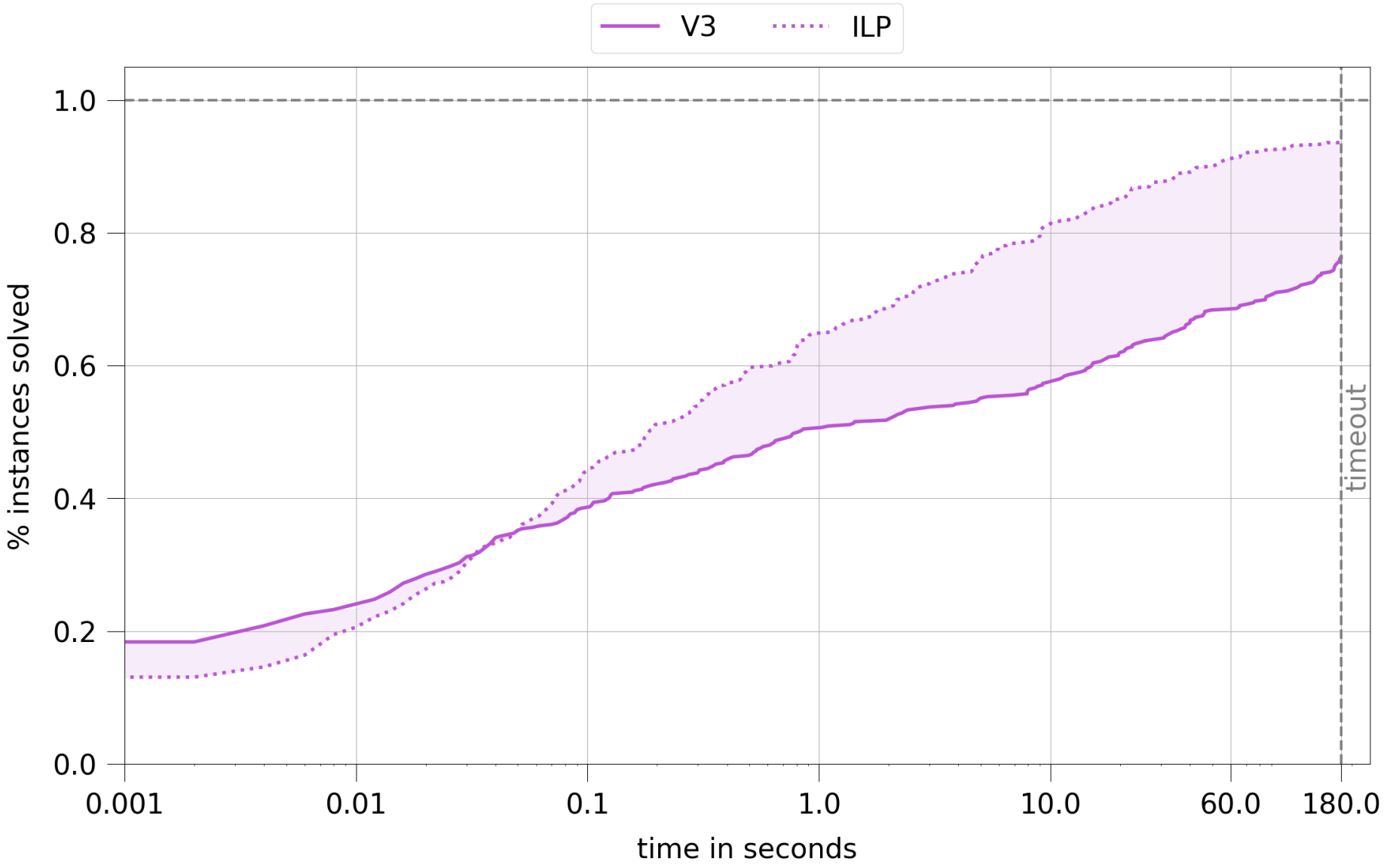
Algorithm Engineering: Milestone 4

Henri Dickel, Matija Miskovic
& Lennart Uhrmacher

Introduction

- We tried out the two ILP-solver approaches:
  - Lazy Cycles
  - Ordering
- Custom constraints:
  - Cycle Packing
  - Additional Cycles
- We used Gurobi for Java
- All plots were created using the newest dataset

