Algorithm Engineering – Exercise 5
or *"What could have gone wrong?"*

Team 2: Denis Koshelev, Julian Fechner, Julio Cesar Perez Duran

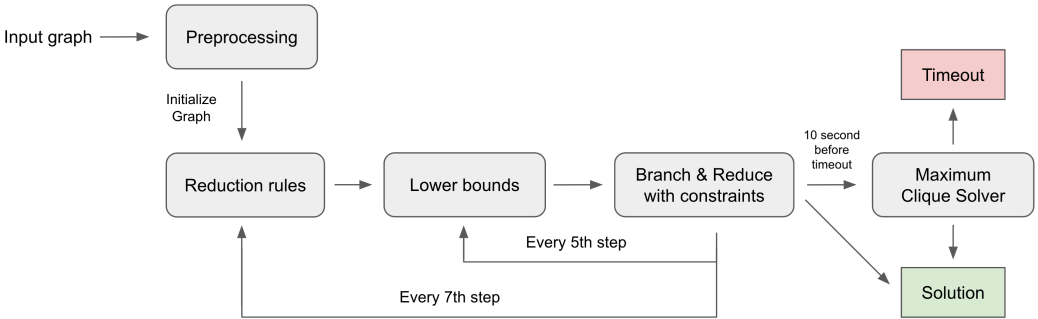
Presentation, February 15

Overview

In this presentation we will talk about:

1. Final solver architecture
2. Implemented features
   - Branch and reduce algorithm
   - Maximum Clique solver
   - Reduction rules
3. Autoconfiguration tool
4. Bottlenecks and optimization
5. Final comparison
6. Statistics

