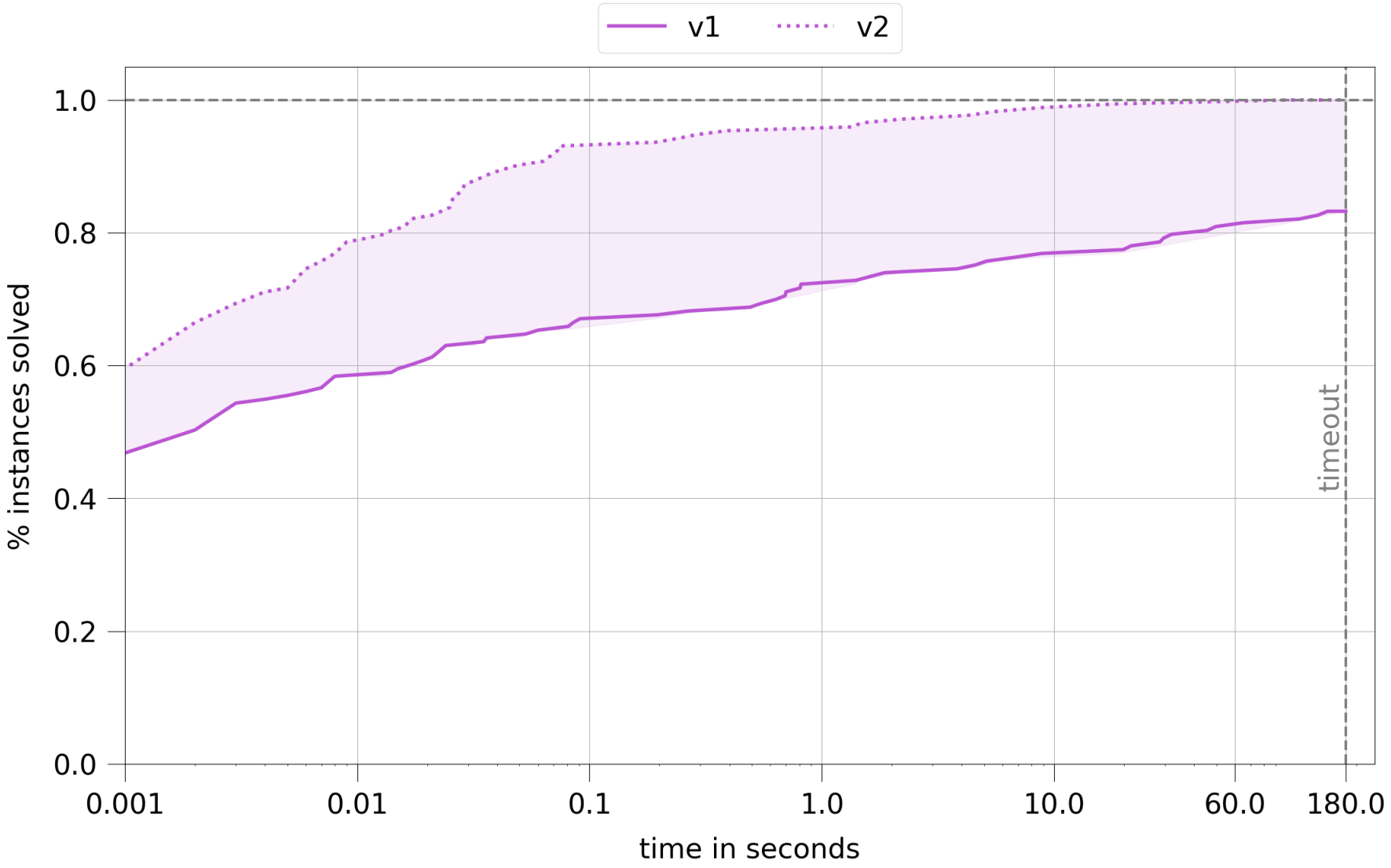
# Algorithm Engineering: Milestone 2

Henri Dickel, Matija Miskovic
& Lennart Uhrmacher

# Introduction

- Previous state:
  - Base search tree algorithm
  - Finding cycles to branch on with DFS
  - Preprocessing with Tarjan's Algorithm
- Current state:
  - Reduction rules in preprocessing
  - Forbidden nodes
  - Flowers
  - Finding cycles with BFS

# Introduction



# Introduction

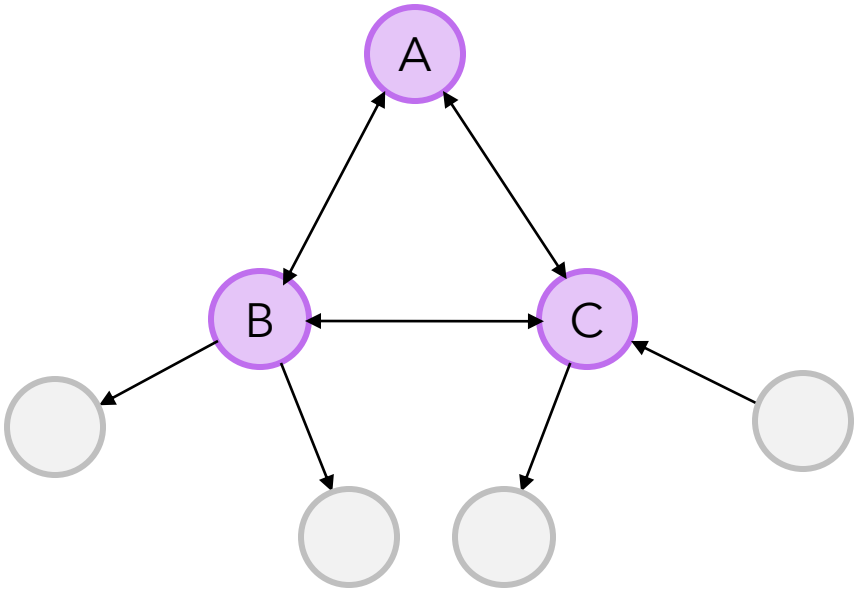
- What will we do today?
  - Go through each new feature
  - Take a closer look on how it works
  - Analyze the importance of the feature

