Applications of computational (co)homology.

Paweł Dłotko
Institute of Computer Science, Jagiellonian University,
Kraków, Poland.

Fields Institute, Toronto, 9 November 2011.

Who am I?

- Finishing PhD Krakow, Poland.
- Working on the edge between disciplines (Mathematics, Computer Science, Engineering).
- Believe in topological nature of the Universe.
- Looking for some nice postdoc position and opportunities to work with nice peoples.

Where do I want to go today?

- Topology and Maxwell's equation.
- Distributed homology computations.

Topology and Maxwell's equations.

With Ruben Specogna and Francesco Trevisan.

Discrete Geometric Approach to Maxwell's equations.

- Formulation of physical laws of electromagnetism by using tools from algebraic topology on a mesh of a circuit.
- Mesh – topologically trivial, consist of conducting and insulating region.
- Idea – Build discrete theory on geometric elements of mesh and construct discrete counterparts of Maxwell's laws.
- Linear system instead of PDE's.
- Unknowns – values of discrete potential.
- Some Maxwell's law hold implicitly, some need to be enforced.
- By Enzo Tonti (1974).

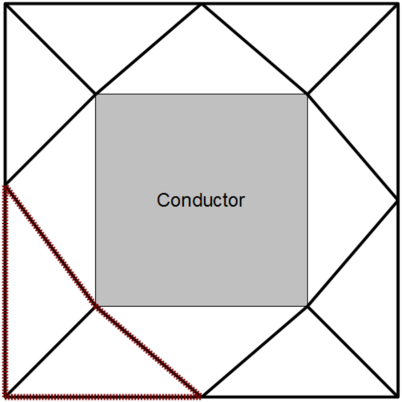
Discrete potential, idea.

- As in continuous case – not possible to define continuous potential (coboundary) on homologically nontrivial region.
- *Cuts* – places where potential is discontinuous need to be found.
- Edge-based elements, discontinuity on edges.
- Let's see the inconsistency based on Ampere's law when there are no cuts!

Local Ampere's law.

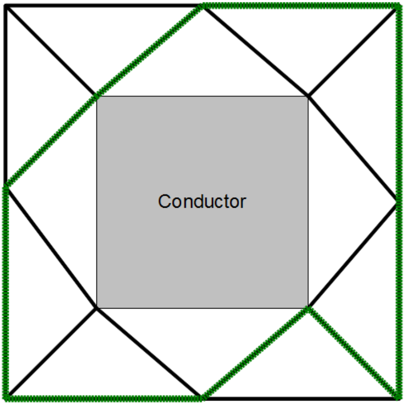
- $\mathbf{I}$ – electric current 2-cochain.
- $\mathbf{F}$ – magneto motive force 1-cochain.
- Local Ampere's law says: $\langle \mathbf{F}, \partial f \rangle = \langle \mathbf{I}, f \rangle$ for every face f in the mesh.
- Non-local Ampere's law says: $\langle \mathbf{F}, \partial c \rangle = \langle \mathbf{I}, c \rangle$ for every 2-chain in the mesh.
- OK for c being boundary, problem for homologically nontrivial c.

Non-local Ampere's law.



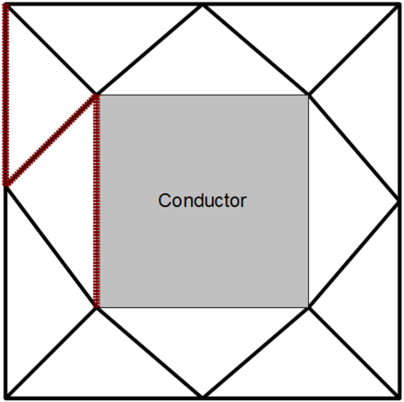
Ampere's law enforce zero on this cycle (fine, no current flow in air).

Non-local Ampere's law.