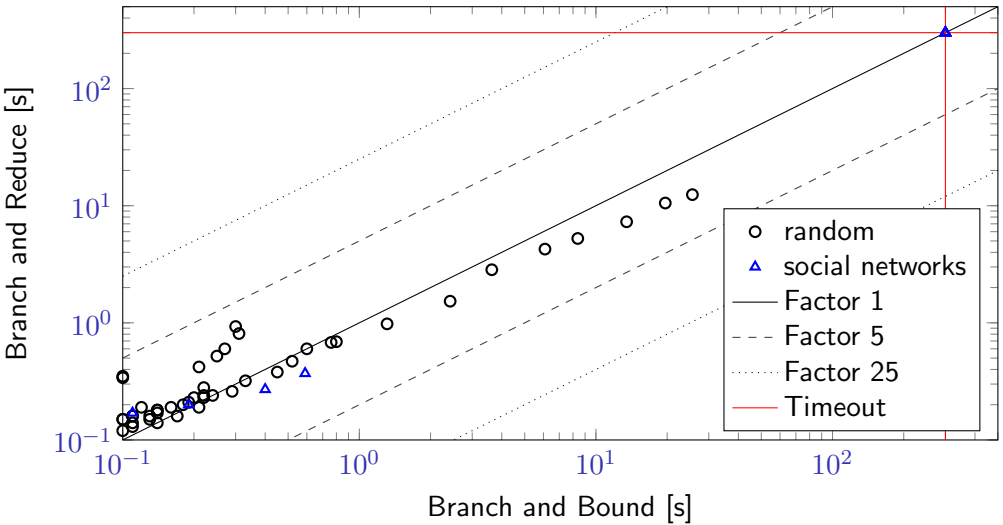
Final solver architecture



Implemented features

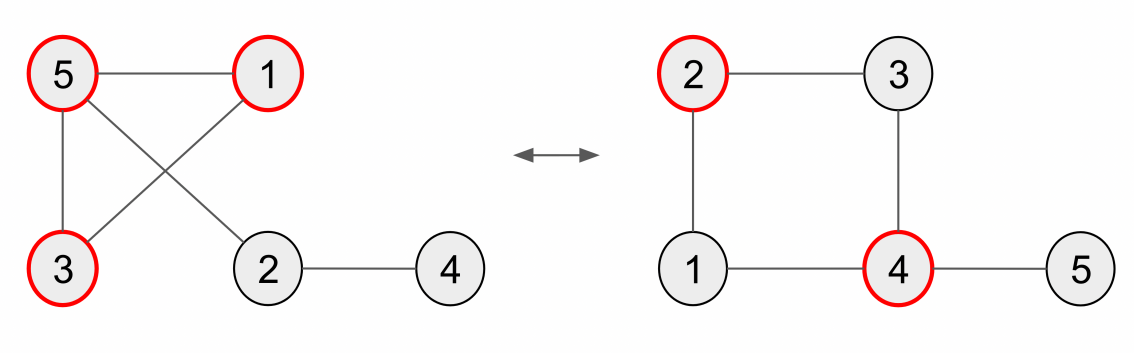
- Branch and reduce algorithm with two constraints
- Maximum Clique Solver intergration
- Reduction rules:
  - LP Reduction via Maximum Flow
  - Degree-Three-Independent-Set Rule
  - Twin Rule

Comparison of raw branching algorithms



Maximum Clique Reduction

Maximum Clique $\leq_p$ Minimum Vertex Cover



Maximum Clique in G

Minimum Vertex Cover in $\bar{G}$

MoMC Algorithm

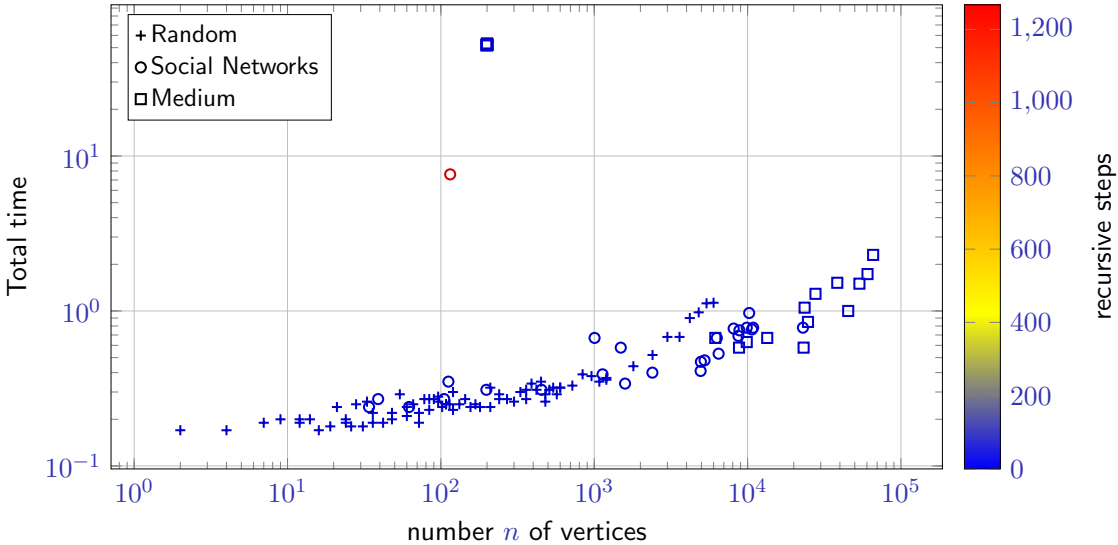
For solving Maximum Clique Problem we used **MoMC algorithm** [Li, Jiang, Manyà; 2017] inspired by the winner of the PACE Challenge for Vertex Cover. It combines:

- Branch-And-Bound Algorithm
- MaxSAT reasoning
- Dynamic and static strategies of reducing

Highlights:

- It works really well on `medium-sized` instance but not on the `social-networks`.
- It wasn't included in the final submission.

Solver with intergrated MoMC



Reduction Rules

We have implemented the following reduction rules:

- LP Reduction via Flow
- Twin Rule
- Independent Rule

LP Reduction via Flow

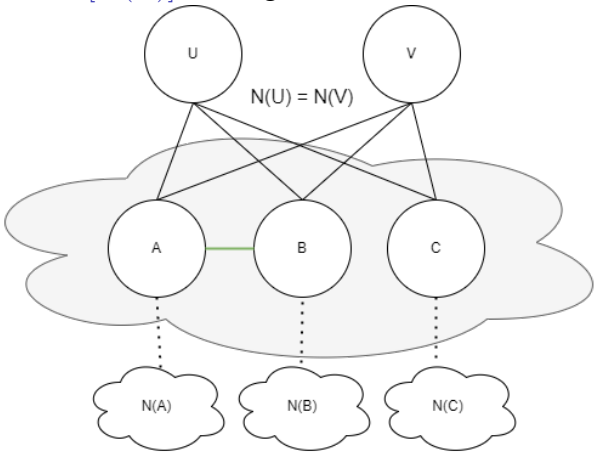
We implemented not optimized version of the flow reduction using **Tarjan's Algorithm** for finding strongly connected components.