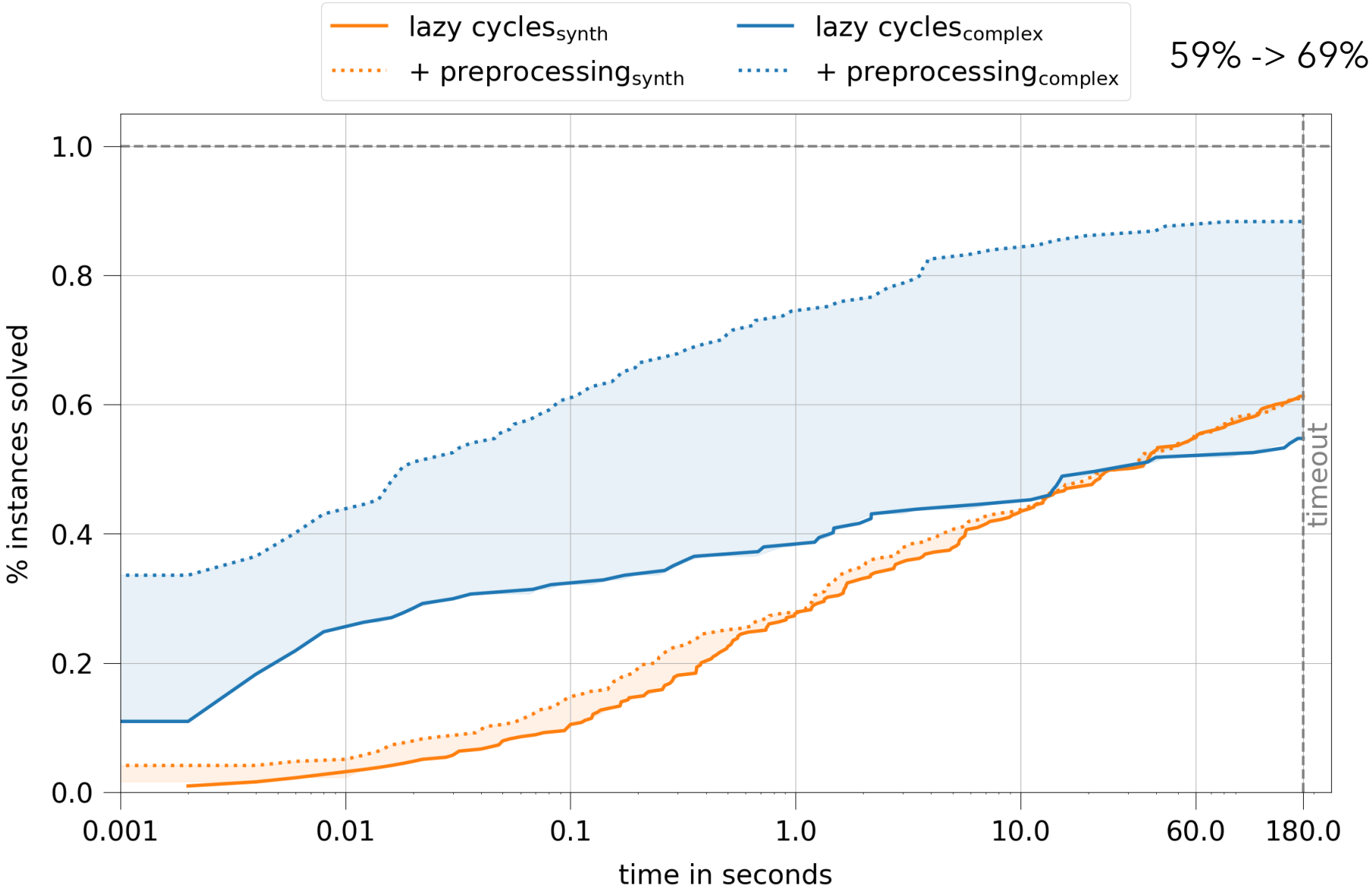
Introduction



Lazy Cycles

- Start with constraint for one shortest cycle (found by BFS)
- In callback, check if the solution graph is a DAG
  - If yes ➔ optimal solution found
  - If no ➔ add another shortest cycle from the remaining graph
- Tried out with and without preprocessing