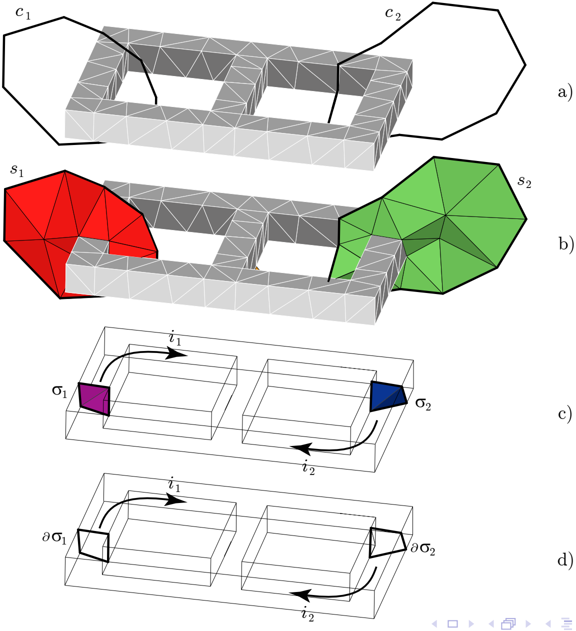
Ampere's law enforce zero on this cycle (wrong, the 2-cycle having red cycle as boundary have to cross conductor!).
This is inconsistency on Ampere's law.

Correction.



- Place some nonzero value ϵ on H^1 generator.
- Ampere's law will enforce ϵ current through conductor.

Panoramic view.



Summary.

- Suppose we want to impose Ampere's law on cycles in (a).
- In general the value of a current is nonzero there (b).
- We introduce a concept of `independent current` being a generator of $H_2(conductor, \partial conductor)$ (c).
- The user (engineer / designer) needs to choose the value for independent currents.
- They are extra degrees of freedom in the problem.

Backwards.

- We need to enforce Ampere's law on cycles in $[H_1(\mathit{insulator})]$ (d).
- To enforce Ampere's law on H_1(insulator) basis we use a dual H^1(insulator) basis elements (multiply the cochains by the value of independent current).
- In this way we impose in parts of conductor the current we want.
- **In practice we do it backwards** – start from H^1(insulator) basis, ending in independent currents.

Why we only care about discrete Ampere's law?

- What about:
  - Discrete current continuity law,
  - Discrete magnetic Gauss's law,
  - Discrete Faraday's law?
- They are all imposed once discrete Ampere's law is imposed.
- Long and technical discussion can be find in: P. D., R. Specogna, *Cohomology in electromagnetic modeling*, M3AS, under review.

Technicalities.

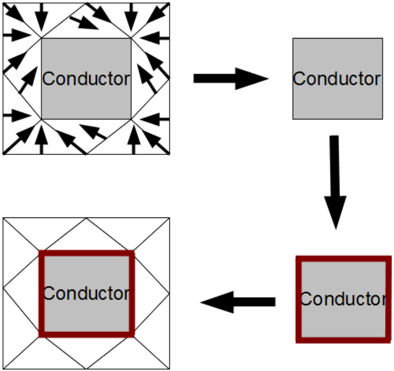
- Cohomology generators are provided as an input for EM solvers.
- They are used to fix the current through some parts of circuit.
- Heuristic methods to meet engineers requirements of generators with minimal support are being developed (P.D, R. Specogna).

How to compute **co**homology gens?

- Easy! Simply use the esisting code for homology computations, but "transpose" the incidence index.
- This would clearly work for SNF ($\partial^T = \delta$).
- However computing SNF for real-world complexes is not the best idea.

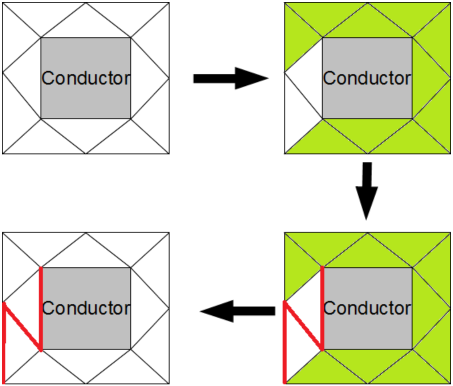
Shavings.

- *Shaving* is a kind of reduction that preserves generators.
- Elementary (Whitehead) reduction is a shaving for homology.



Shavings.

- Clearly Whitehead's reductions are not shaving for cohomology
- Removing acyclic subspace is!



Perspectives.

- Very good combination of EM and cohomology code.
- Planning to use it in modeling plasma inside ITER fusion device (finally, after 4 years we have its mesh!).
- Code for (co)homology computations for hybrid meshes (hexahera, tetrahedra, pyramids,...) – in fact we can handle any regular CW-complex (Thomas Wanner was to talk about this...)
- Cohomology – useful in many other fields – from texture matching in graphics to obstacle avoidance in robotics.
- Look for others nice applications of cohomology. We already have code and some experience. Everyone's invited!

