Computational Topology for Text Mining

ATMCS 2012, Edinburgh

Hubert Wagner

Jagiellonian University

July 9, 2012

What is it all about?

- Application of persistent homology.
- Context: text mining.
- I worked with text mining - internship at Google (2008).
- We got Google Research Award to apply computational topology in this context (2011-2012).
- Joint work with: Pawel Dlotko.
- Supervised by: Marian Mrozek and Witek Jarnicki (Google).
- This is already described in: HW, P.Dlotko, M.Mrozek, "Computational Topology for Text Mining", CTIC 2012.
- (Also supported by UE Programme: "Geometry and Topology in Physical Models").

Plan

- A real-world application of text mining - Google Alerts.
- Data representation [some concepts from text-mining].
- Results of computations.
- New graph-based algorithms [persistence + discrete Morse theory].

Practical example of text mining application

- Google Alerts (www.google.com/alerts)
- 'Monitor the Web for interesting new content'.
- You specify the query (topic, keywords).
- It 'googles' the given topic every day for you.
- Email notification when something *new* (fresh) if found.
- Problem: lots of spam: most results people got pointed to very *similar* webpages/documents.

Interesting property of natural text data.

- Zipf law, intuitively: relative frequency of the k-th most popular word is roughly 1/k.
- For a corpus of 10M distinct words:
  - k = 1 gives 6% (in English: 'the')
  - k = 2 gives 3% ('of')
  - k = 3 gives 2% ('to')
- 150 most popular words contribute to over 1/3 of all appearances.
- It works for all natural languages...
- Consequence: we are in the realm of complex networks, scale-free/small-world simplicial complexes.

How was the textual data represented?

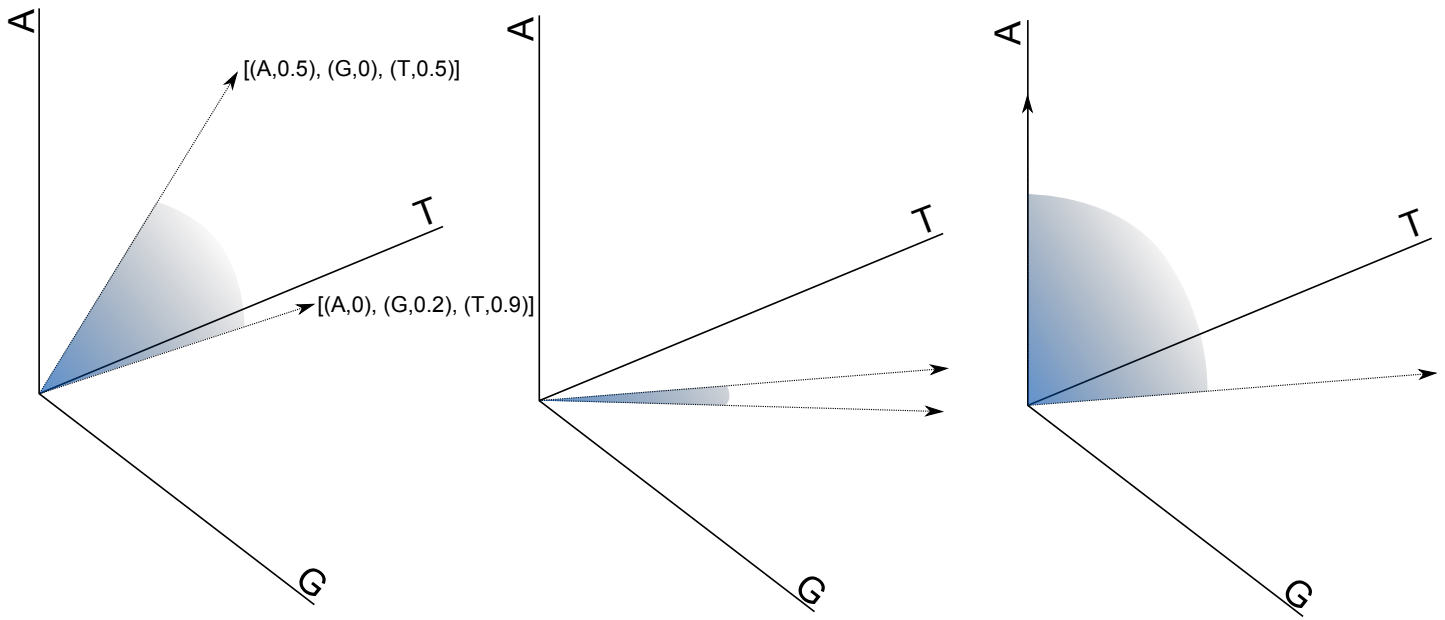
- *Corpus* = (large) set of (real-world) text documents
- Some well-known concepts from text mining:
- Term-vectors
- Vector Space Model
- Similarity