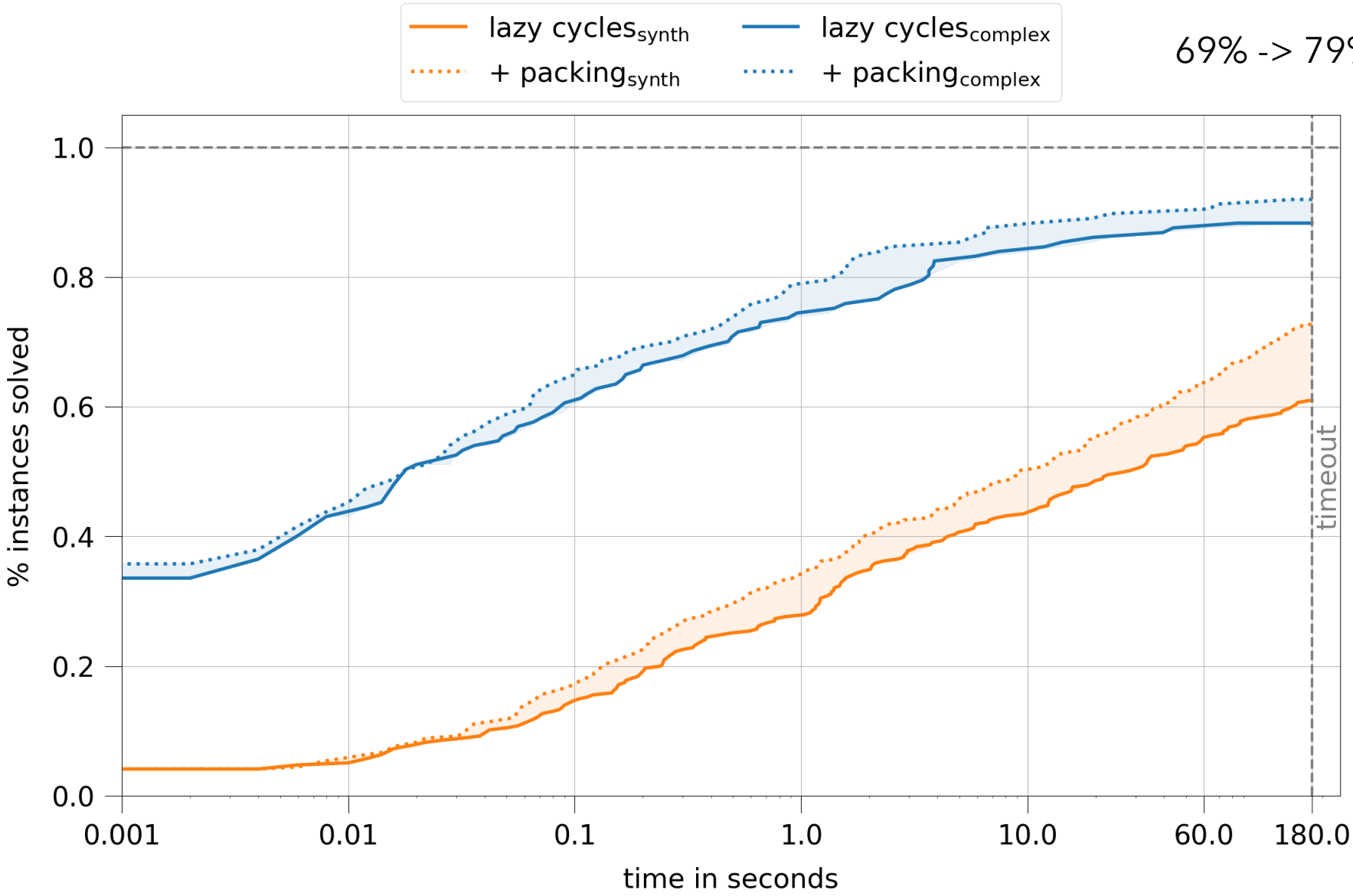
Lazy Cycles


lazy cycles$_{synth}$
+ preprocessing$_{synth}$
lazy cycles$_{complex}$
+ preprocessing$_{complex}$
59% -> 69%
% instances solved
1.0
0.8
0.6
0.4
0.2
0.0
timeout
0.001
0.01
0.1
1.0
10.0
60.0
180.0
time in seconds

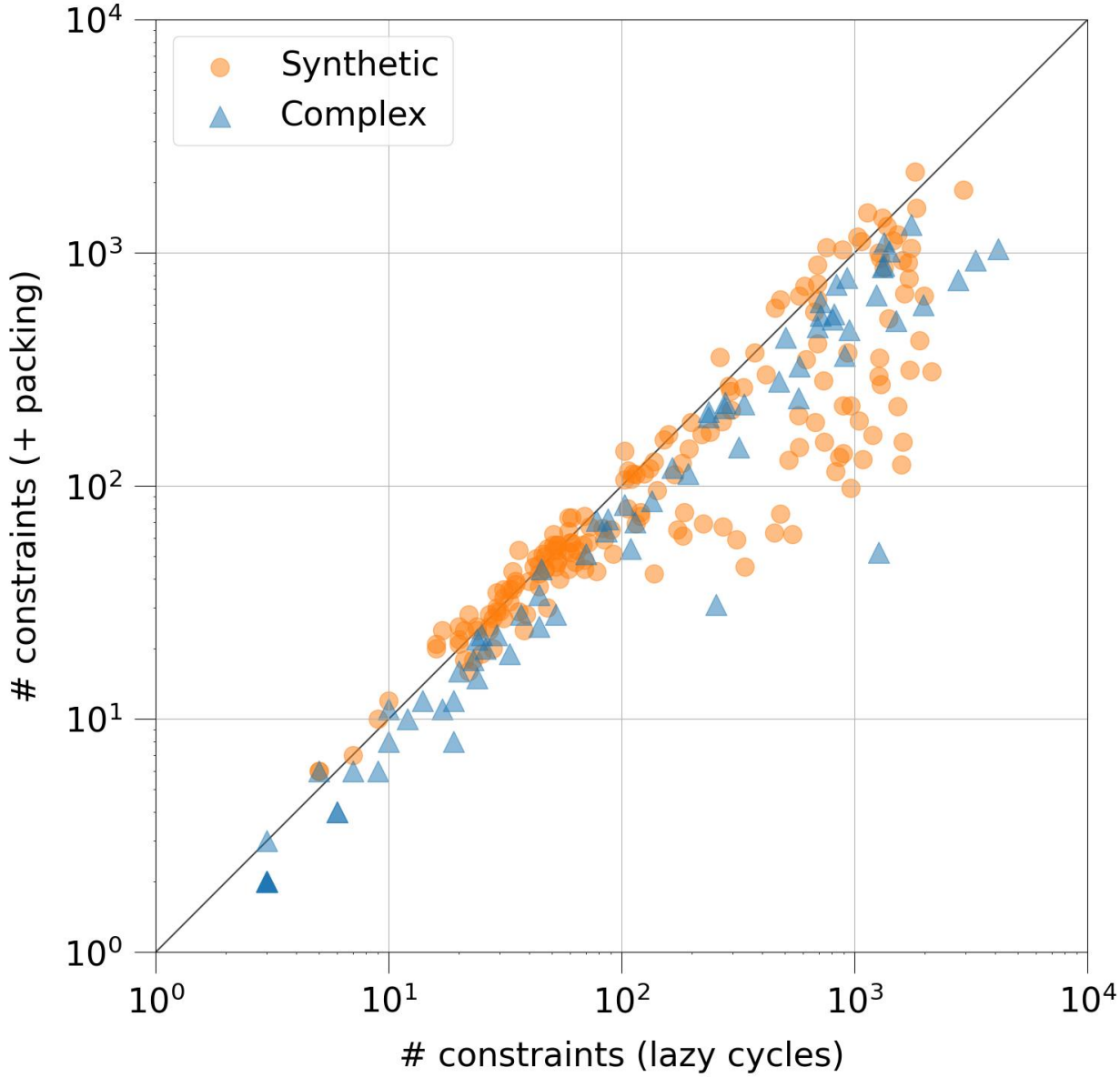
Cycle Packing Constraints

- Idea: initially use the Cycle Packing algorithm to get a set of distinct cycles
  - For each cycle, add a constraint to the ILP
  - With a good packing, only a small amount of additional callback constraints is needed
  - Benefit: the constraints don't share any variables

