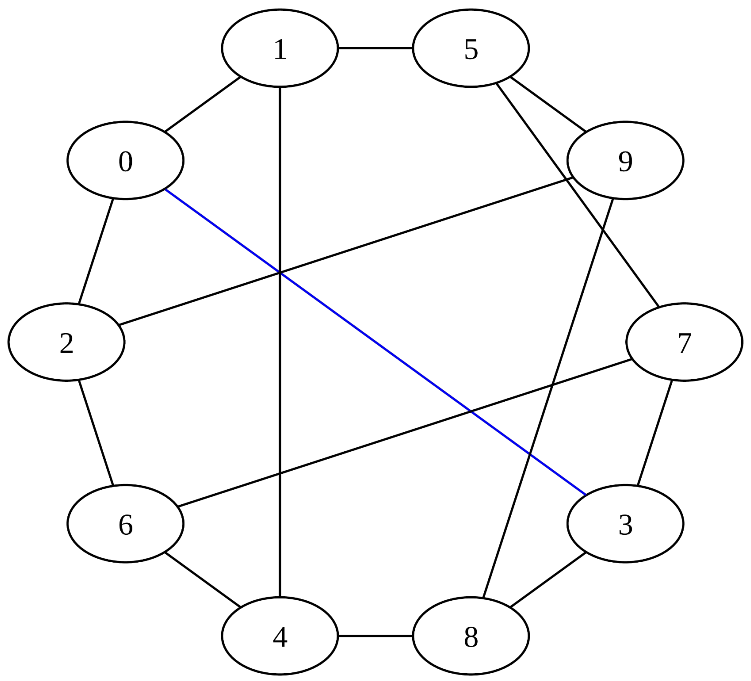
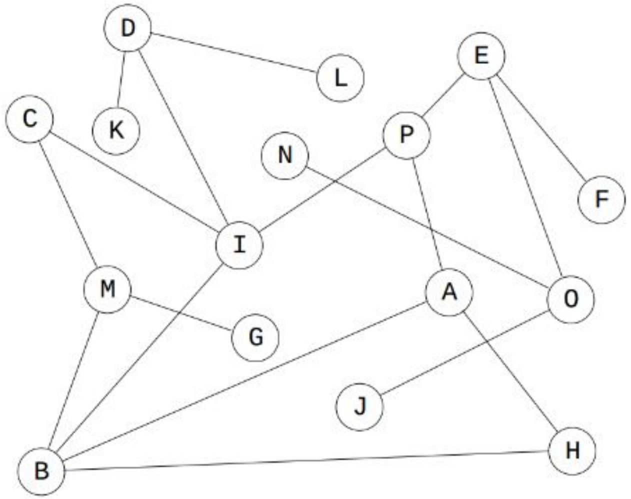
Practical Aspects of Recognising Outer *k*-Planar Graphs