Distributed computations of (co)homology over field.

(work in progress)

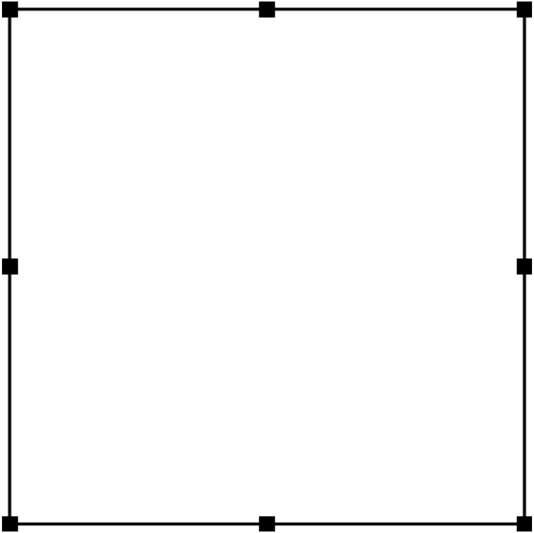
Beginning

- P.D. M. Mrozek and H. Wagner – make a few people in Google interested in application of topology for text mining.
- Problem we have faced – how to compute homology for **huge** point clouds?
- Working on highly efficient C++ implementation of Flag complexes.
- Use Discrete Morse Theory to save as much memory during complex construction as possible.
- Still – size of RAM memory is our limitation.
- Even with largest computers available we cannot handle the data of interest of Google.
- Way out – distribute computations – do far we have some experience from sensor networks.

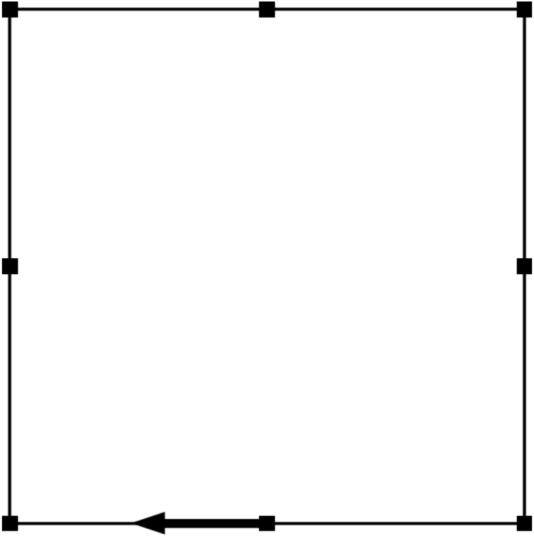
Beginning

- P.D. M. Mrozek and H. Wagner – make a few people in Google interested in application of topology for text mining.
- Problem we have faced – how to compute homology for **huge** point clouds?
- Working on highly efficient C++ implementation of Flag complexes.
- Use **Discrete Morse Theory** to save as much memory during complex construction as possible.
- Still – size of memory is our limitation.
- Even with largest computers available we cannot handle the data of interest of Google.
- Way out – distribute computations – do far we have some experience from sensor networks.