Concept: Term-vectors

- Used to extract characteristic words (or *terms*) from a document.
- Each term is weighted according to its relative 'importance'.
  - Words which appear often in a document are weighted higher. But this is offset by their global frequency. ('tf-idf')
- Document is represented as vector of pairs: $(term_i, weight_i)$.
- Examples: [('cats', 0.4), ('dogs', 0.7), ('food', 0.1)] or [('mice', 0.8), ('men', 0.1)]
- These vectors are called *term-vectors*.

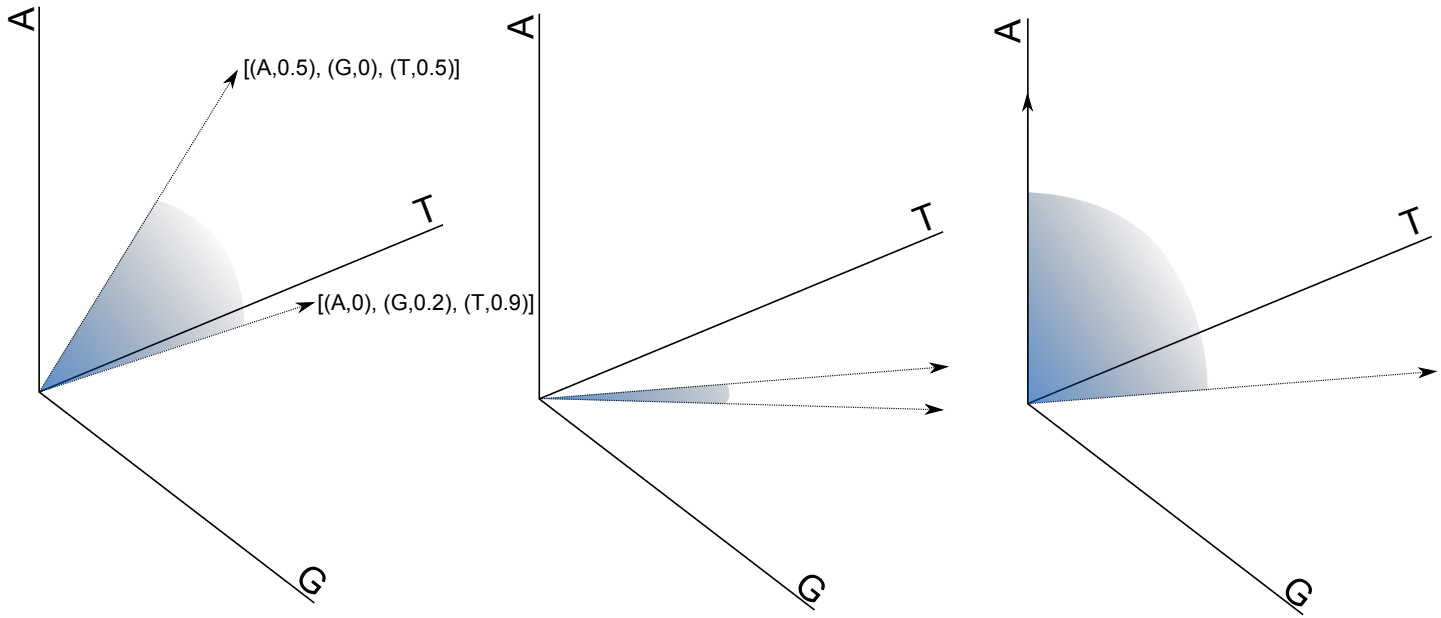
Concept: Vector Space Model

- *Vector Space Model* maps a corpus to $\mathbb{R}^d$.
- Each document is represented by its term-vector.
- Each unique term becomes a basis vector of this space, so the (embedding) dimension d can be very high.
- Term-vectors give the coordinates of documents in this space.



Concept: Vector Space Model and similarity

- We use the *cosine similarity*. Forget the Euclidean metric!
- Sometimes it's handy to talk about dissimilarity. We can define it as : $dsim(a, b) := 1 - sim(a, b)$.
- Dissimilarity is not a metric (no triangle inequality).



What do we want get out of this data?