Cycle Packing Constraints



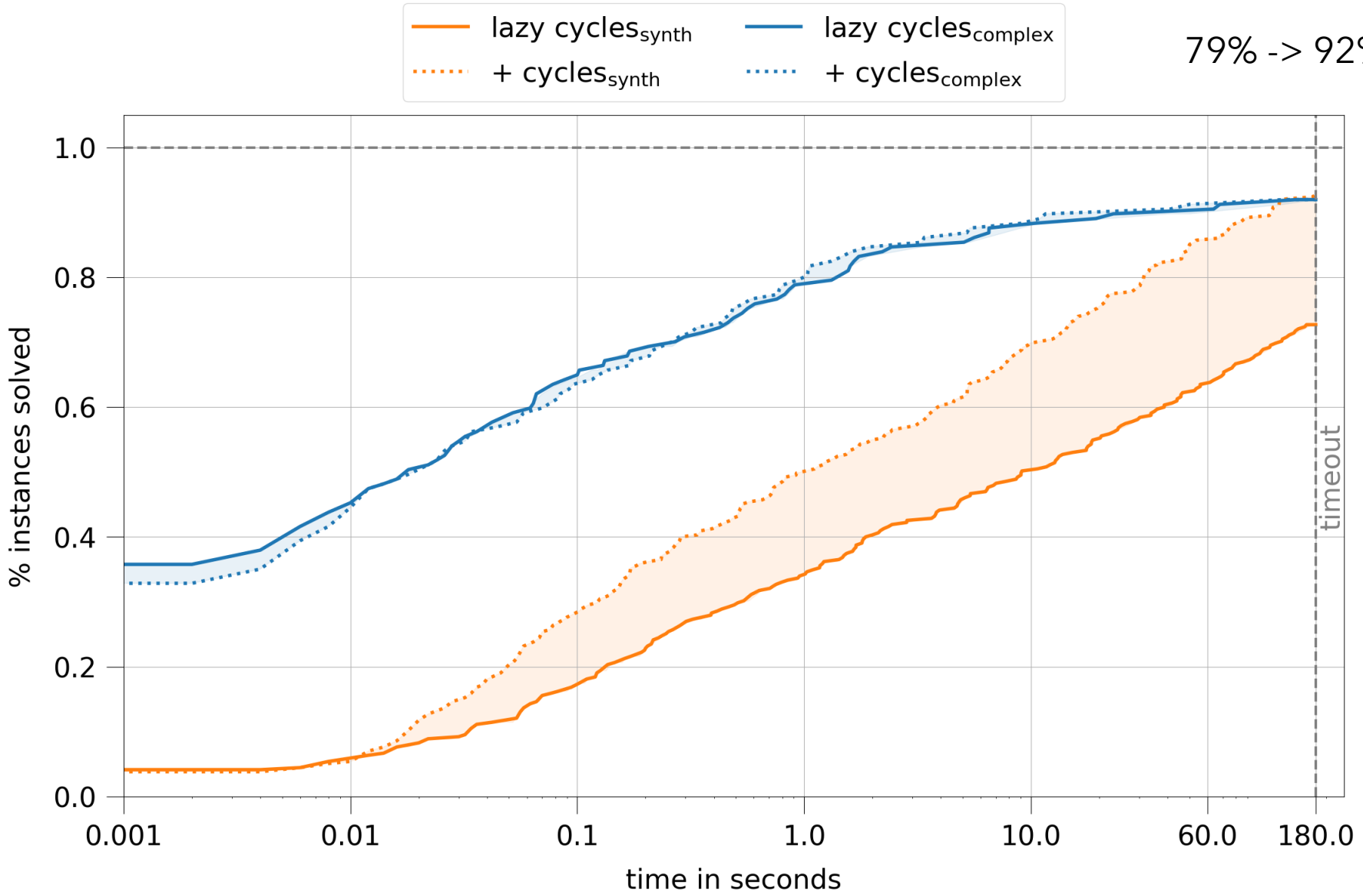
Cycle Packing Constraints



Additional Cycle Constraints

- Idea: Initially find the shortest cycle for each node with BFS and add a constraint
- N additional, but small constraints