# Reduction rules

- Decomposition into cyclic components already used
- Reduction rules:
  - Remove self edges
  - Chain rule
  - Remove trivial vertices
- Advanced triangle reduction rule
- Only used as preprocessing

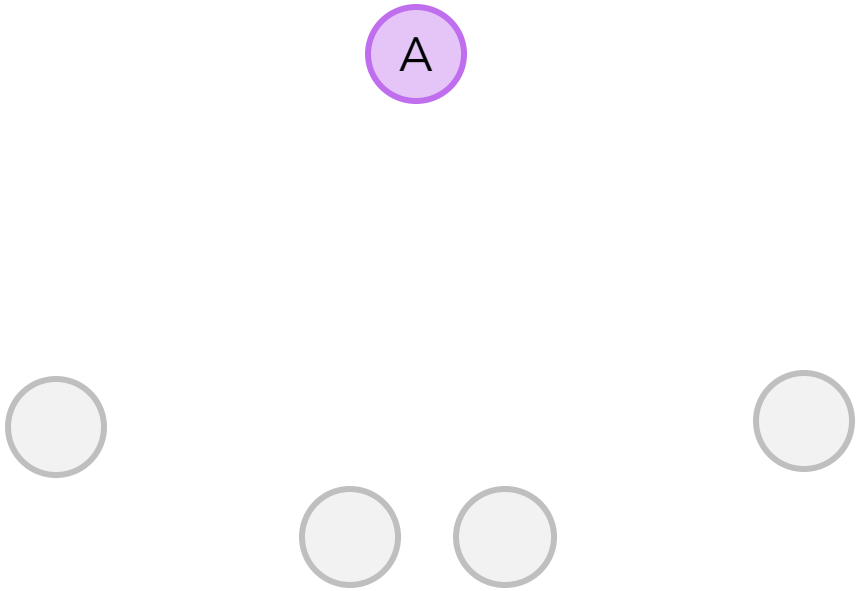
# Reduction rules

- Advanced Triangle reduction rule:

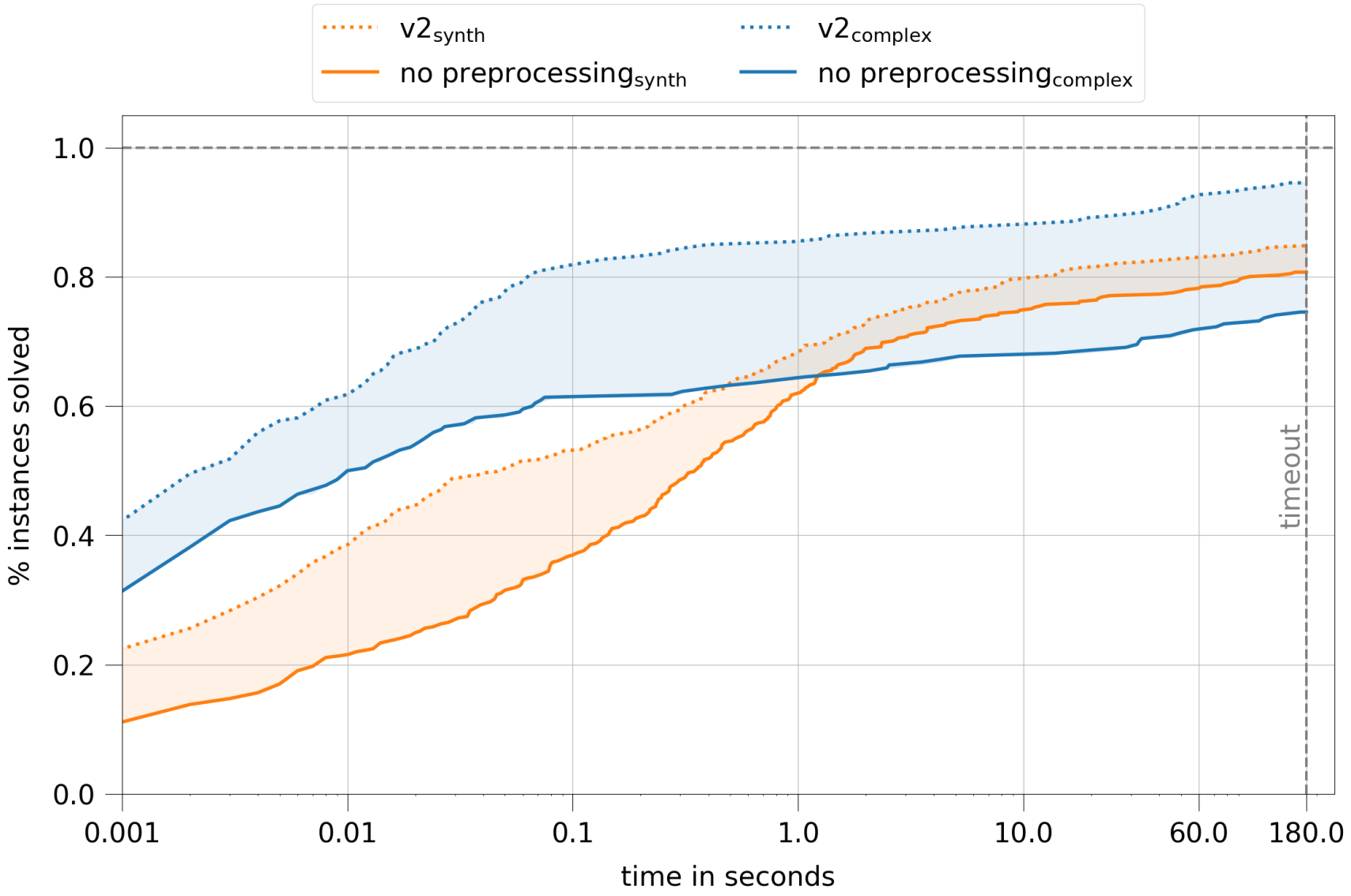


# Reduction rules

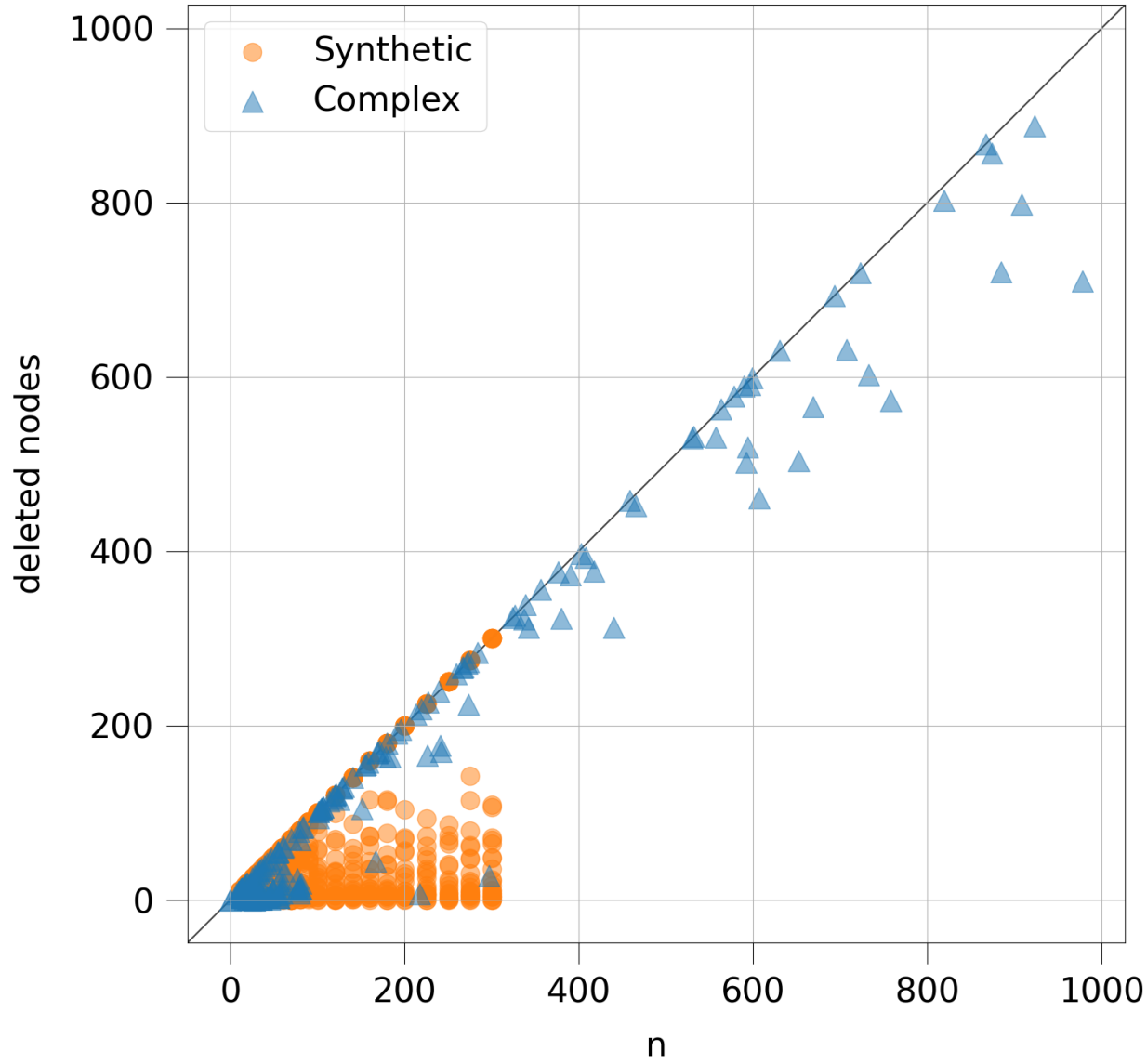
- Advanced Triangle reduction rule:



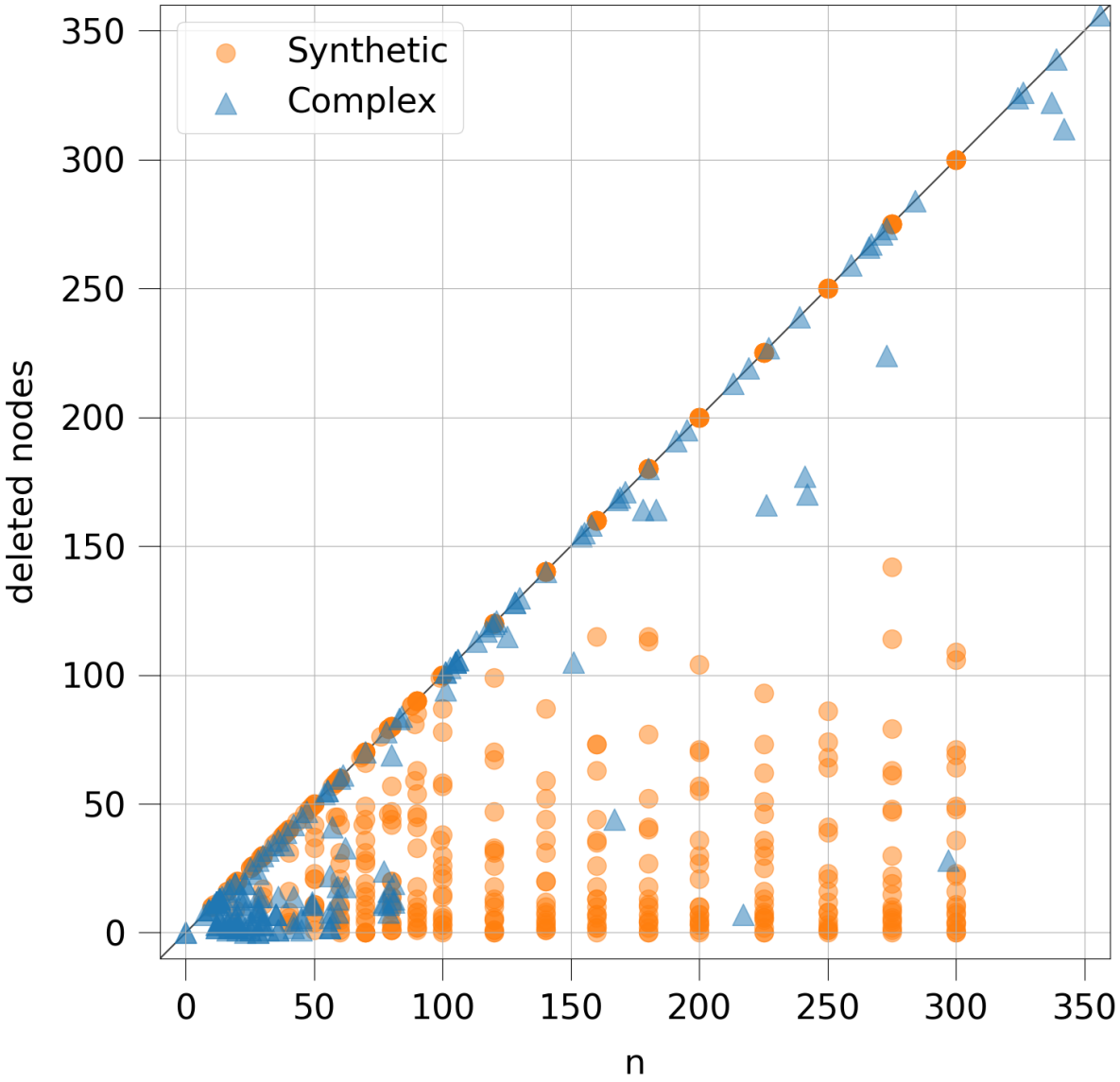
# Reduction rules



# Reduction rules



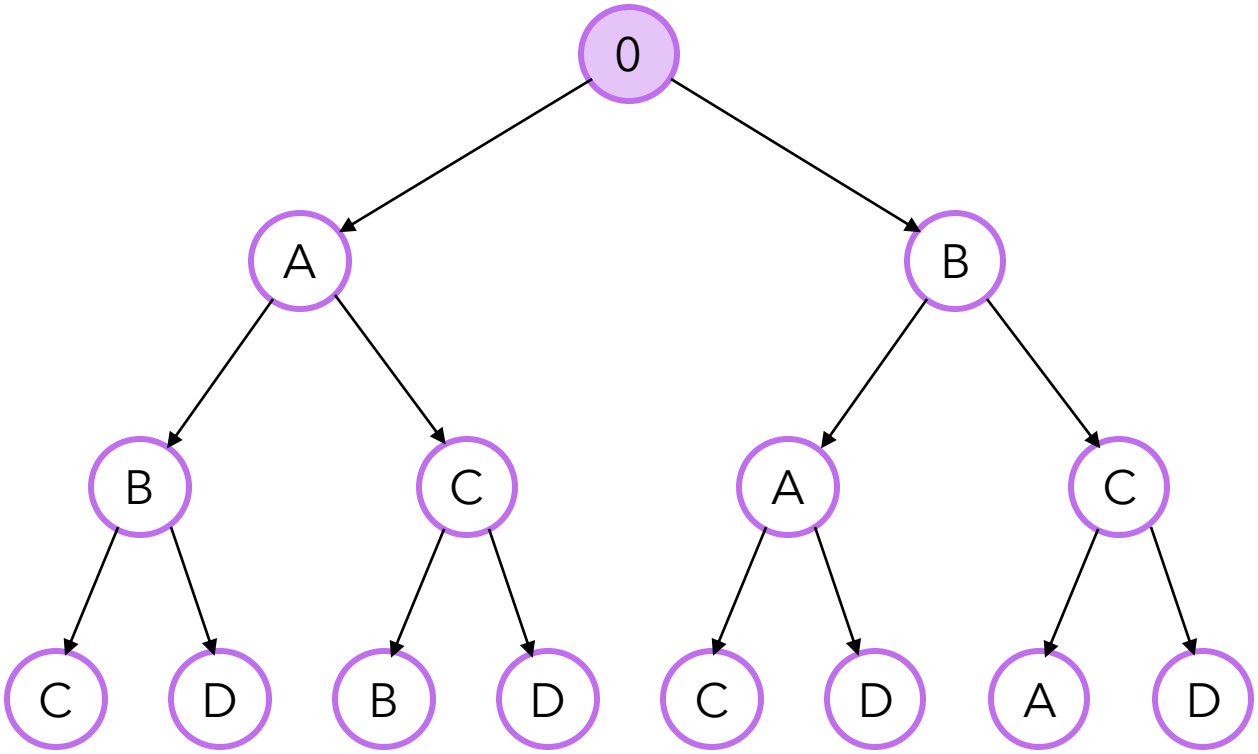
# Reduction rules



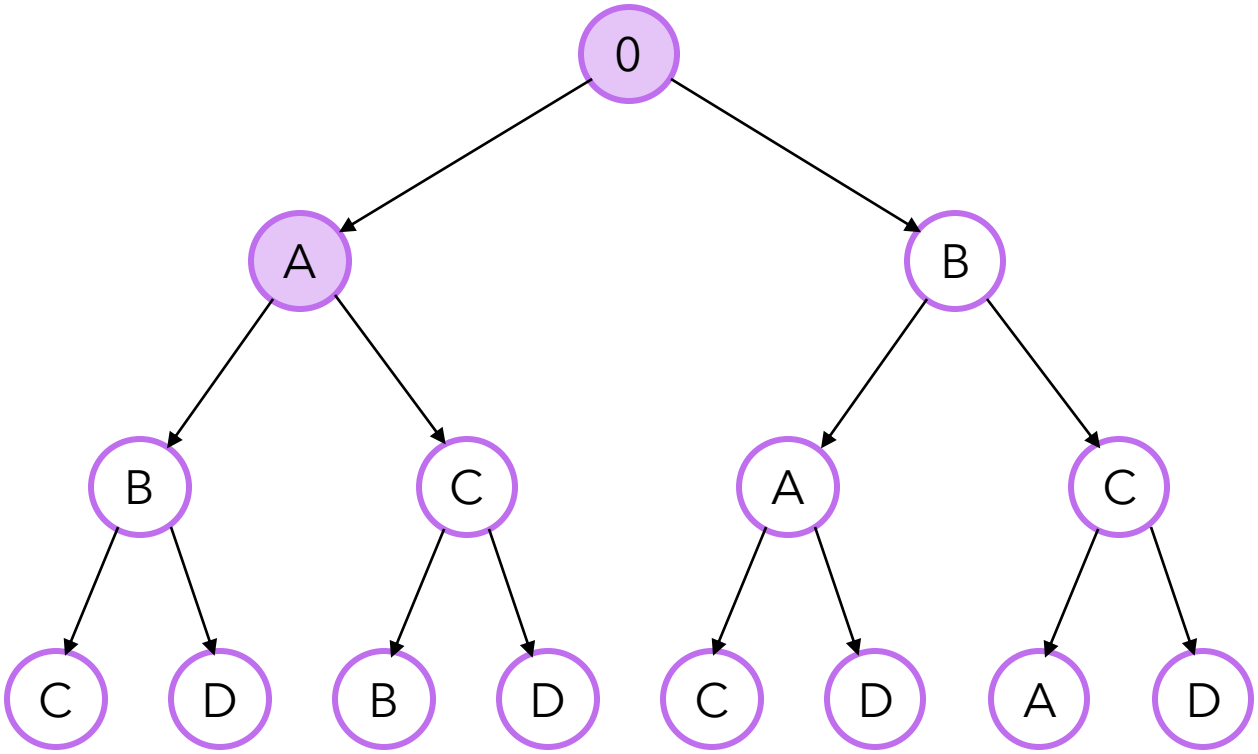
# Forbidden nodes

- Forbidden node algorithm:
  - When a node is deleted, it is labeled as 'forbidden'
  - In further branching, 'forbidden' nodes can be skipped