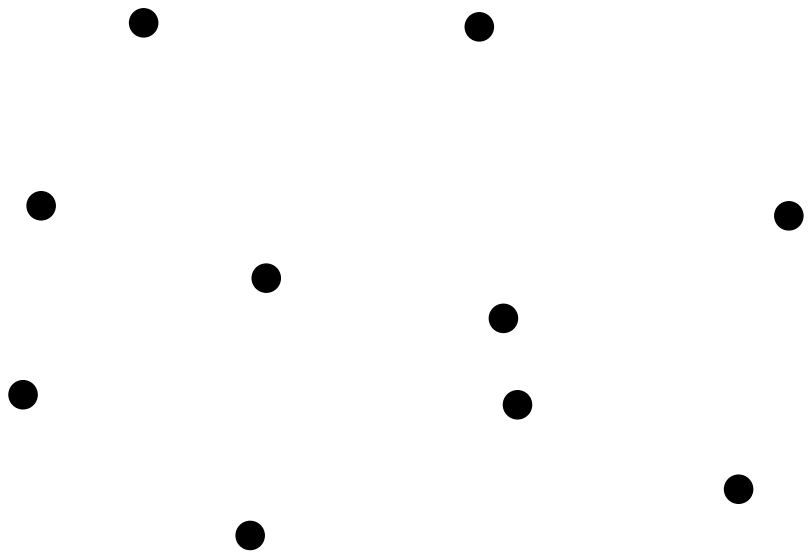
- We are interested in 'topology' of textual data in this representation.
- More precisely: in the structure of similarities among documents. Especially higher-dimensional similarities.
- We can compare different corpora (scientific articles vs sports news). Can we compare 'languages'? (We can match persistence diagrams in a stable way.)
- Can we simplify the topology - dimensionality reduction, manifold learning?

Our simplest experimental setting.

- We use documents from the English Wikipedia.
- Input: point cloud $\subset \mathbb{R}^d$
- Build filtered Rips complex up to some small dimension (edge weights are the dissimilarities).
- Compute persistence diagrams.

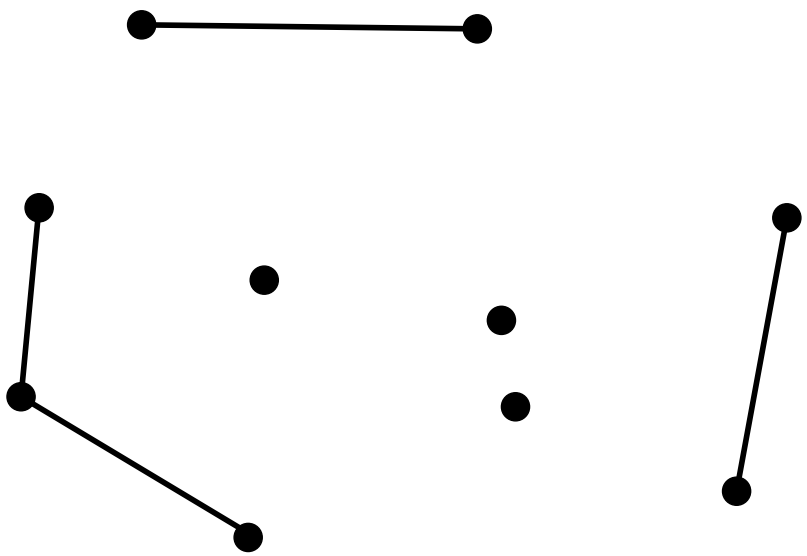
How do we construct the Rips complex?

- Edge (a,b) gets value = dsim(a,b), vertices get value 0.
- Higher dimensional simplices get the maximum value of their faces.
- Then we add simplices representing subsets of documents with increasing *dissimilarity*.
- Herbert's interpretation of what we do: "persistence describes the holes in our knowledge".



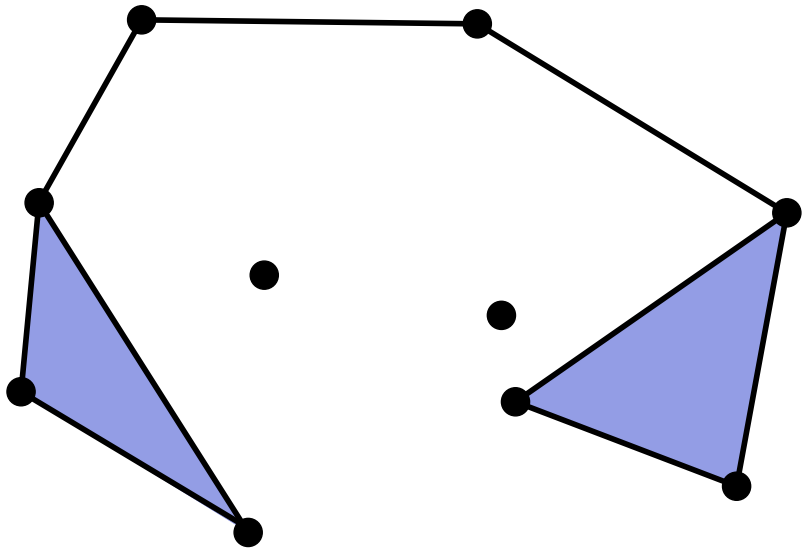
How do we construct the Rips complex?