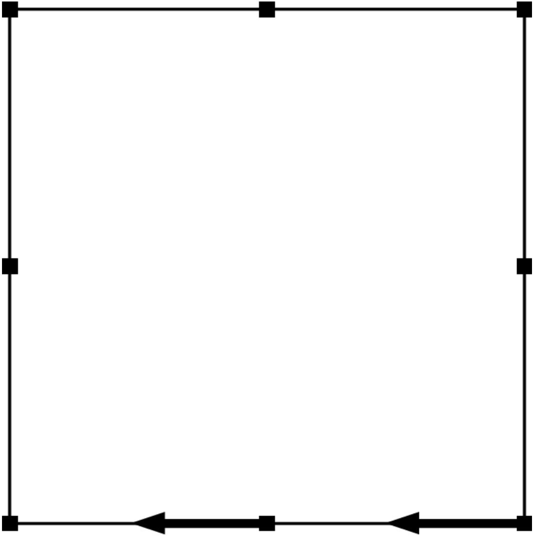
Illustration



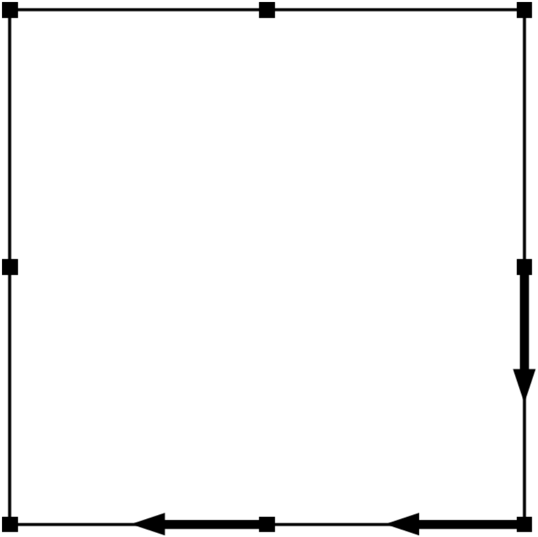
Illustration



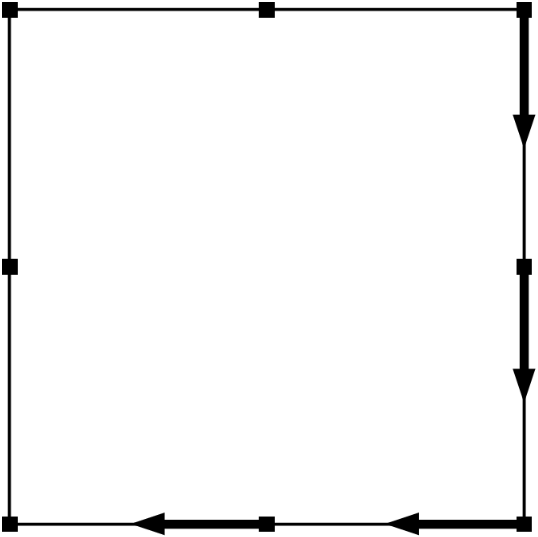
Illustration



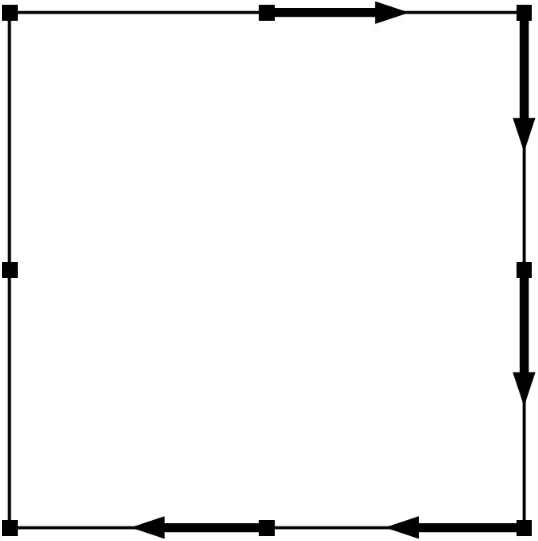
Illustration



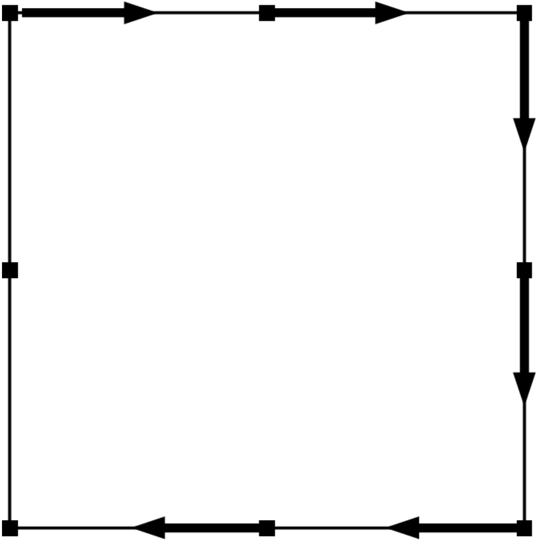
Illustration



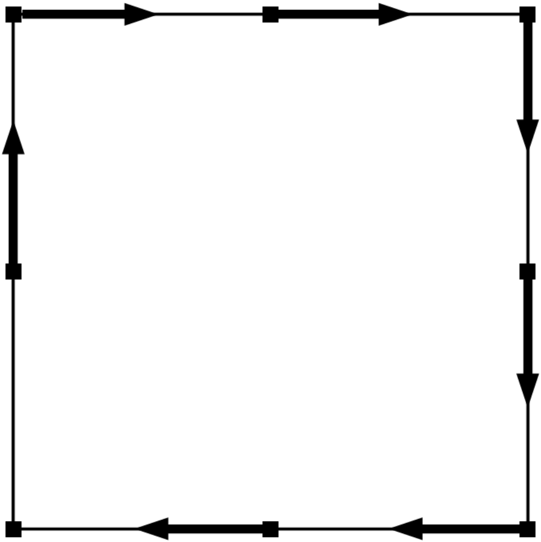
Illustration



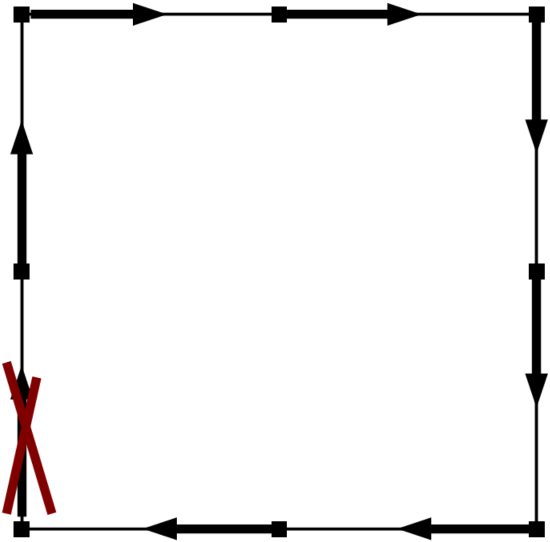
Illustration



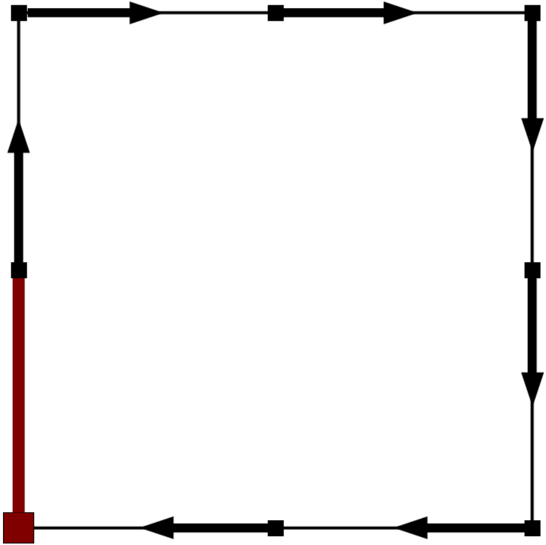
Illustration



Illustration



Illustration



Morse complex.

- Theory by Robin Forman.
- Cells of Morse complex – critical cells of original one.
- By looking at gradient flow path p we can see how orientation of a cell A induces the indicated orientation on another cell B - $o(p, A, B)$.
- Incidence between cells A and B in Morse complex: $\sum o(p, A, B)$ for every p joining A and B.
- Homology of Morse complex are homology of initial complex.
- Discrete Morse theory as presented by Robin Forman – already used to compute homology (Thomas Leviner).

Towards distributed computations, speculations.

- Observation – a single Morse pairing is very "local".
- Suppose we have a way of building Morse pairings in a distributed way...
- ...so that no closed V-paths appears.
- Then only hard to distribute part of the computations is computing incidence of cells in Morse Complex.
- What if we do the pairings in a way, that it is easy to get incidence?

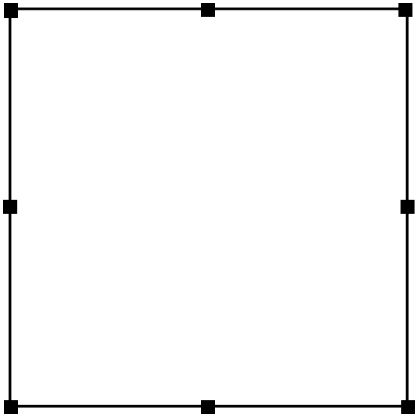
Towards distributed computations, fact.