Shevchenko Ivan

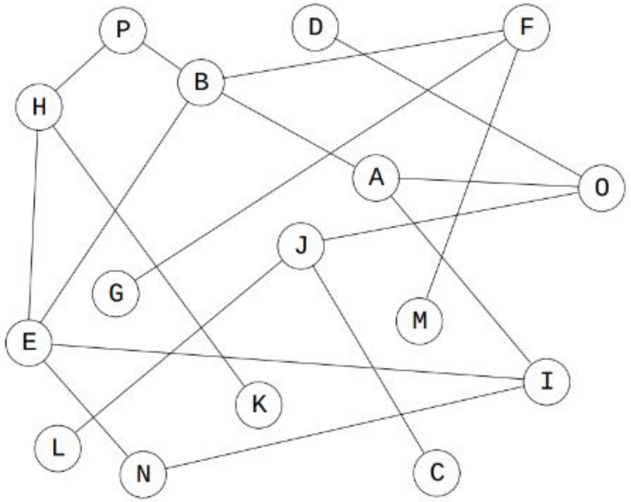
Supervisor: Prof. Dr. Alexander Wolff



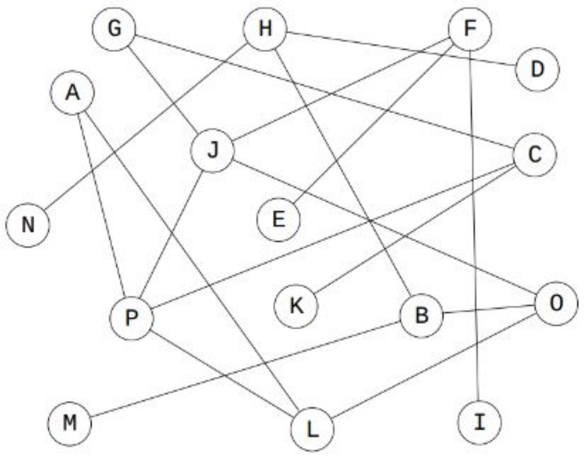
Intersections in graph drawings



scf



scs

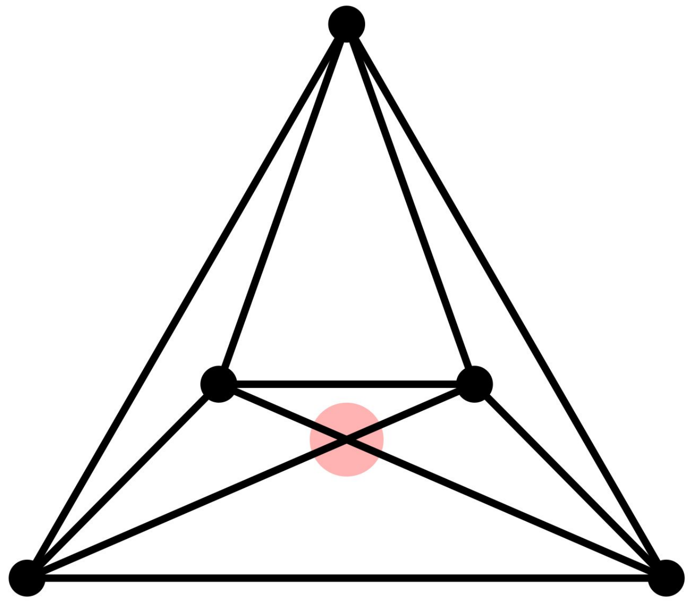


scm

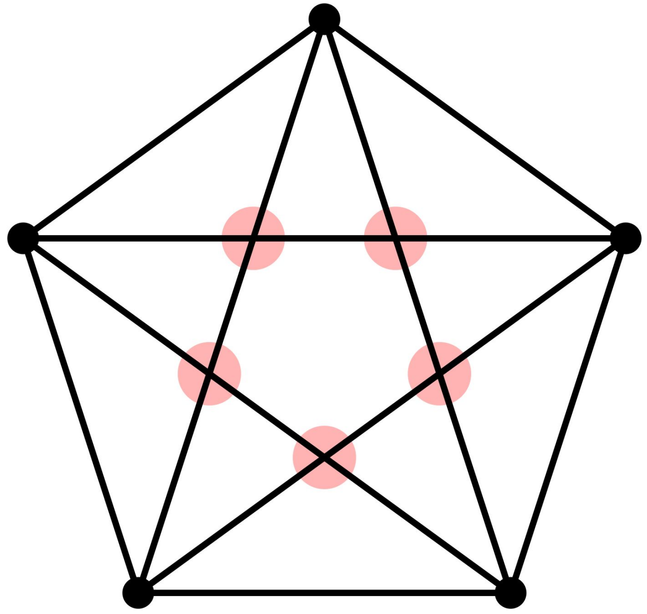
Drawings of sparse graphs with few some and many crossings*

*: Helen C. Purchase, Robert F. Cohen, and Murray I. James. “An experimental study of the basis for graph drawing algorithms”. In: ACM Journal of Experimental Algorithmics 2 (1997), pp. 4–es.

Beyond planarity



1-planar drawing

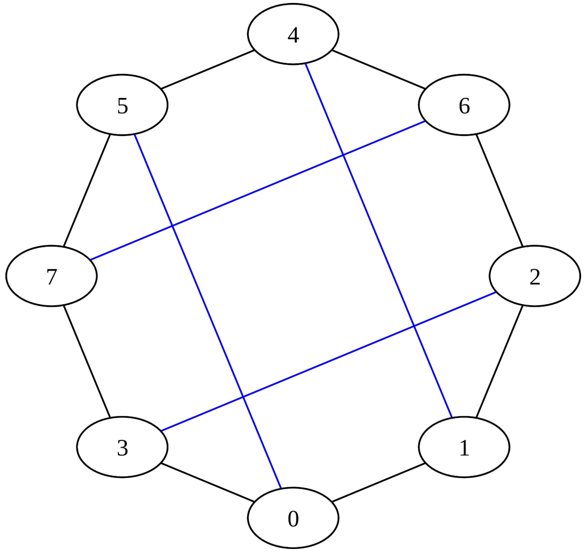


outer 2-planar drawing

Drawings of complete graph with 5 vertices

Problem formulation

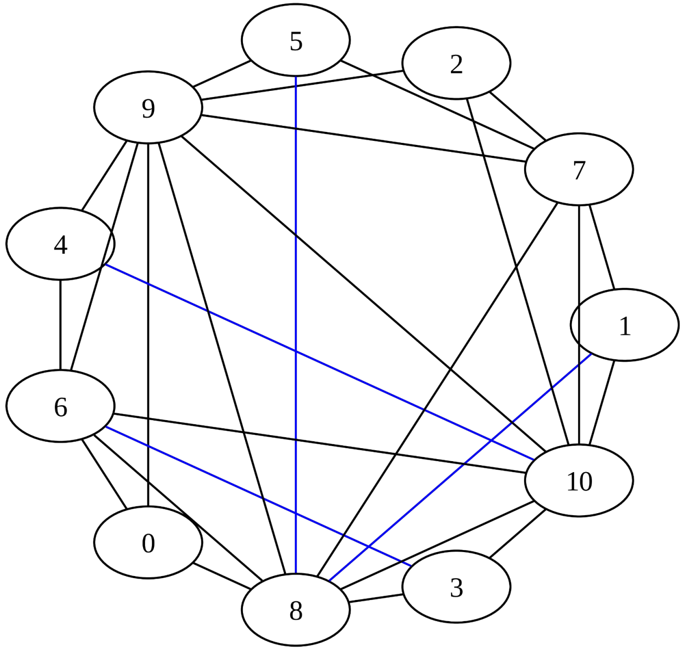
- Given
  - a graph **G**
- find the smallest value of *k* such that **G** admits a drawing where
  - the vertices are drawn as pairwise different points on a circle
  - the edges are drawn as straight lines
  - each edge is crossed at most k times



Outer 2-planar drawing of cubical graph

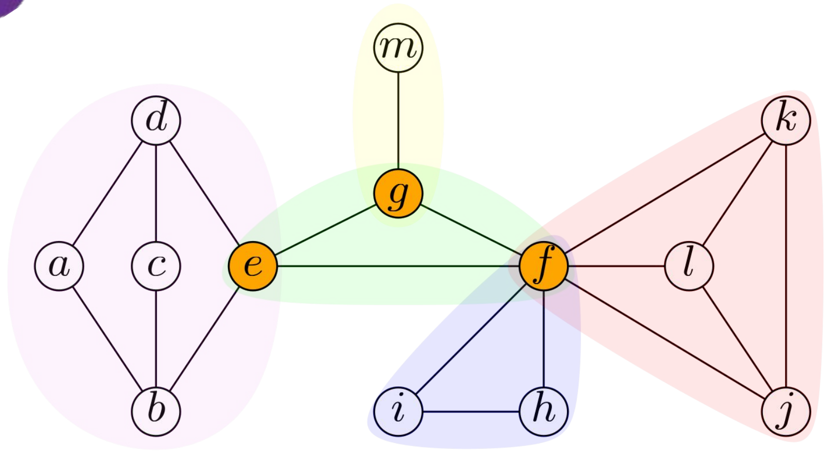
Our contribution

- Implementation of three recognition algorithms:
  - ILP-based
  - SAT-based
  - DP-based
- An interface to invoke implemented algorithms:
  - **Input:** graph in GraphVIZ format
  - **Output:** minimal *k*, outer *k*-planar drawing