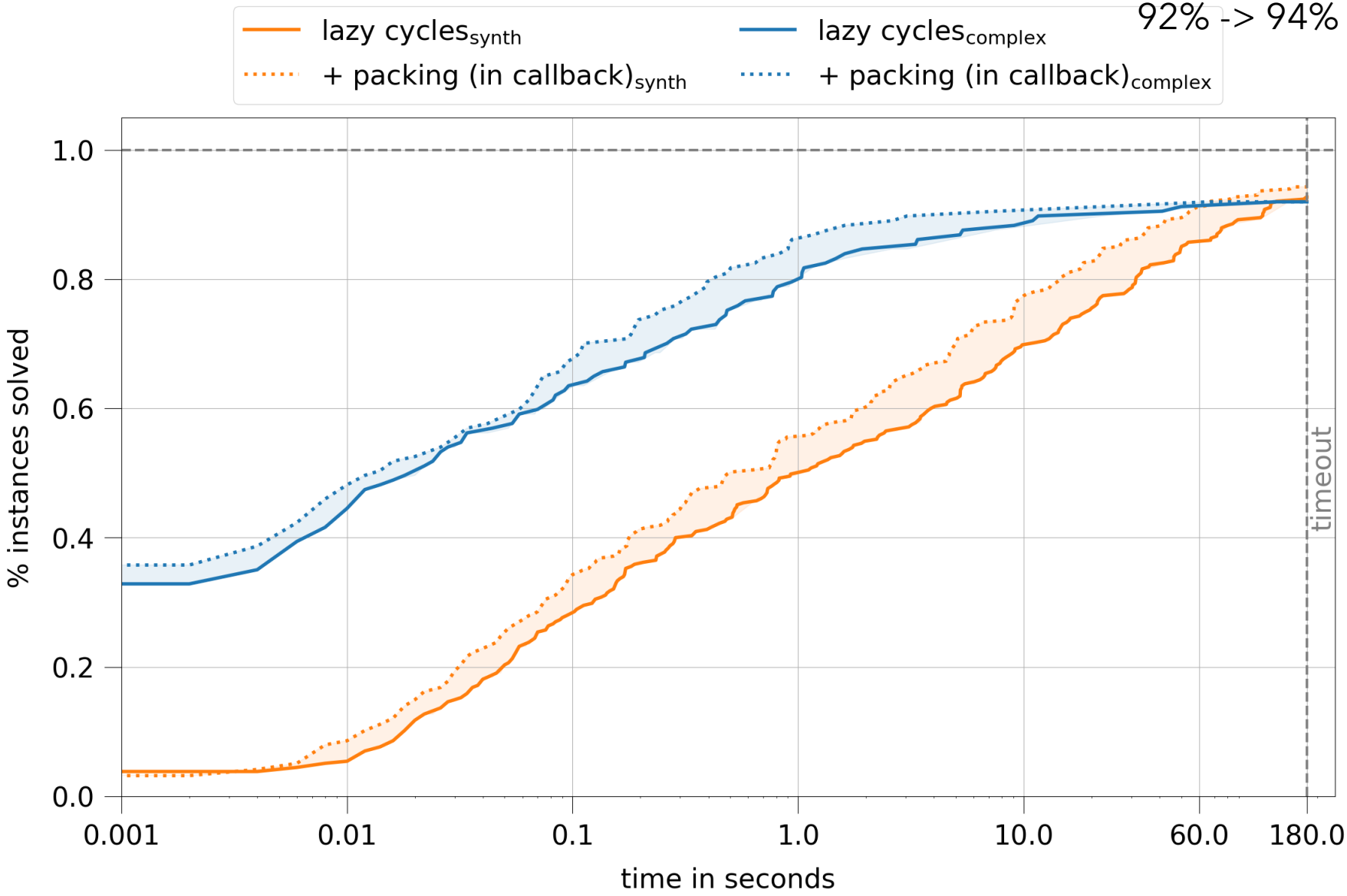
Additional Cycle Constraints



Cycle Packing in Callback

- Currently, in the callback we add a constraint for a single cycle of the remaining graph
- Instead, we could create a cycle packing on the remaining graph and add a constraint for each cycle