- Suppose we iteratively compute Morse complexes so that at least one pair is created at each step,
- then after some number of iteration the process stabilizes,
- in this way we can obtain homology over a field.

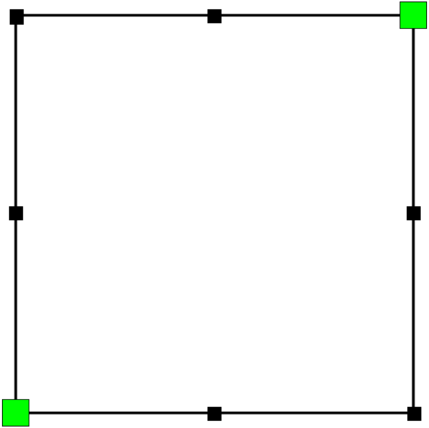
Cone contraction algorithm.

1. Boundaryless cell with nonempty coboundary – `cone`.
2. Let us have a set of cones in our complex lying at least 3 hops one from another.
3. Each process work on a single cone and simplices with all vertices lying no further than 2 from a single cone.
4. A Morse contraction among simplices incidental to cone is made.
5. Then the state of the complex is written back to hard disk.
6. When there are no more cones, finish.

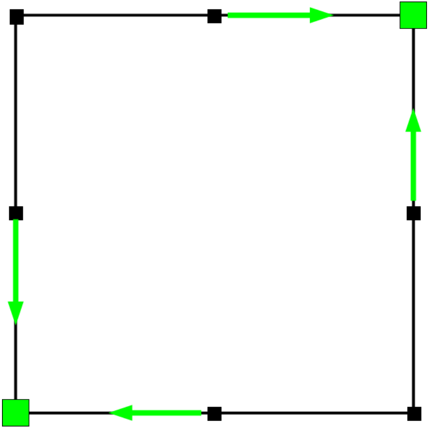
Simple graph example.



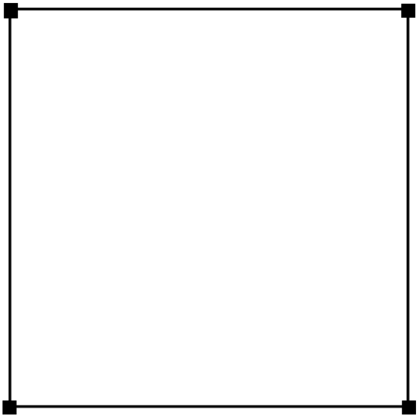
Simple graph example.



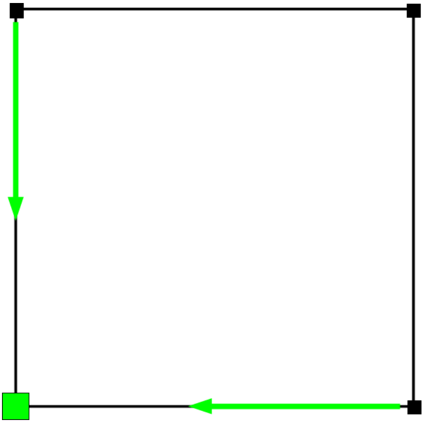
Simple graph example.



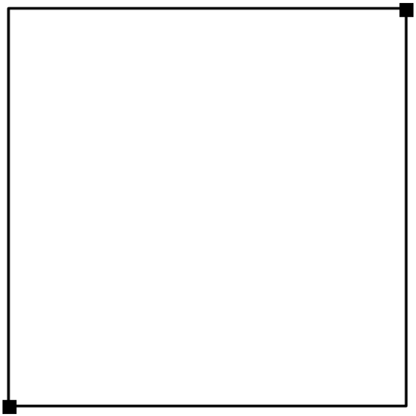
Simple graph example.



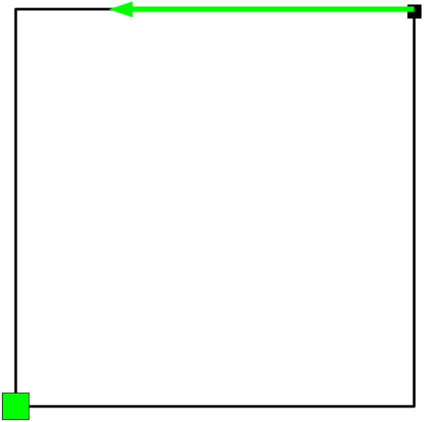
Simple graph example.