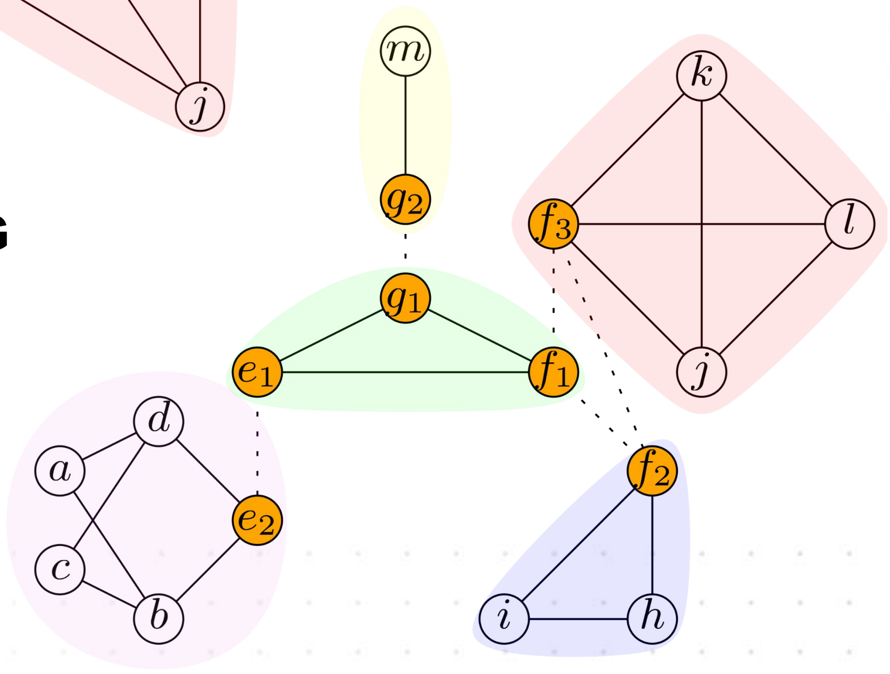


Returned drawing for Goldner-Harary graph

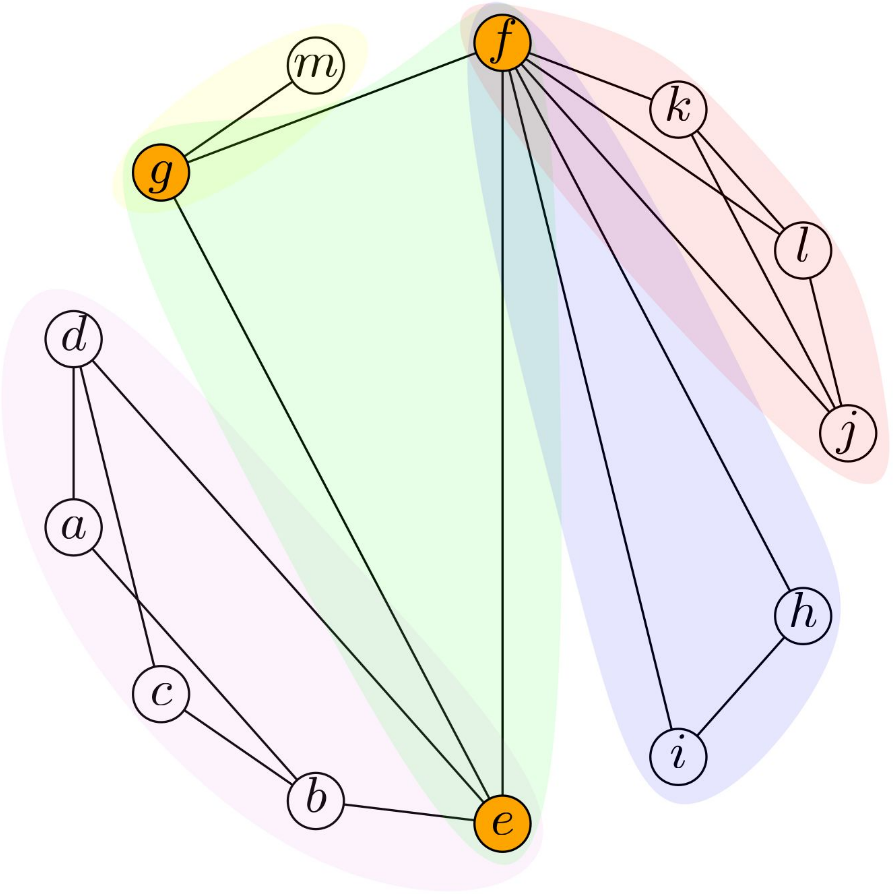
Biconnected decomposition



An original graph **G**



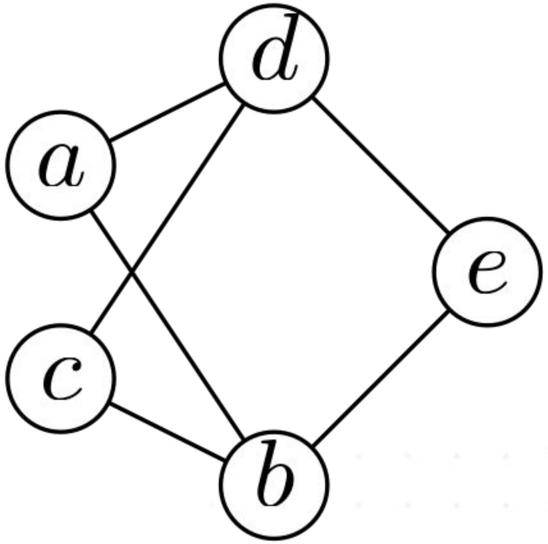
Outer k-planar drawing of each component