Data structured used both for `FlowGraph` and `ResidualGraph`:

- Adjacency map with vertices and corresponding edges
- Hash set of all edges

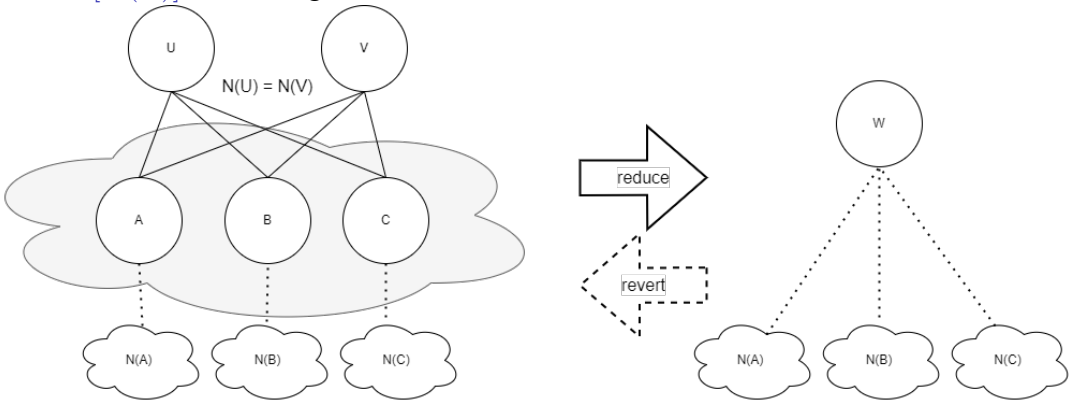
Twin Rule [Hespe et al.; 2020]

Let $u, v \in V$ where $N(u) = N(v)$ and $|N(u)| = 3$.
Case 1: $G[N(U)]$ has edges



Twin Rule [Hespe et al.; 2020]

Let $u, v \in V$ where $N(u) = N(v)$ and $|N(u)| = 3$.
Case 2: $G[N(U)]$ has no edges



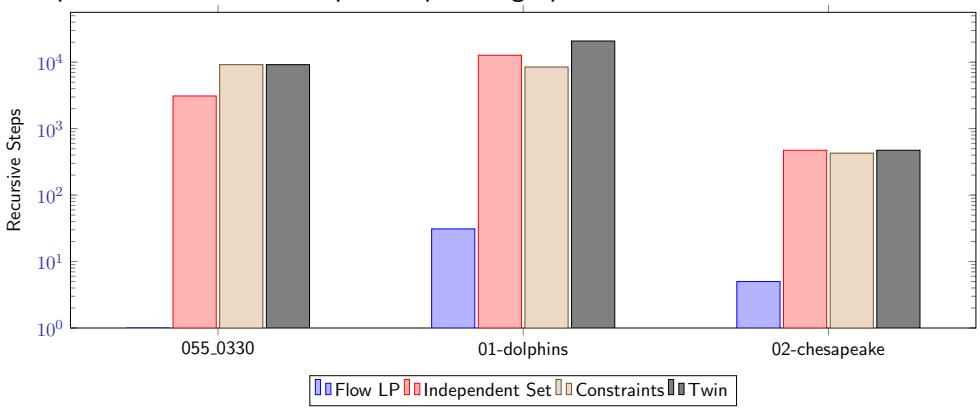
Independent Rule

As described in the lecture:

Let $v \in V$ with $N(v) = \{a, b, c\}$ and $N(v)$ is an Independent Set.
We can remove v and add following edges:
$\{\{a, b\}, \{b, c\}\} \cup \{\{a, x\} | x \in N(b)\} \cup \{\{b, x\} | x \in N(c)\} \cup \{\{c, x\} | x \in N(a)\}$

Reduction Rules

Comparison of recursive steps for specific graph instances.