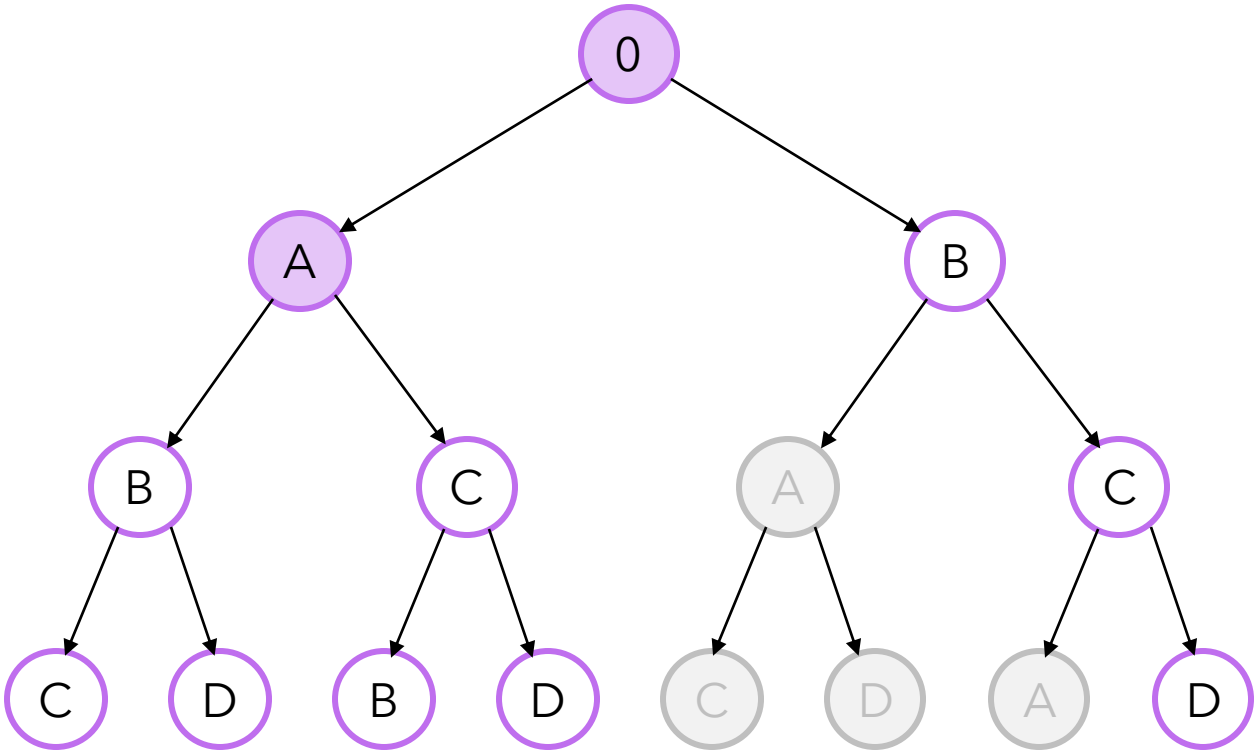
# Forbidden nodes



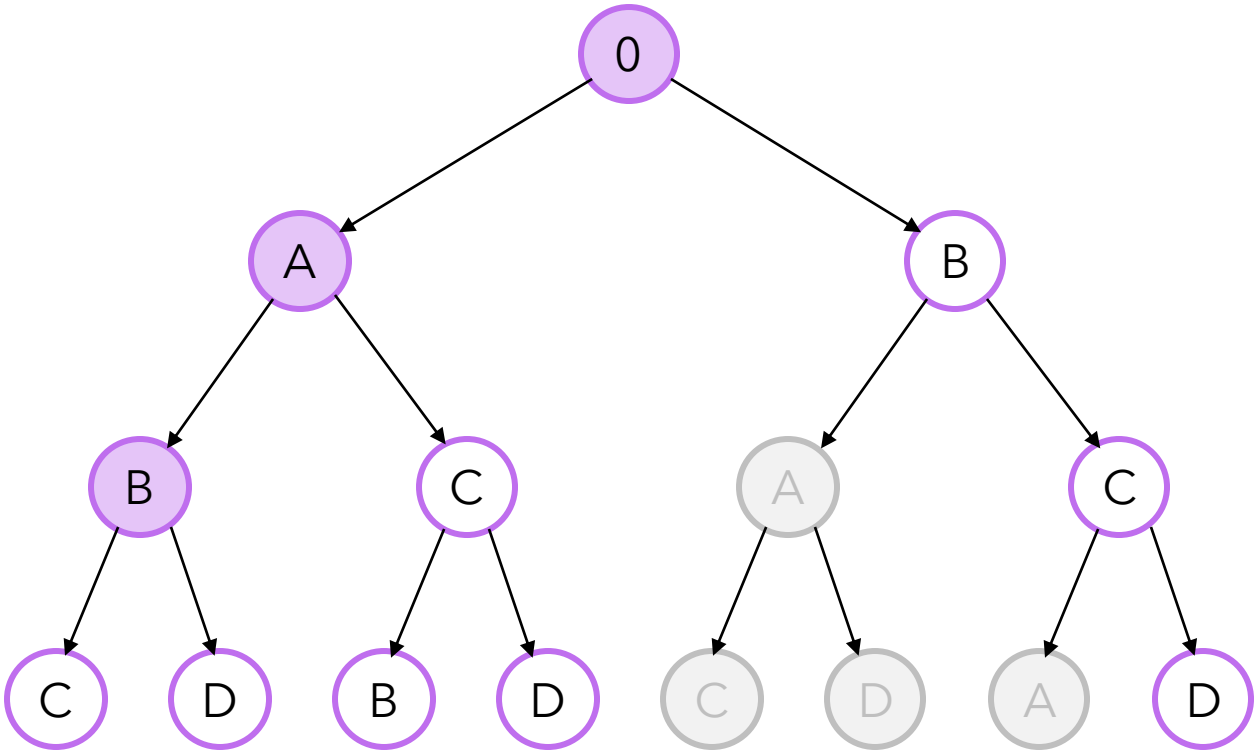
# Forbidden nodes