Outer k-planar drawing of **G**

Some details

- ILP GUROBI OPTIMIZATION
  https://www.gurobi.com
- SAT **Kissat**
  https://fmv.jku.at/kissat/



Graph drawing

$\Rightarrow$

a	d	e	b	c

Representation of the drawing

Encoding the problem

- Ordering variables:

  $\forall u, v \in V(G)$ the following holds:

$$a_{u,v} = 1 \quad \Leftrightarrow \quad \boxed{u} \cdots \boxed{v}$$

Ensuring transitivity:

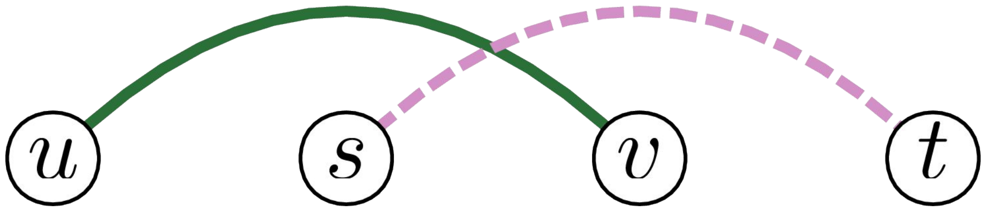
$$a_{u,v} = 1 \;\wedge\; a_{v,w} = 1 \quad \Rightarrow \quad a_{u,w} = 1$$
$$a_{u,v} = 0 \;\wedge\; a_{v,w} = 0 \quad \Rightarrow \quad a_{u,w} = 0$$

Encoding the problem

- Ordering variables: $a_{u,v} = 1 \iff$

- Crossing variables:

$\forall uv, st \in E(G)$ the following holds:

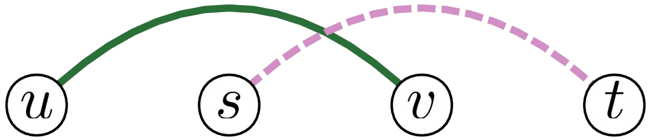


$\Rightarrow \quad c_{uv,st} = 1$

$$a_{u,s} = 1 \ \wedge \ a_{s,v} = 1 \ \wedge \ a_{v,t} = 1 \quad \Rightarrow \quad c_{uv,st} = 1$$

Encoding the problem

- Ordering variables: $a_{u,v} = 1 \Leftrightarrow$ 

- Crossing variables: 

$\Rightarrow \; c_{uv,st} = 1$

- **ILP**: **minimise** k
$$k \geq \sum_{e' \in E(G)} c_{e,e'}, \qquad \forall e \in E(G)$$

- **SAT**:
$$\forall f, e_1, e_2, \ldots, e_{k+1} \in E(G)$$
$$\neg(c_{f,e_1} \land c_{f,e_2} \land \cdots \land c_{f,e_{k+1}})$$

Crossing variables

- Idea #1:

  Take into account other non-crossing arrangements:

$$\Rightarrow \quad c_{uv,st} = 0$$

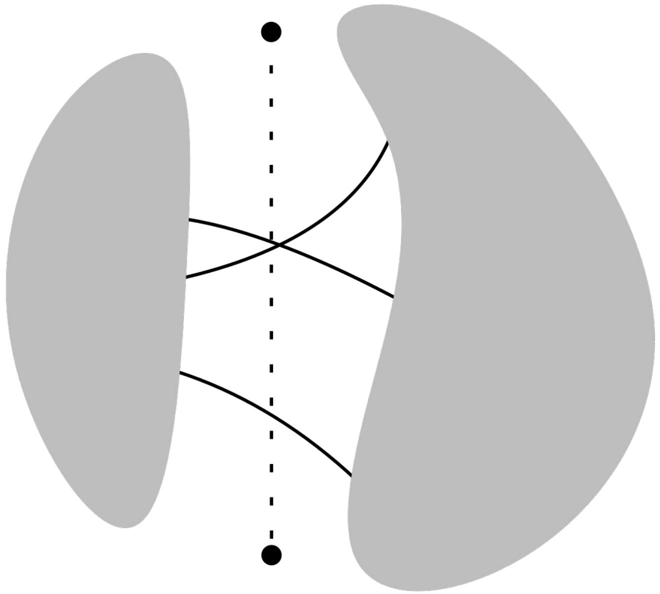
- Idea #2:

  Add an extra term to the ILP objective:

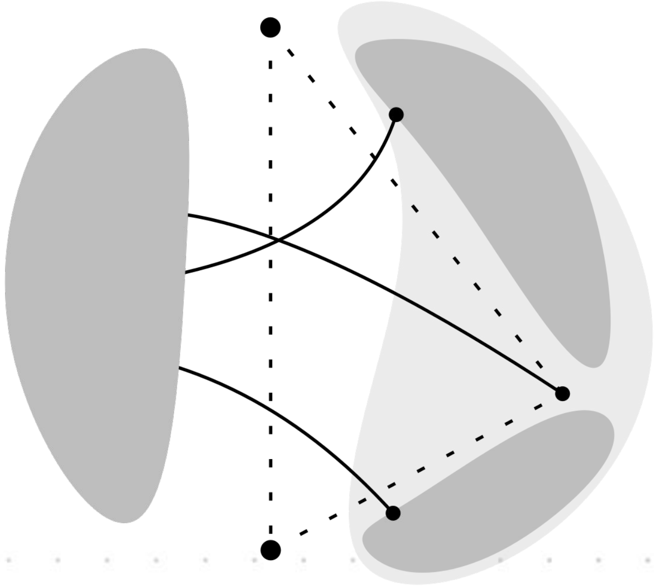
minimise $$k + \sum \frac{c_{e_1,e_2}}{|E|^2}$$

