $\rightarrow$ easy to compute.

End-user programs to compute various statistics on Persistence landscapes.

1. Computations of distance matrix.
2. Computation of averages landscapes.
3. Standard deviation.
4. Computations of integrals.
5. Moments computations.
6. Permutation test.
7. T-test, anova.
8. Classifiers.
9. Normalization of barcodes.
10. Plots.

Let's check out the library!

1. Dataset: Let us sample 11 times $50n$ points from wedge of $n-$circles iid with some error.
2. Compute Rips complex and persistence of each of the point clouds.



How to obtain?

1. Go to http://www.math.upenn.edu/~dlotko/persistenceLandscape.html.
2. Linux, windows and osx executables which can perform most typical tasks are provided.
3. Source code (still a bit messy) for advanced users. A lot of comments are provided in the code.

What do you need first?

1. You need a persistence intervals in a form of a file:
   1 2
   4 5
   9 22
2. They can be obtained with various programs to compute persistent homology.
3. Dyinizous, JPlex, Perserus, Phat, Plex.
4. I do not yet have an input parser for Ghudi.

Distance matrix

1. Go to `http://www.math.upenn.edu/~dlotko/persistenceLandscape.html`.
2. Linux, windows and osx executables which can perform most typical tasks are provided.
3. Source code (still a bit messy) for advanced users. A lot of comments are provided in the code.
4. Construct to files with paths to the barcodes.
5. Call *DistanceMatrix* program.

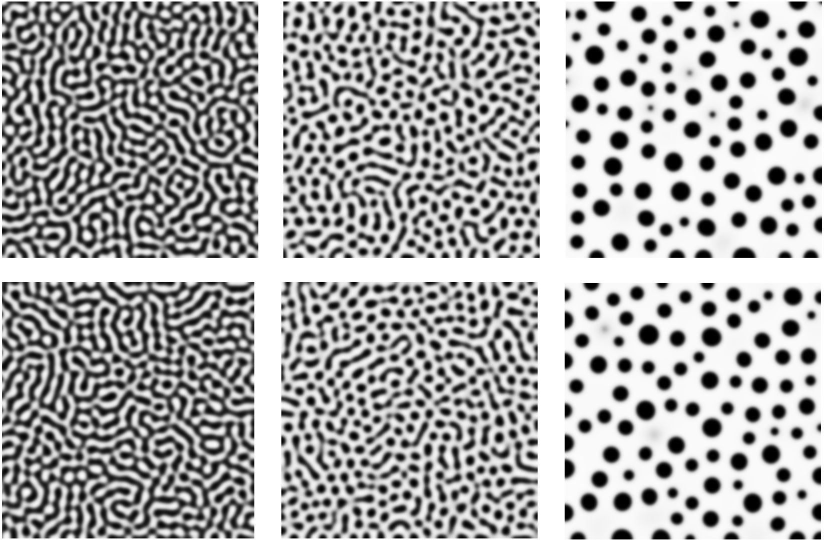
Others...

1. Let us try standard deviation (*StandardDeviation*),
2. Permutation test (*PermutationTest*),
3. Computations of averages (*ComputeAverage*),
4. Plotting subroutines (*PlotsOfLandscapesViaScripts*).
5. Classification (in dimension 1) (*ClassifierBasedOnSingleDimension*).

Applications overview.

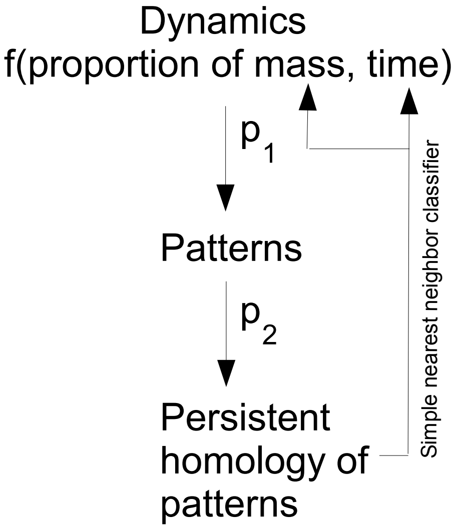
1. Patterns from numerical analysis (Cahn-Hiliard-Cook, Diblock-Copolymer equations).
2. Efficient distance matrix computations (granular media analysis).

Classification example – Cahn Hilliard Cook patterns.



50/50 **60/40** **75/25**

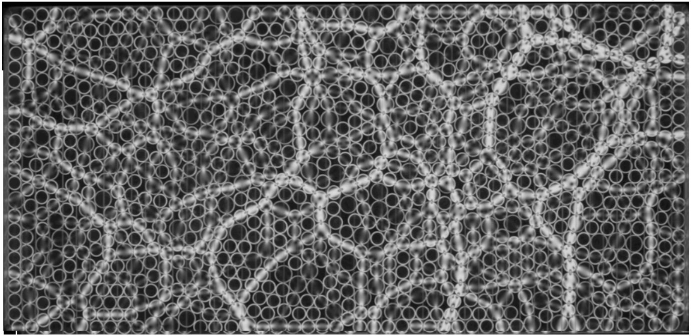
Topological classifier.



Granular media (by Miro Kramar and others).

1. *Granular media* – large conglomerations of discrete macroscopic particles.
2. Behaves differently from solids, liquids, or gases.

Granular media, R.P. Behringer



Kaboom!

Persistence and siloses.

1. A persistence is shown to be correlated with the forces inside the media.
2. A lot of comparison between persistence is needed to detect a threat.
3. Due to the current lack of efficient Bottleneck or Wasserstein distance computations, PLT plays an important role in this project.

More to come soon.

I hope...

Thank you for your time!

Pawel Dlotko
Inria, Saclay
pawel.dlotko@inria.fr
pawel_dlotko @ skype
pdlotko @ gmail