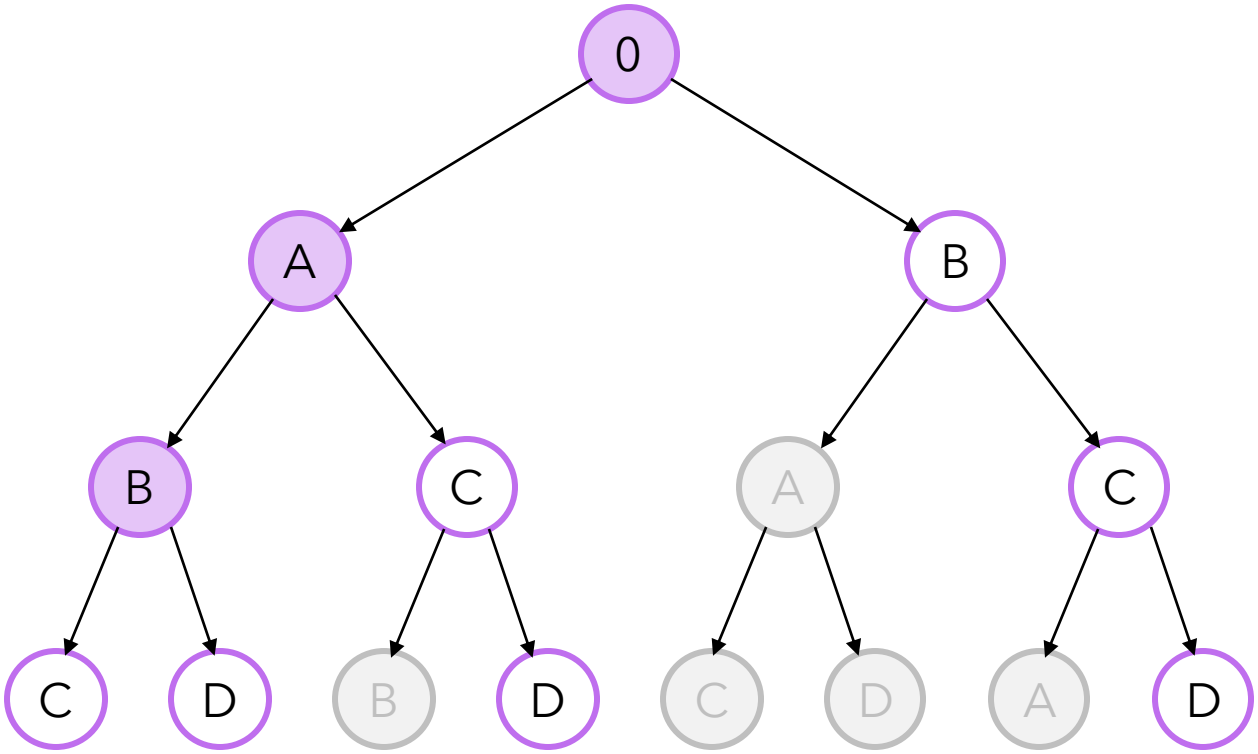
# Forbidden nodes



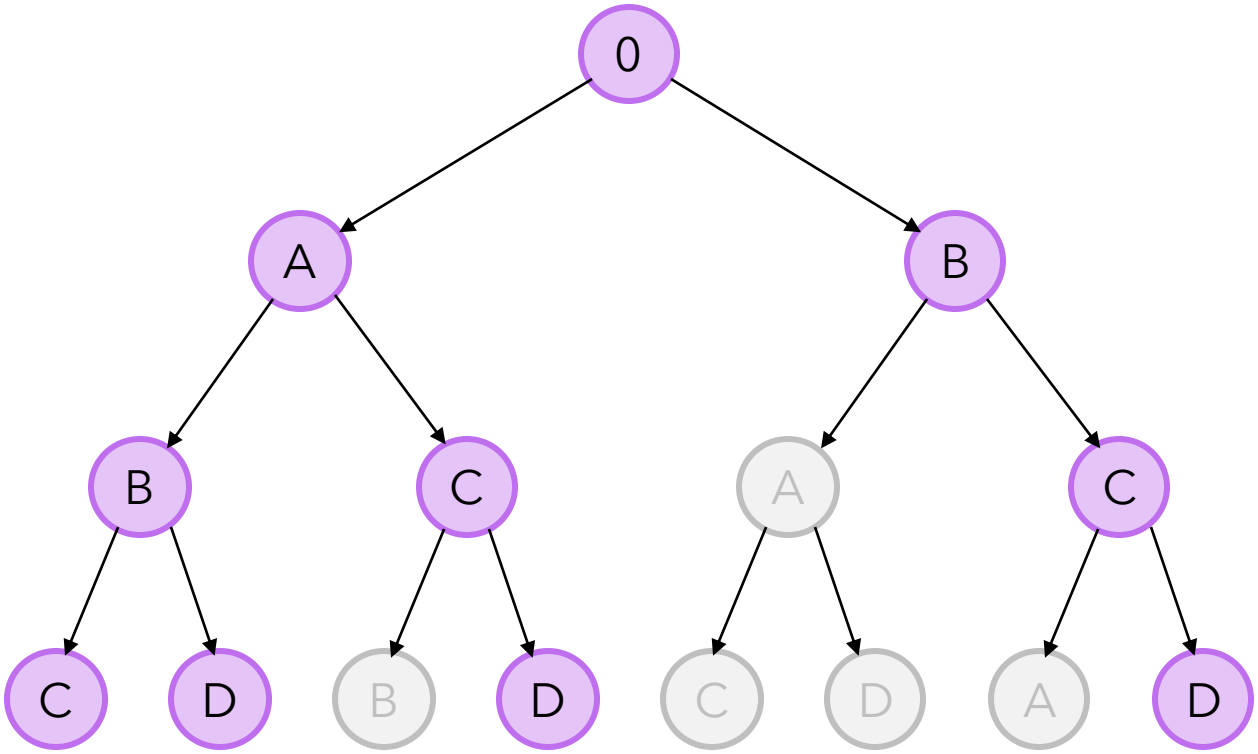
# Forbidden nodes



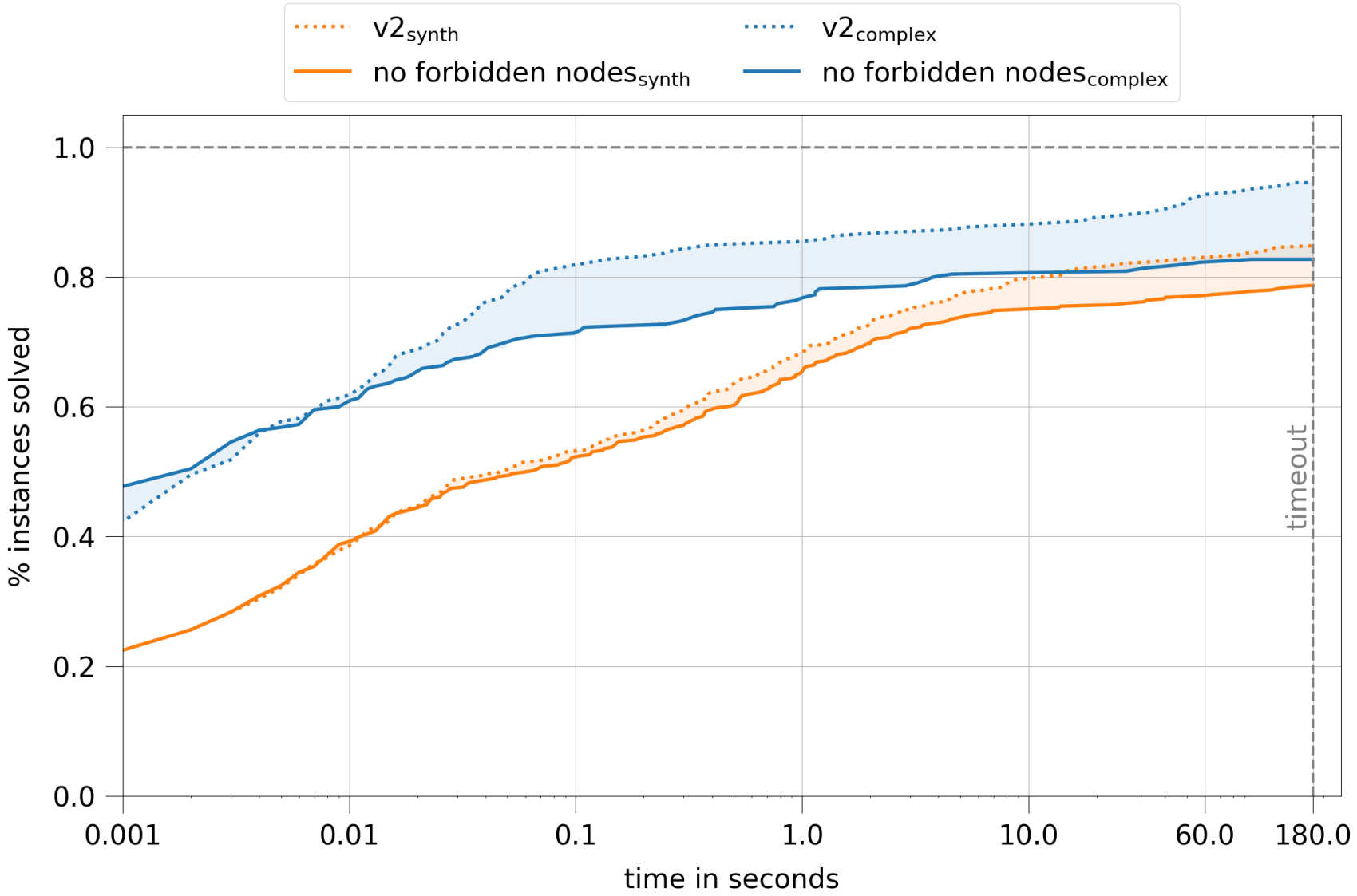