Main idea

- Split process into steps parameterised by:
  - active link
  - right side
  - set of edges crossing link
- Construct drawings using previously completed steps

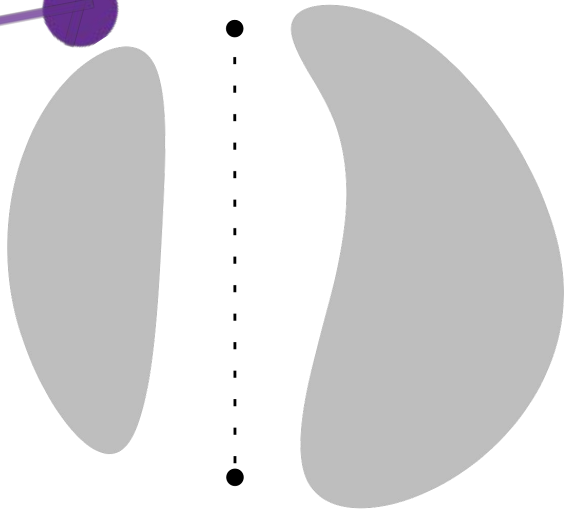


Step configuration

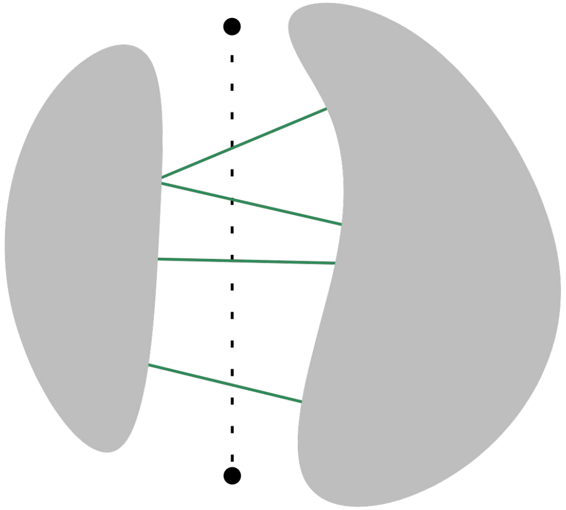


Drawing construction

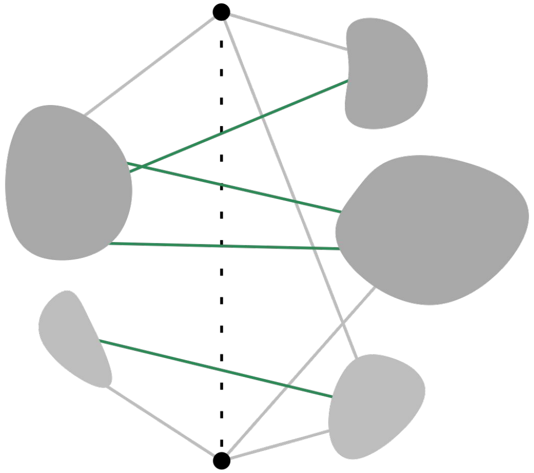
Index construction



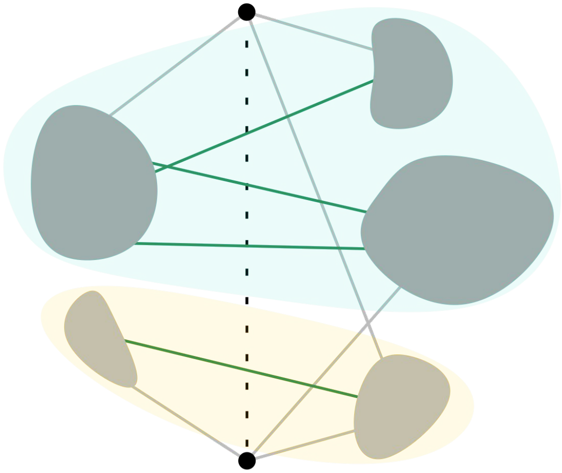
1. Select active link



2. Select number of edges and the edges

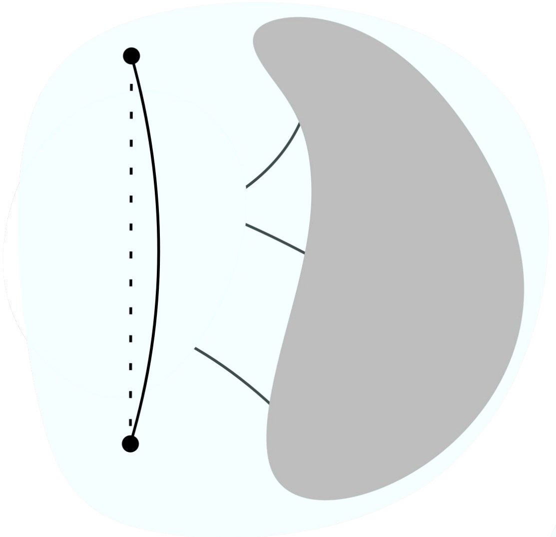


3. Split remaining graph into components

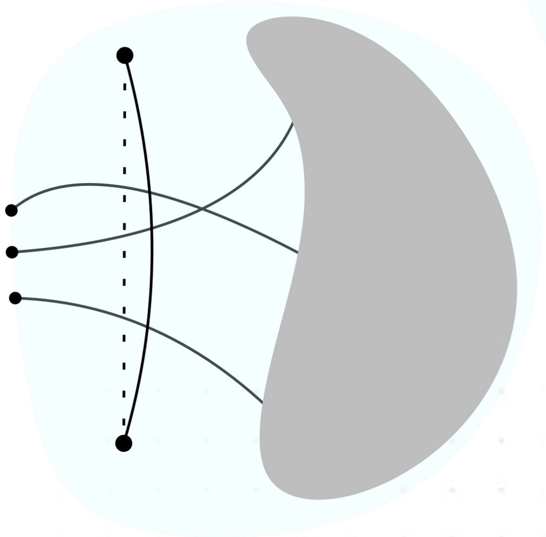