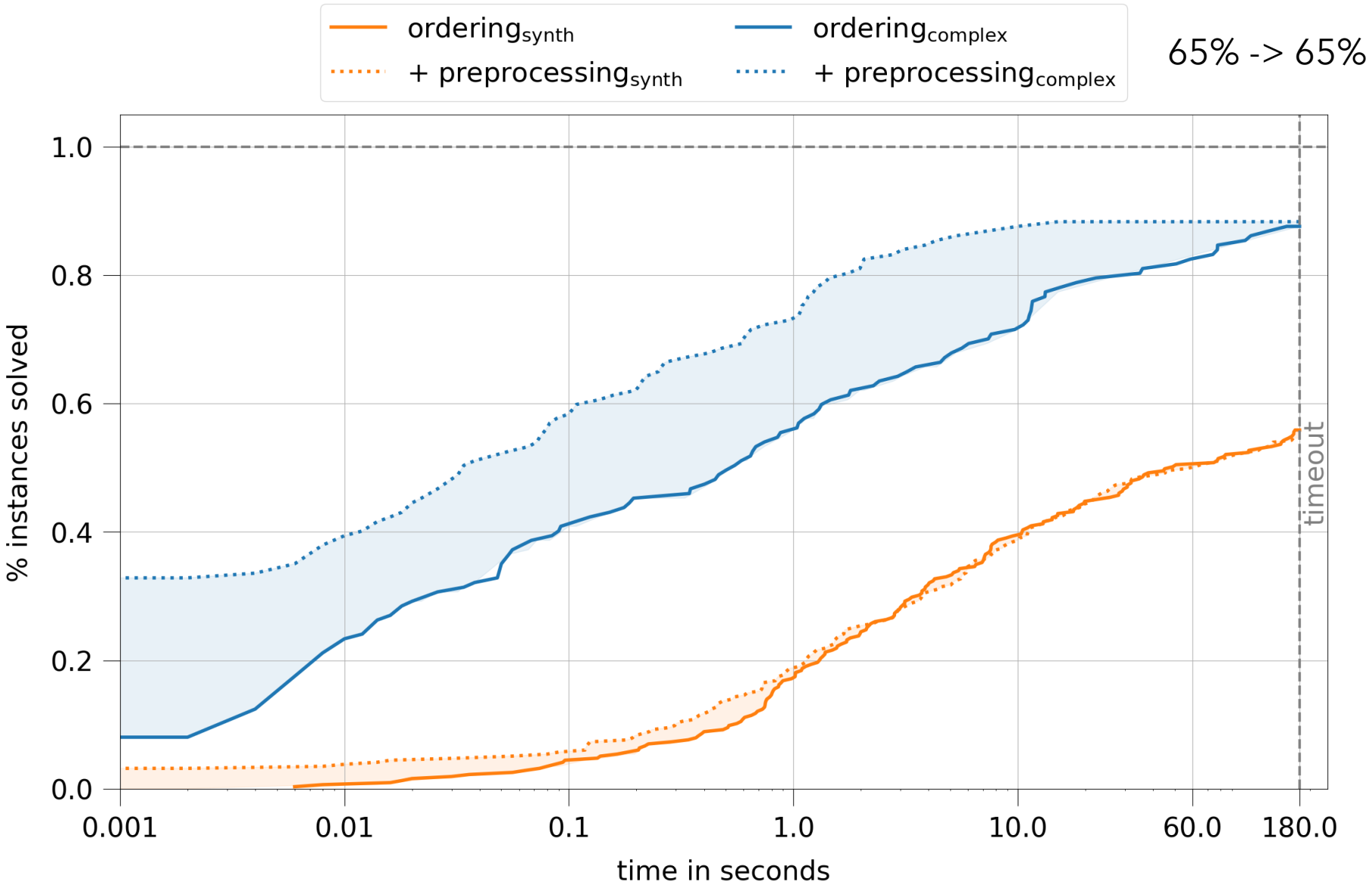
Cycle Packing in Callback



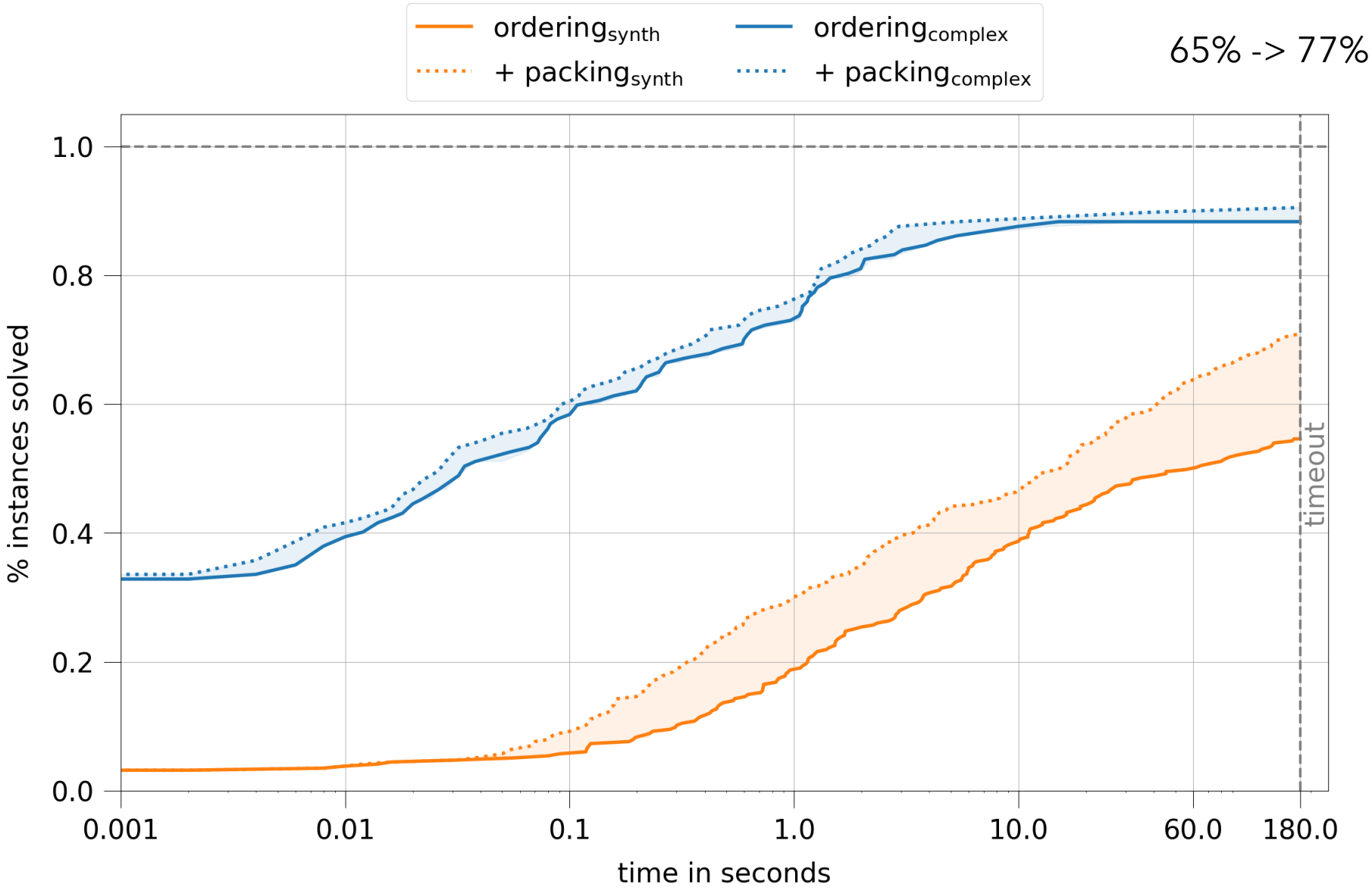
Ordering

- Add one constraint for each edge
- Higher number of variables and constraints, but no callback is needed
- Tried out with and without preprocessing