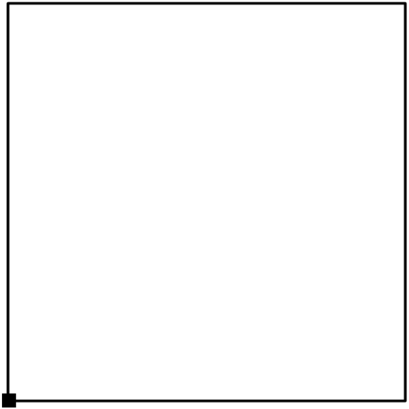
Simple graph example.



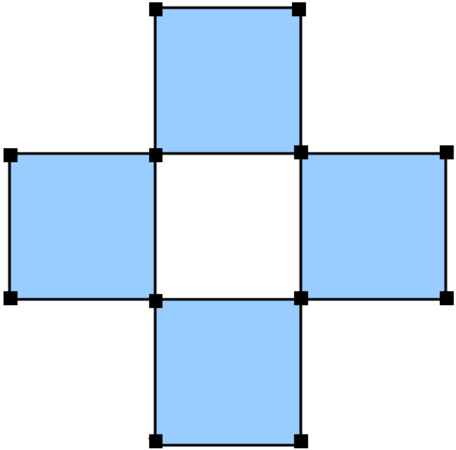
Simple graph example.



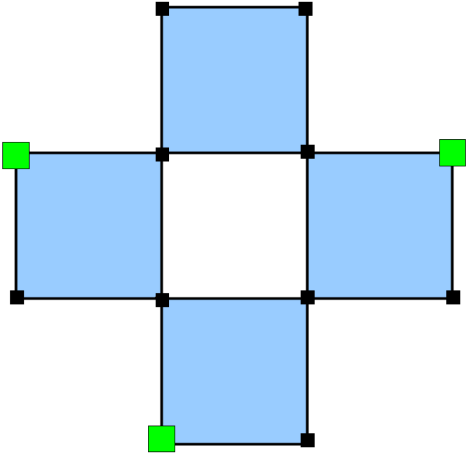
Simple graph example.



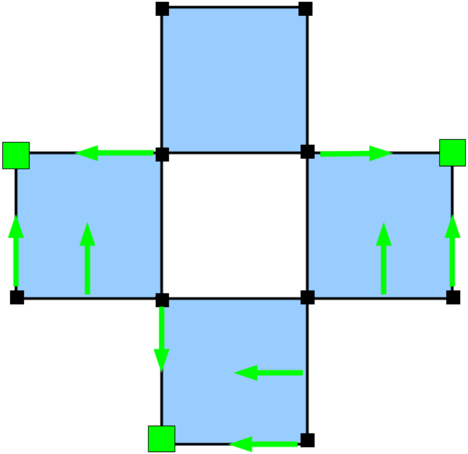
More complicated example.



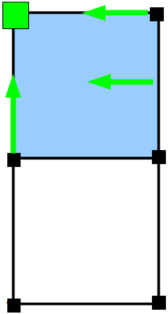
More complicated example.



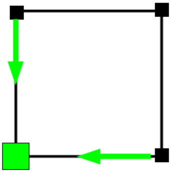
More complicated example.



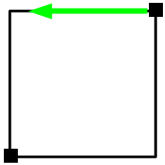
More complicated example.



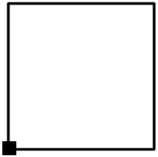
More complicated example.



More complicated example.



More complicated example.



Cone contraction algorithm.

- Pessimistic number of iteration – cubical.
- In flavor of distributed graph (MapReduce) algorithms (do not have shared memory).
- Easy to construct Flag / VR complex in distributed way.
- Experimental code for a single machine (whole complex still at RAM).
- Distributed implementation for a single machine and based on MPI in progress.
- MapReduce implementation in plans.

The end.

Thank you for your attention!

Contact info: Paweł Dłotko, Institute of Computer Science,
Jagiellonian University, Kraków, Poland.
mail : pawel.dlotko@ii.uj.edu.pl
www.ii.uj.edu.pl/ dlotko