4. Construct all possible right sides from the combination of components

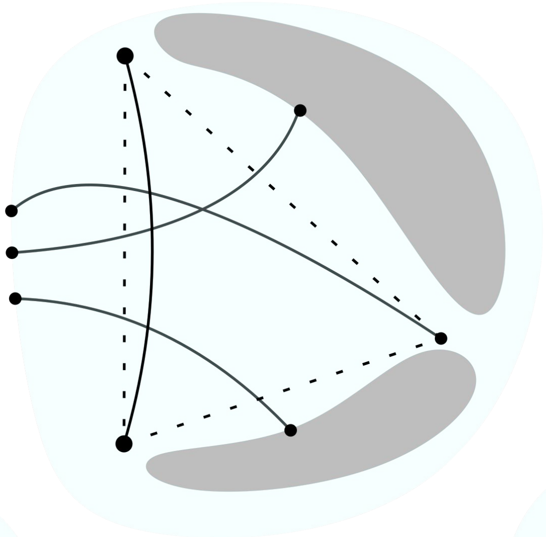
Process configurations



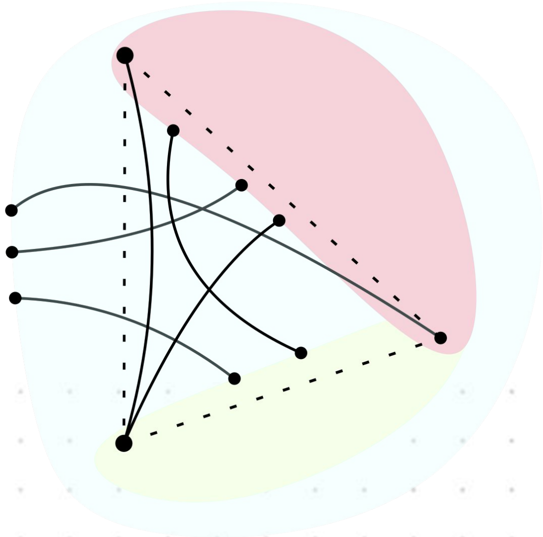
1. Step configuration



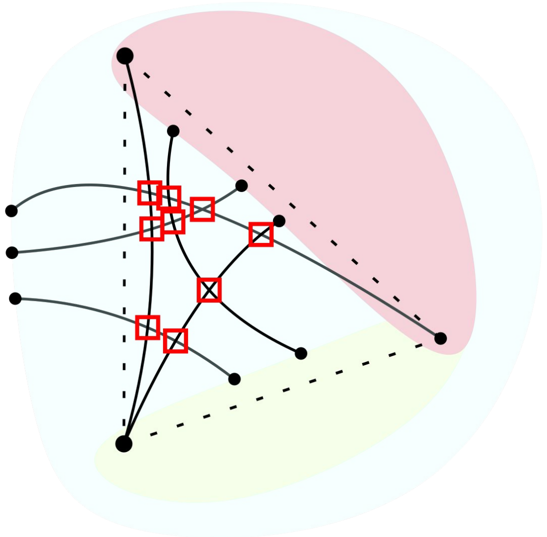
2. Select the order of piercing edges



3. Select a split vertex and partition the rest



4. Select the drawing for each partition

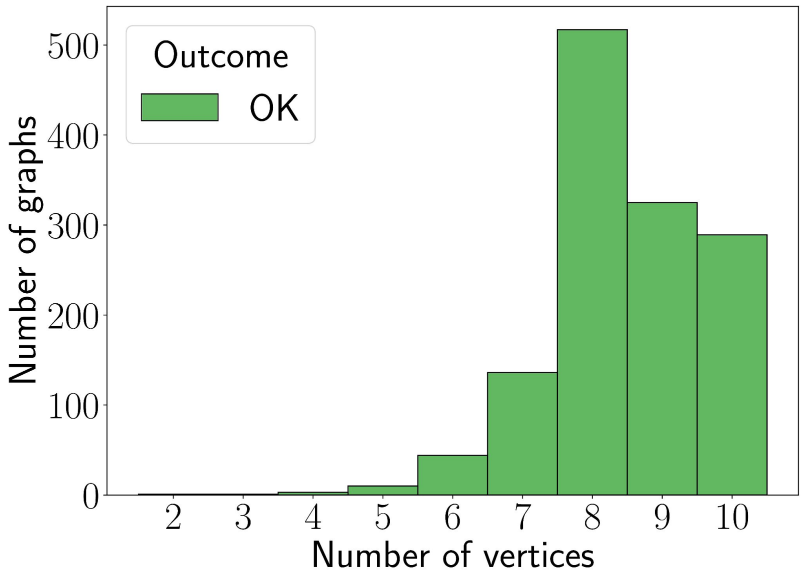


5. Check whether the drawing is valid

Sample graphs

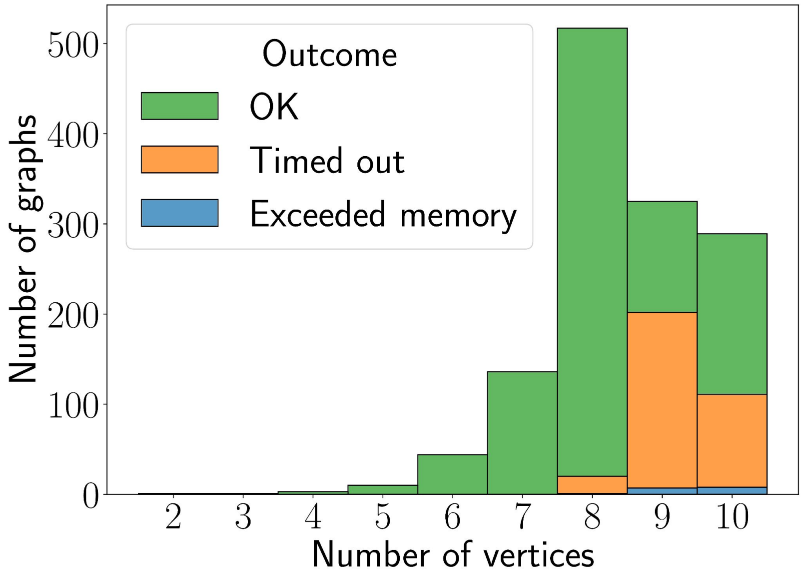
Queried from The House of Graphs:
- connected
- at most 10 vertices