- Edge (a,b) gets value = dsim(a,b), vertices get value 0.
- Higher dimensional simplices get the maximum value of their faces.
- Then we add simplices representing subsets of documents with increasing *dissimilarity*.
- Herbert's interpretation of what we do: "persistence describes the holes in our knowledge".



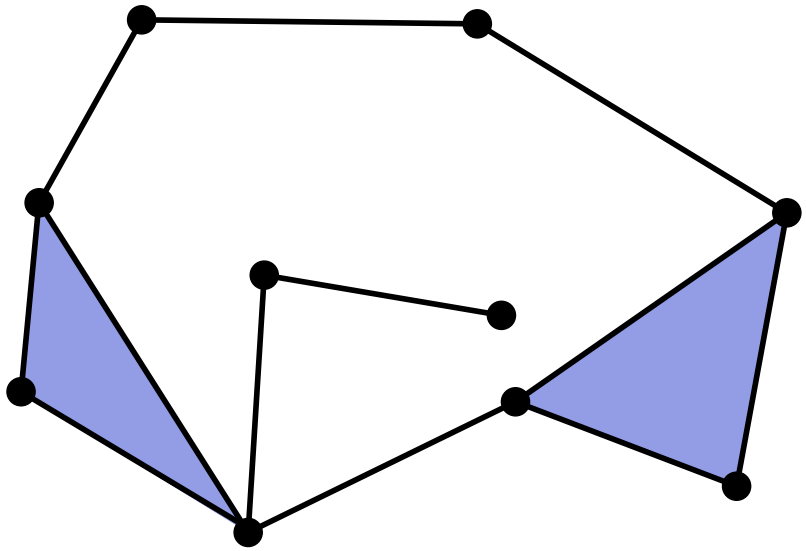
How do we construct the Rips complex?

- Edge (a,b) gets value = dsim(a,b), vertices get value 0.
- Higher dimensional simplices get the maximum value of their faces.
- Then we add simplices representing subsets of documents with increasing *dissimilarity*.
- Herbert's interpretation of what we do: "persistence describes the holes in our knowledge".



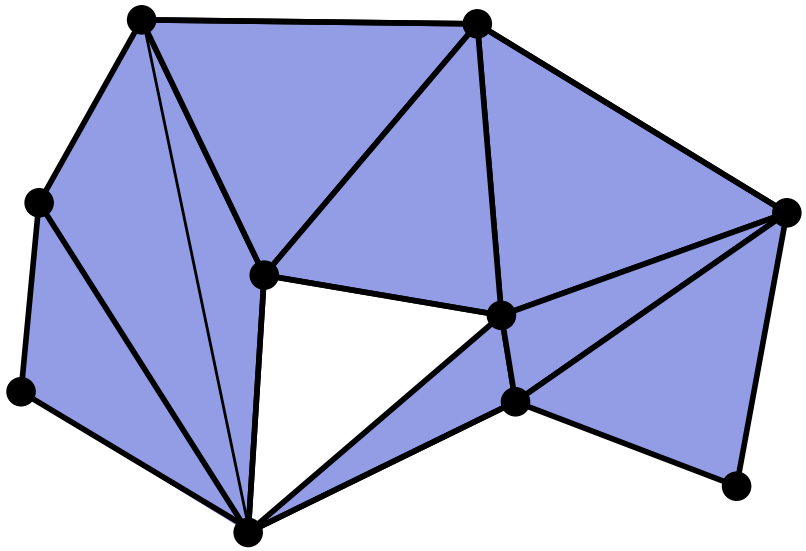
How do we construct the Rips complex?

- Edge (a,b) gets value = dsim(a,b), vertices get value 0.
- Higher dimensional simplices get the maximum value of their faces.
- Then we add simplices representing subsets of documents with increasing *dissimilarity*.
- Herbert's interpretation of what we do: "persistence describes the holes in our knowledge".