Autoconfiguration tool

SMAC = **S**equential **M**odel **A**lgorithm **C**onfiguration

Problems and Challenges:

- How to choose useful timeouts?
- How to choose training dataset?
- How many parameters to optimize?
- How can we compare results from multiple testruns?

Autoconfiguration tool

First Experiment:

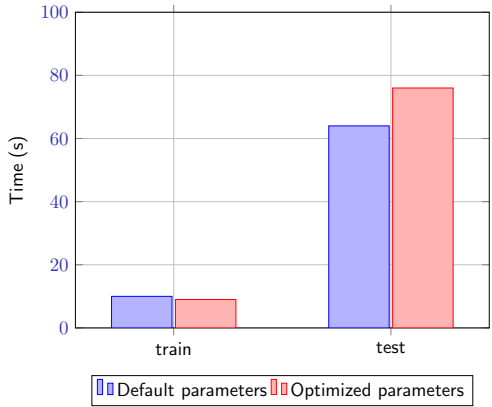
- cutoff-time = 300sec
- max-runtime = 24h
- custom training dataset
- all possible parameters

Second Experiment:

- cutoff-time = 150sec
- max-runtime = 24h
- custom training dataset
- depth parameters only

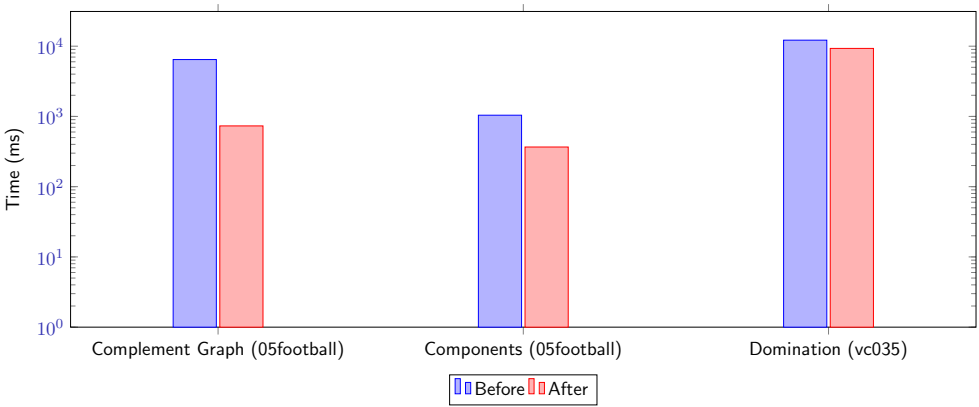
Autoconfiguration tool

And how did the optimized parameters perform?

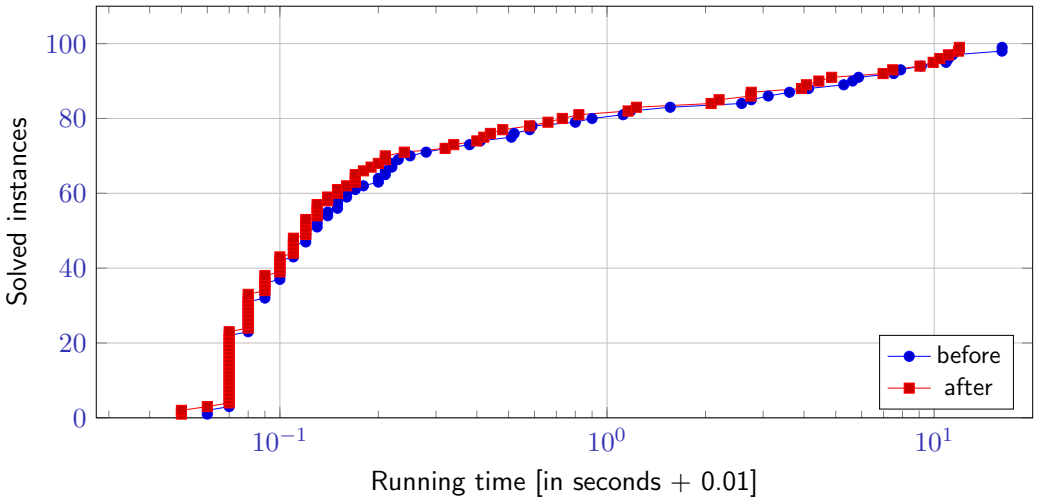


Bottlenecks and optimization

- Complement graph for Clique Bound
- Graph constructor for components
- Domination Rule