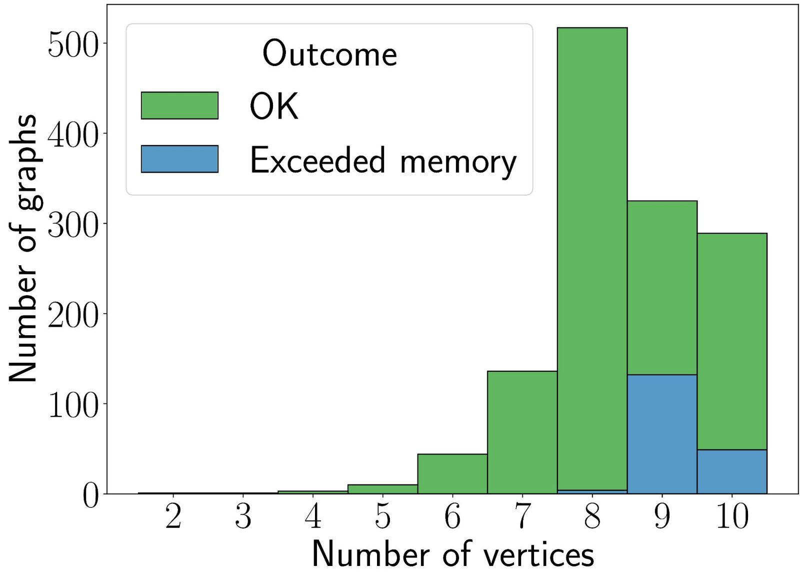
https://houseofgraphs.org/



ILP

DP

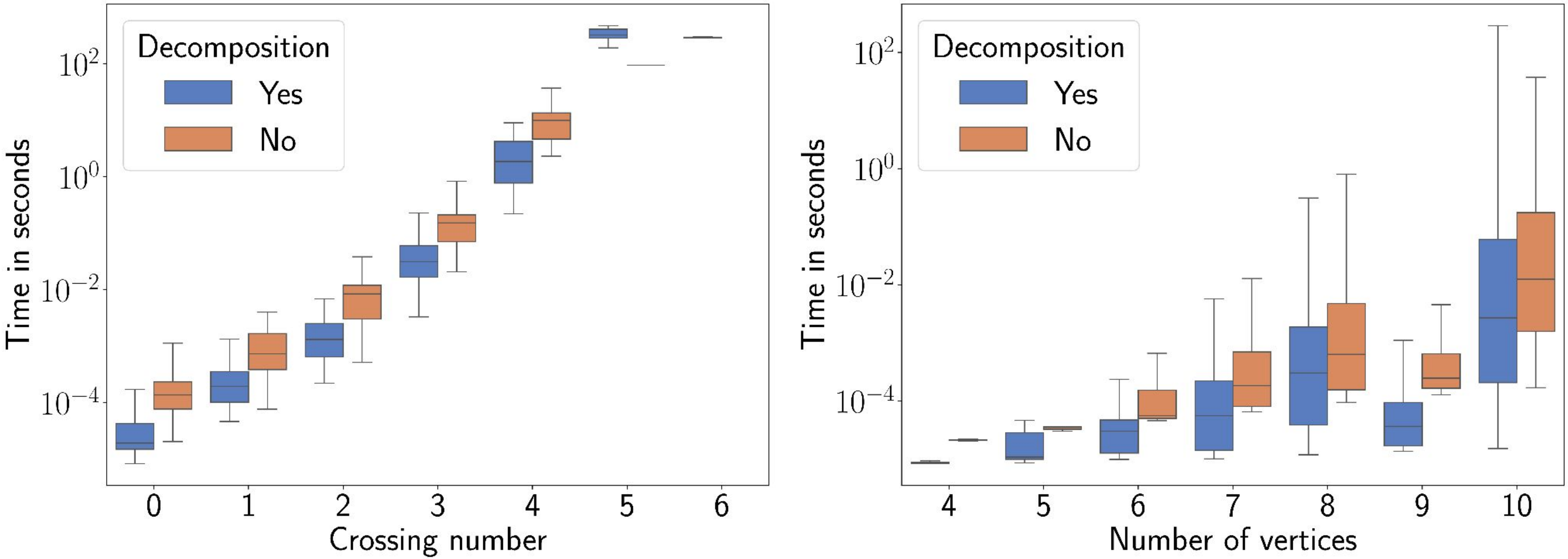




SAT

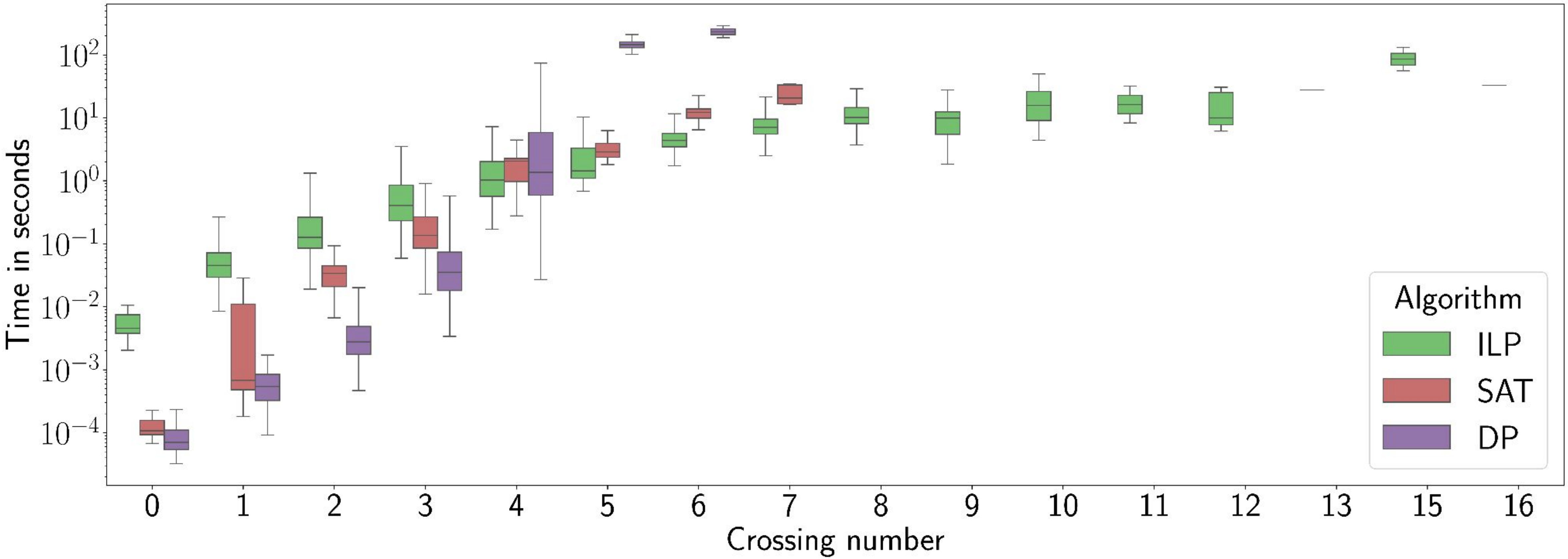
Biconnected decomposition

Results of the experiment for DP-based algorithm



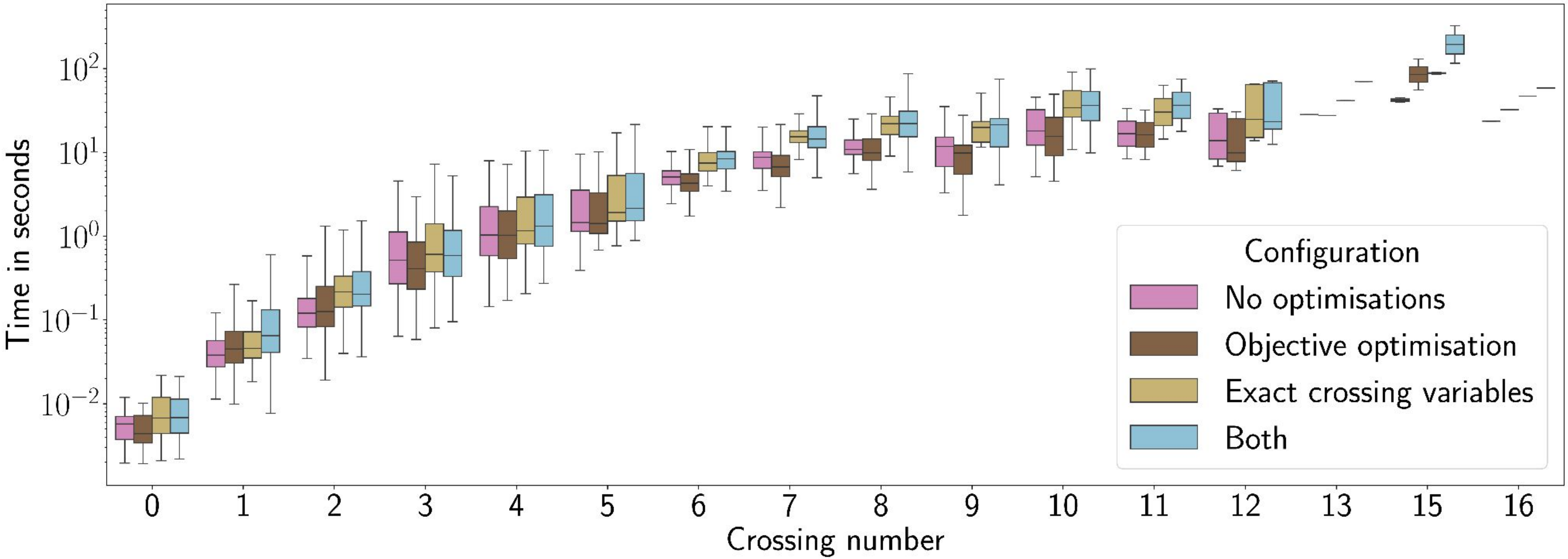
Algorithm comparison

Results of the comparison grouped by crossing number



Optimisations benchmark

Results of the experiments grouped by crossing number



Summary

In this thesis we

- introduced two methods:
  - ILP-based
  - SAT-based
- implemented three methods:
  - DP-based
  - ILP-based
  - SAT-based
- combined them in a single interface

We experimentally showed that

- biconnected decomposition improves performance
- requirement for resources grow slower for ILP-based method
- for DP- and SAT-based methods they grow exponentially
- objective optimisation improves runtime for ILP-based algorithm

Reviewer comments

- Figure 3.1(d) does not add much clarity beyond 3.1(c).
- Optimisations for ILP- and SAT-based algorithms feel somewhat ad-hoc.
- Providing a fully reproducible setup for the experiments as a Docker container or a Nix package would be great.

Q&A