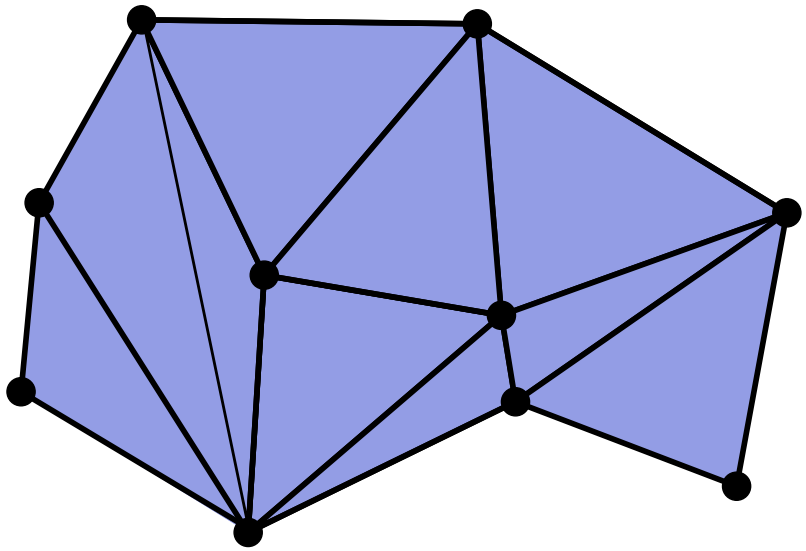
How do we construct the Rips complex?

- Edge (a,b) gets value = dsim(a,b), vertices get value 0.
- Higher dimensional simplices get the maximum value of their faces.
- Then we add simplices representing subsets of documents with increasing *dissimilarity*.
- Herbert's interpretation of what we do: "persistence describes the holes in our knowledge".



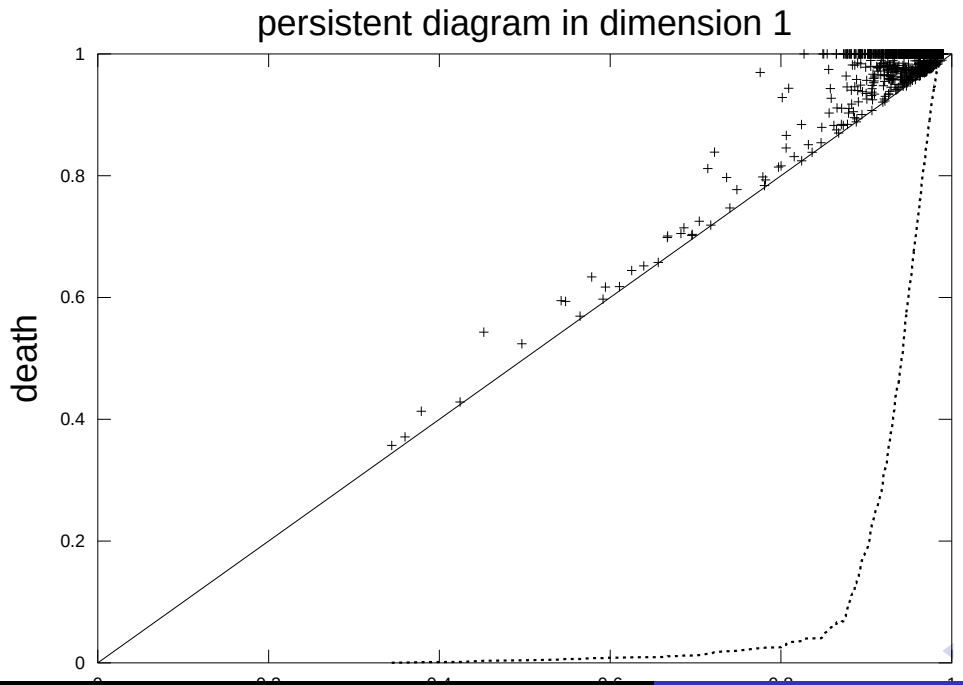
How do we construct the Rips complex?

- Edge (a,b) gets value = dsim(a,b), vertices get value 0.
- Higher dimensional simplices get the maximum value of their faces.
- Then we add simplices representing subsets of documents with increasing *dissimilarity*.
- Herbert's interpretation of what we do: "persistence describes the holes in our knowledge".