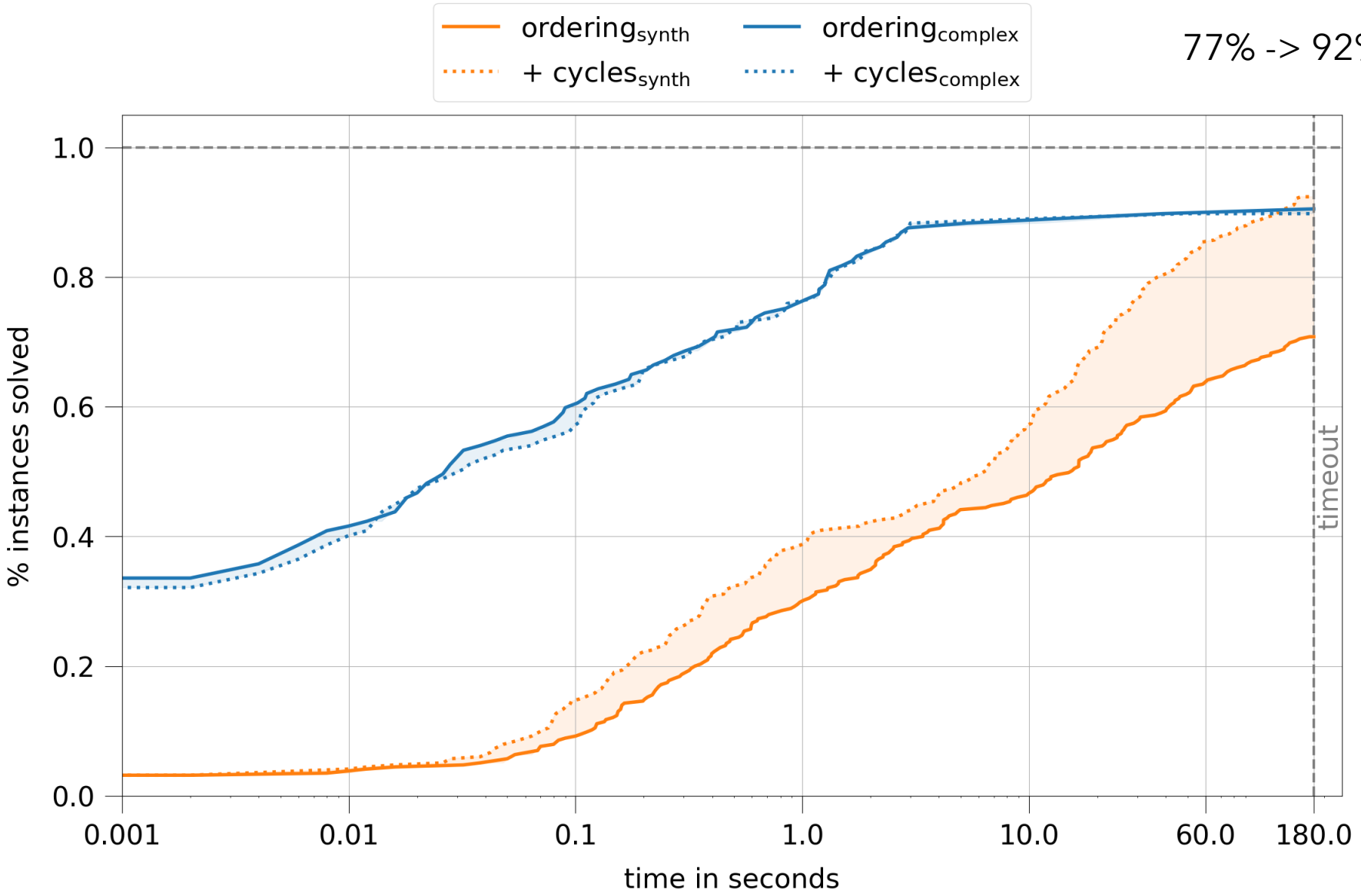
Ordering (Preprocessing)



Ordering (Cycle Packing)



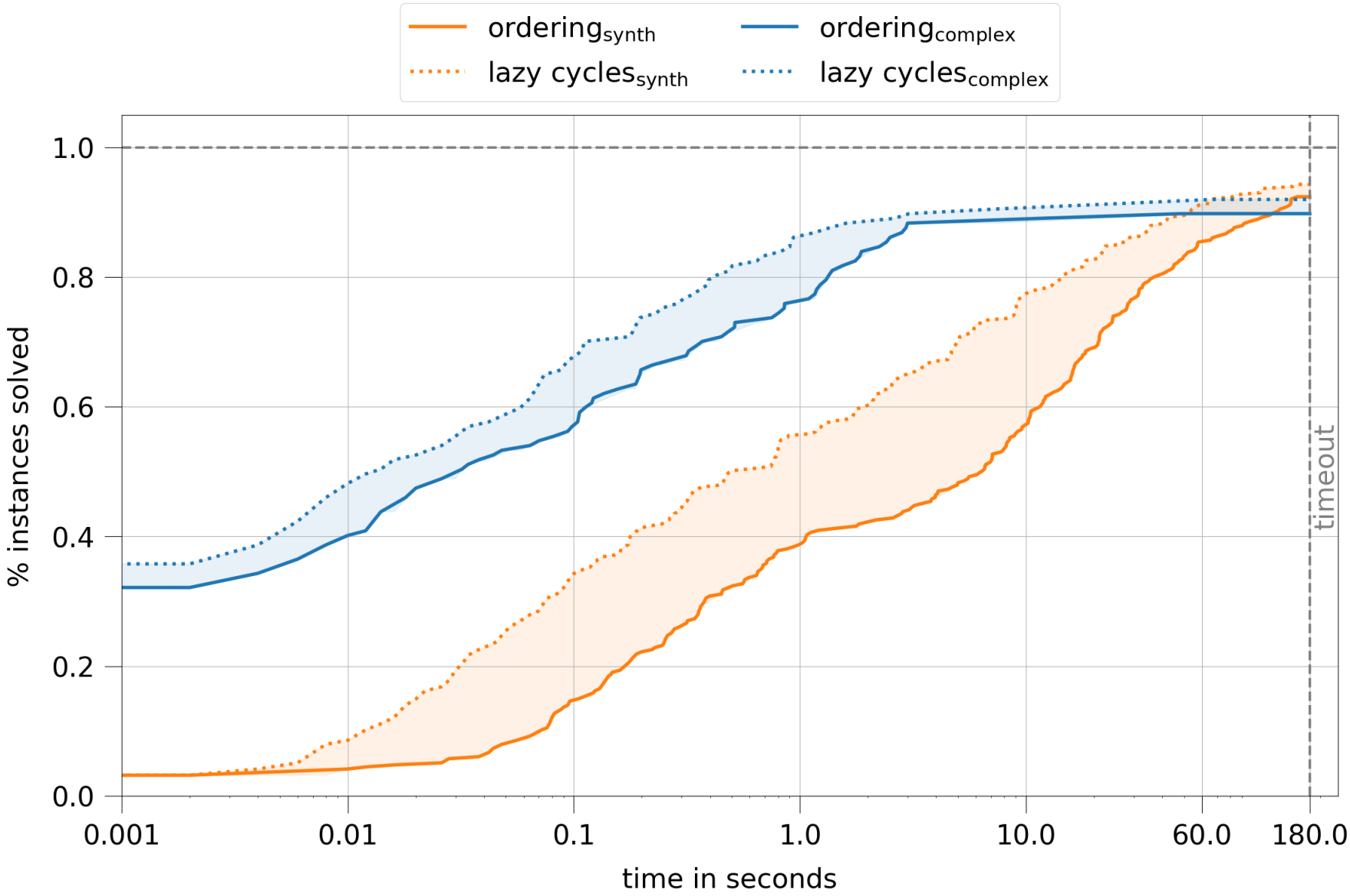
Ordering (Additional Cycles)