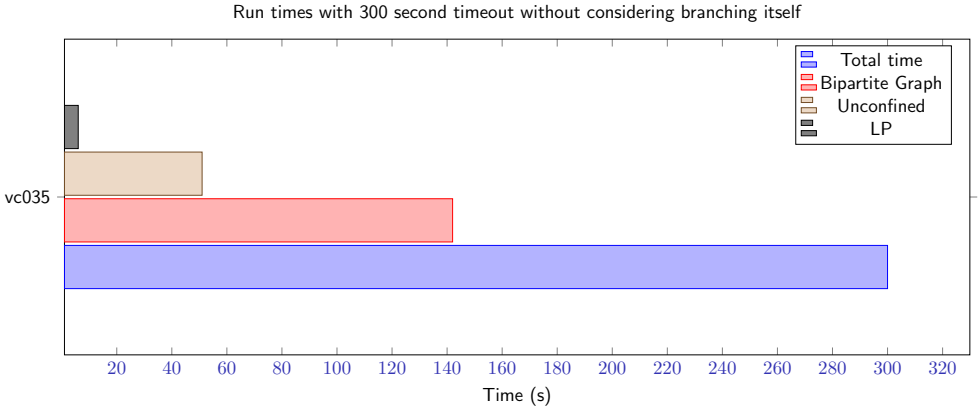
Bottlenecks and optimization



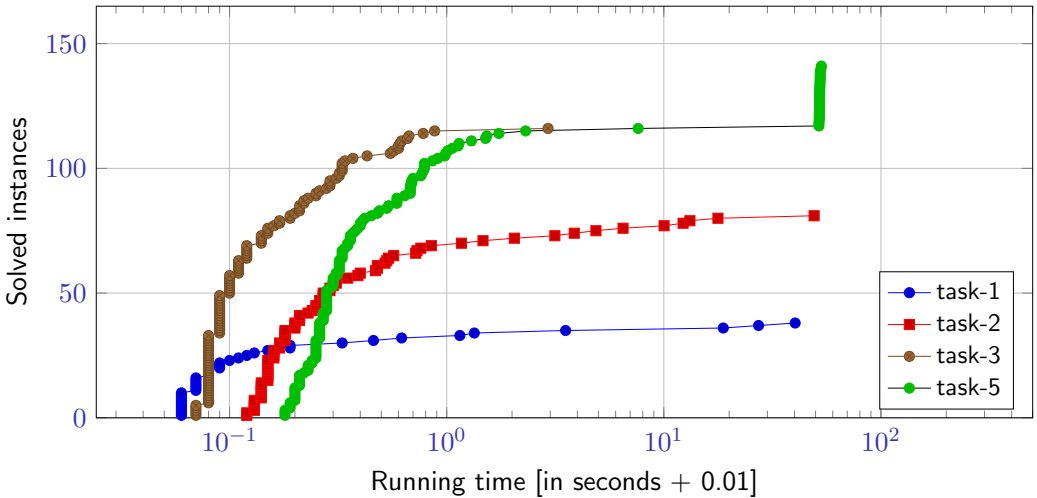
Remaining Bottlenecks

We discovered following bottleneck in our current implementation using *VisualVM* Profiler:

1. Updating bipartite graph for LP Bound
2. Unconfined Rule
3. Old LP Reduction



Comparison of all solvers



Statistics

During this course we:

- Made 536 commits
- Created 33 branches
- Run pipeline 226 times
- Had 12 sleepless nights (3 submissions * 4 exercises)
- Had a lot of joy engineering algorithms

Thank you for your attention!
Questions or Feedback?