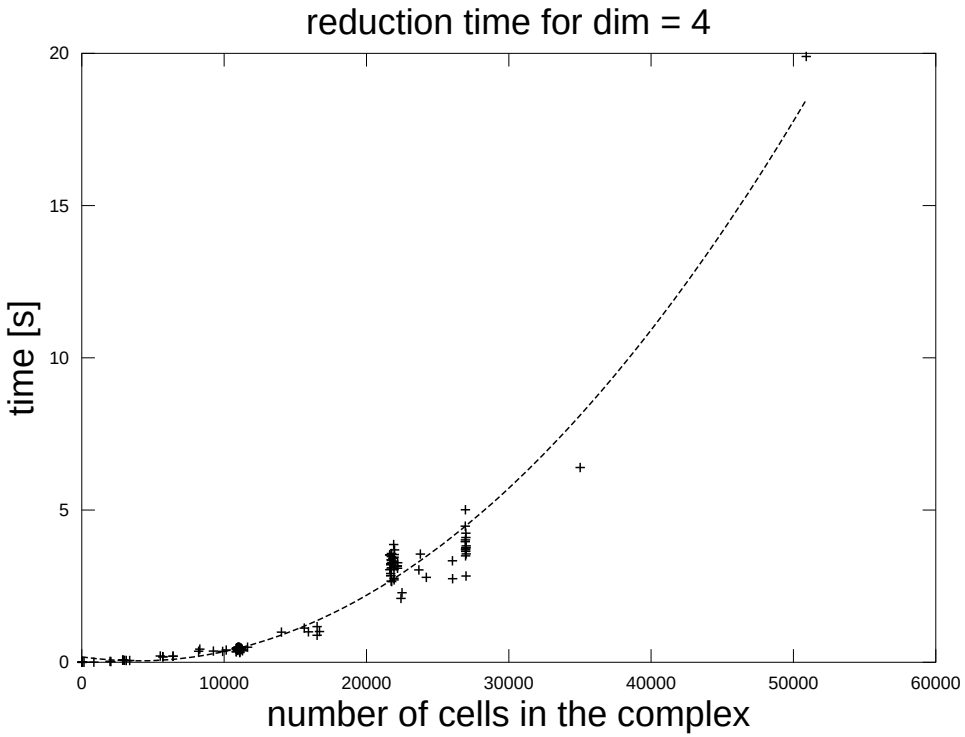
Computational results

- 1-dimensional persistence.
- The dotted graph shows cumulative persistence.
- Clearly topology gets more complicated if we allow dissimilarity ≥ 0.9.
- We don't want to analyze persistence diagrams of different dimensions in separation. Maybe it makes sense to treat the dimension as a (second) parameter of a filtration?



Computational results: "Can you do this for 10^{14} points?"

- Standard method to compute persistence: reduce the sorted boundary matrix.
- Efficiency is a problem for such datasets (quadratic scaling, worst-case is cubic).



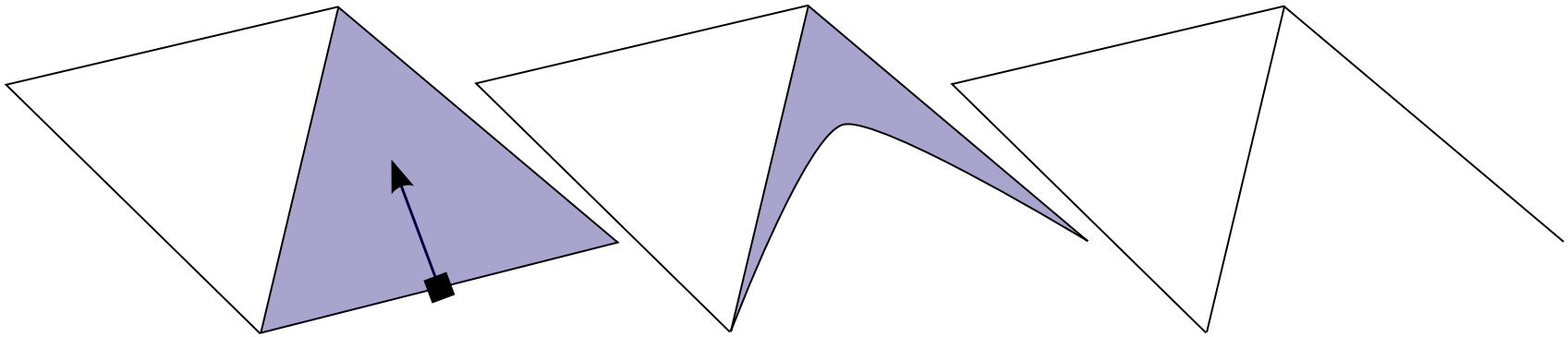
What are we aiming at?

- We want a *scalable* method to compute (persistent) homology.
- It should work for simplicial complexes in any dimension.
- Ideally, it should be based ONLY on graph theory.
- Well-developed, new remarkable results in approximate algorithms for matching etc.
- Google, Amazon... handle huge graphs. Ideally we could just adapt their tools...
- Discrete Morse theory is essentially graph theoretical (bipartite matchings, DAGs...)
- We show how we use DMT and bypass some problems.