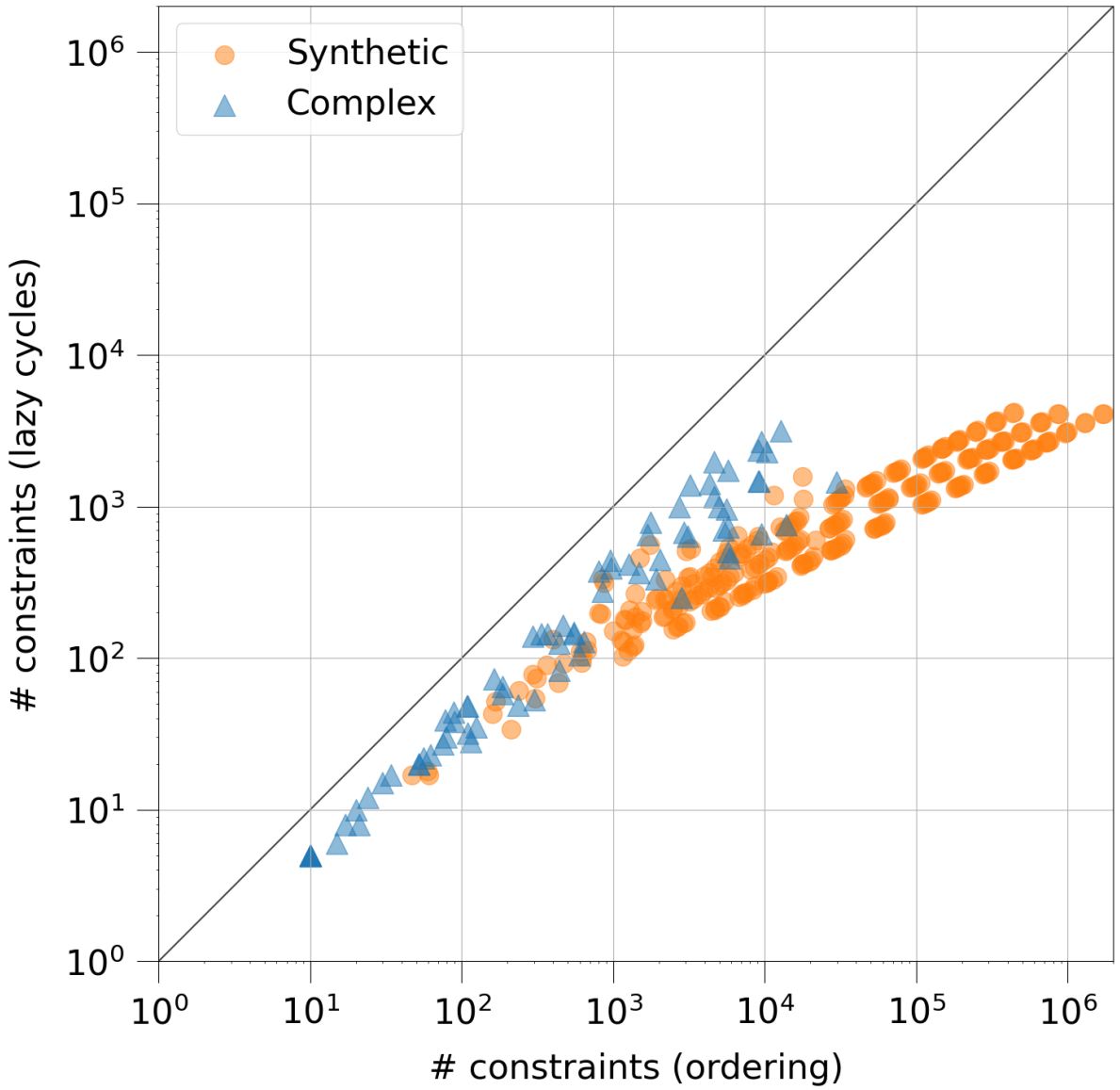
Further constraints

- We found it challenging to find constraints that improve the runtime
- All beneficial constraints are related to cycles
- Making a lower-bound-/ upper-bound-constraint for k increased the runtime in most cases

Lazy Cycles vs Ordering



Lazy Cycles vs Ordering



Summary

- Lazy Cycles: 423/452 (93.6%)
  - Synthetic: 297/315 (94.3%)
  - Complex: 126/137 (92.0%)
- Ordering: 414/452 (91.6%)
  - Synthetic: 291/315 (92.4%)
  - Complex: 123/137 (89.8%)
- Both ILP Solvers perform pretty good
  - However, we will try to catch up with our own solver!

Do you have questions?