

# On linear layouts with SAT

---

Mirco Haug

July 10, 2019

Eberhard Karls Universität Tübingen

# Contents

Motivation

Theory

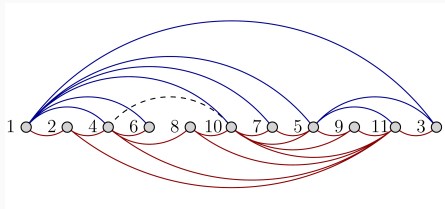
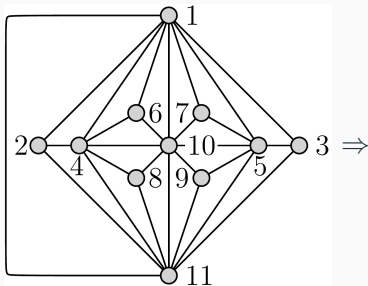
Implementation

Demo

# Motivation

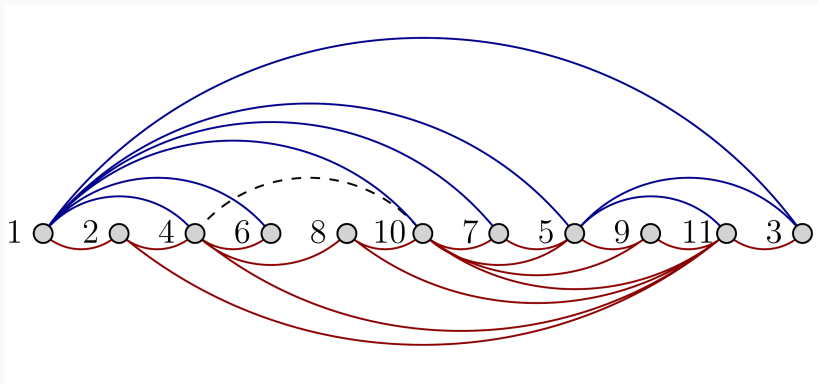
---

# What to do?



# The bookembedding problem

- No crossings
- Multiple pages / Colors
- Node order
- Edge assignments



# Approach

- Combinatoric explosion
- SAT solvers
- <http://be.cs.arizona.edu>
- Rather static
- Flexibility needed
- Old proof sketch of Yannakakis[?]

# Theory

---



## Standart book embedding[?]

- Node order  $\sigma(n_1, n_2)$
- Edge assignment  $\phi_{P1}(e_1)$

# Constraints

- EDGES\_ON\_PAGES
- EDGES\_SAME\_PAGES
- EDGES\_DIFFERENT\_PAGES
- EDGES\_TO\_SUB\_ARC\_ON\_PAGES
- EDGES\_FROM\_NODES\_ON\_PAGES
- NODES\_PREDECESSOR
- NODES\_REQUIRE\_ABSOLUTE\_ORDER
- NODES\_REQUIRE\_PARTIAL\_ORDER
- NODES\_FORBID\_PARTIAL\_ORDER
- NODES\_CONSECUTIVE

# Constraints

- EDGES\_ON\_PAGES
- EDGES\_SAME\_PAGES
- EDGES\_DIFFERENT\_PAGES
- EDGES\_TO\_SUB\_ARC\_ON\_PAGES
- EDGES\_FROM\_NODES\_ON\_PAGES
- NODES\_PREDECESSOR
- NODES\_REQUIRE\_ABSOLUTE\_ORDER
- NODES\_REQUIRE\_PARTIAL\_ORDER
- NODES\_FORBID\_PARTIAL\_ORDER
- NODES\_CONSECUTIVE

# Implementation

---

# Main classes

- `app.py`
- `model.py`
- `solver.py`

## Linear layout API <sup>1.0</sup>

[ Base URL: / ]

<http://127.0.0.1:5000/swagger.json>

Through this API one can request for a linear layout of a graph in graphml format.

The actual computation of the linear layout is done using SAT solving. The instances are solved using [lingeling](#)

### default Default namespace



**POST** /embeddings Create a new embedding

**GET** /embeddings List all embeddings

**GET** /embeddings/{id} Get an embedding by id

### Models



- Schema definition
- Deserialize
- Validate

- Clause generation
- DIMACS Generation
- Parser for lingeling result



- Glue
- Calls lingeling
- Service interface

# Demo

---

`http://algo.inf.uni-tuebingen.de/linearlayouts/index.  
html#or174`