Discrete Morse Theory

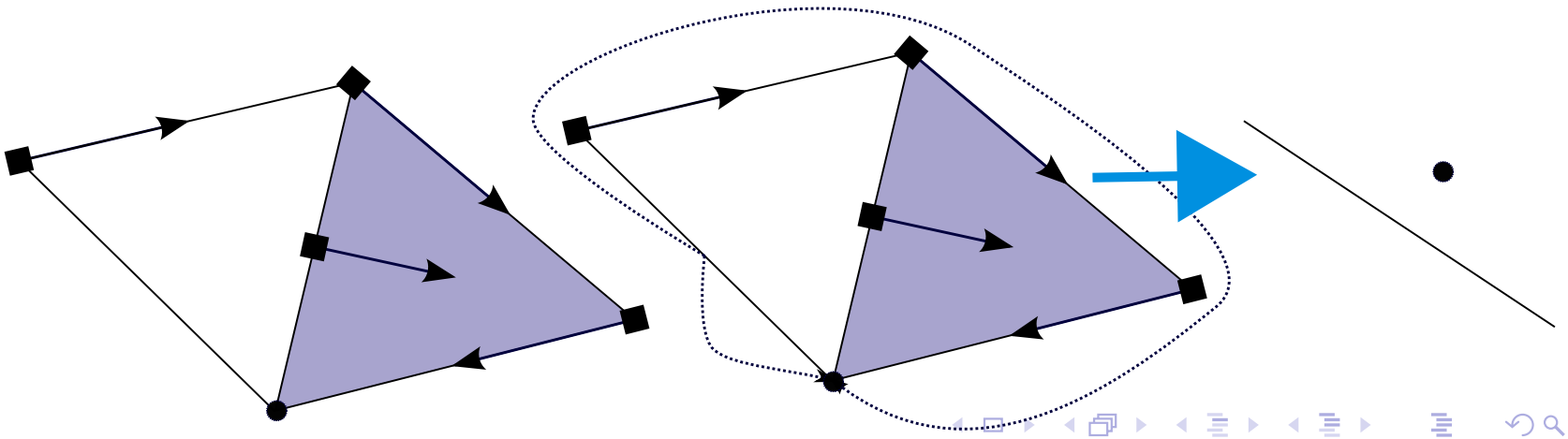
- DMT can be used to simplify the input complex: usually much smaller complex, the same (persistent) homology.
- Thomas Lewiner, Shawn Harker: algorithms for standard homology.
- Dmitri Feichtner-Kozlov: theory for arbitrary chain complexes [2005].
- Vanessa Robins et al: preprocessing for persistence, 'optimality' in 3D! [2011]
- Gunther, Reininghaus, HW, Hotz. Consequence: practical implementation for 3D cubical data. [2011]
- Recently: Vidit Nanda, Konstantin Mischaikow: preprocessing for persistence in any dimension.
- Abhishek Rathod: approximate (in some sense) algorithms for Morse complexes.
- We build on top of these and propose an algorithm to *compute* persistence without matrix reductions.

DMT: definitions and notation

- 
- The arrow denotes elements of the Morse matching (aka gradient vector field).
- Matching is done between cells of adjacent dimension (vertex-edge, edge-triangle etc.).
- If you like to think about Morse *functions*, it can be easily reconstructed from our matching/gradient.

DMT: Building the Morse complex

- We do homology with $\mathbb{Z}_2$ coefficients!
- Input: cell complex C. Three main phases:
  - Compute *any* acyclic Morse matching on C
  - Find the critical (unmatched) cells.
  - Compute the boundary relations between the critical cells.
    - For each critical cell follow the V-paths (boundary-arrow-boundary-arrow-...-boundary) leading to critical cells of lower dimension
- Critical cells with their boundary relations form a chain complex called the discrete *Morse complex*, having homology isomorphic with C [Forman, Kozlov].
-



DMT: problems

- Here we could just count the cells to get Betti numbers.
- Is it always so *perfect*?
- In general: no, we can get (many) additional cells.
- (The resulting boundary matrix can even be dense!)
- Constructing a *perfect* Morse matching is sometimes impossible.
- Example: *dunce hat*: contractible (we should get just 1 vertex) but non-collapsible [Whitehead?].
- Also, constructing an *optimal* ('as good as possible') Morse matching is NP-hard (even MAX-SNP-hard) [Lewiner, Lopez].

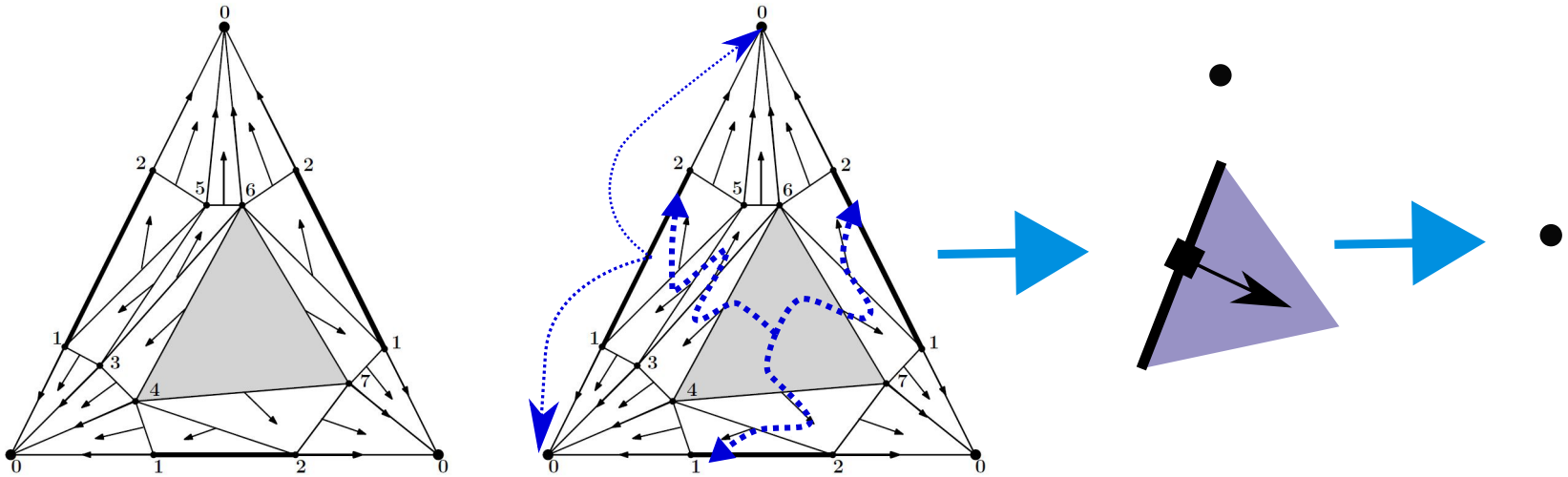
Iterated Morse construction

- We propose a new approach for computing homology with $\mathbb{Z}_2$ coefficients (or any finite filed).
- (Shaun Harker and Konstantin use similar techniques in their new paper.)
- We use the algebraic version of DMT from Kozlov's 2005 paper. Formally, Forman's theory is not enough!
- His theory works for arbitrary chain complexes.

Iterated discrete Morse construction

- Idea: *iterated* discrete Morse complex construction.
- (We build Morse complex of Morse complex of Morse complex ... of the input complex).
- Let M denote a functor which constructs a Morse complex of a given chain complex.
- Let C_0 be the input (simplicial) complex and $C_{i+1} = M(C_i)$, where M means constructing a Morse complex.
- We look at fixed point of M, which we call the *Seal* complex of C.

Example for 'Dunce hat'

- Reminder: single iteration computes a Morse complex, and has 3 phases:
  - Find *any* acyclic Morse matching on C_i
  - Find the critical (unmatched) cells
  - Computed boundary relations.
- C_{i+1} = critical cells with boundary relations.
- 

- (Leftmost image by Rafael Ayala et al.)

Iterated Morse construction

Lemma

- *(H.W, P. Dlotko, 2011) The resulting Seal complex contains only cells with null boundary.*

  - You can *always* read the $\mathbb{Z}_2$ Betti numbers by counting the number of cells.
  - Works in $O(n^3)$ worst-case time. (What about randomized?)
  - Contradictory with the mentioned results?
  - Hardness? No, the Seal complex is not a Morse complex *on the initial space*.
  - Non-collapsibility? No, it does not define a sequence of collapses *on the initial space*.

How can we use this method?

- To compute homology with field coefficients (no need to use matrix reduction).
- Our latest result: Computing persistence (no need for matrix reduction in the end).
- Our text-mining project was the main motivation. Now we start applying these results to get better efficiency.
- Also: Jonathan Heras et al. do validated computations of homology. Validating matrix operations is hard and slow, validating our algorithm is easy and fast.
- Straightforward consequence: Preprocessing for persistence. Optimal (smallest possible) resulting complex. Generalization of Robins' result, which worked for 3D.

Thank you! (Pawel Dlotko will talk more on using our method to compute persistence.)