# Forbidden nodes



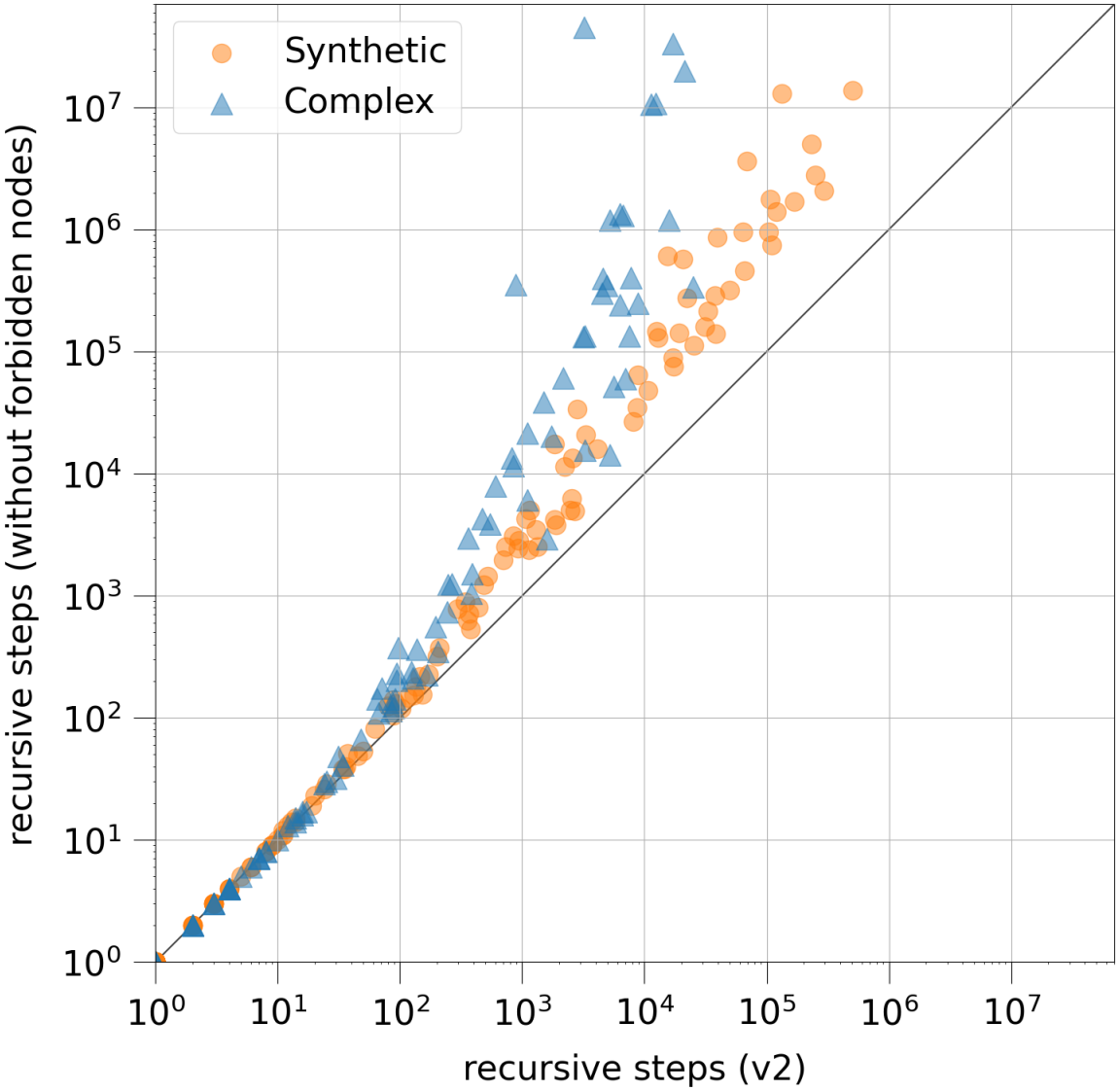
# Forbidden nodes



# Forbidden nodes



# Forbidden nodes



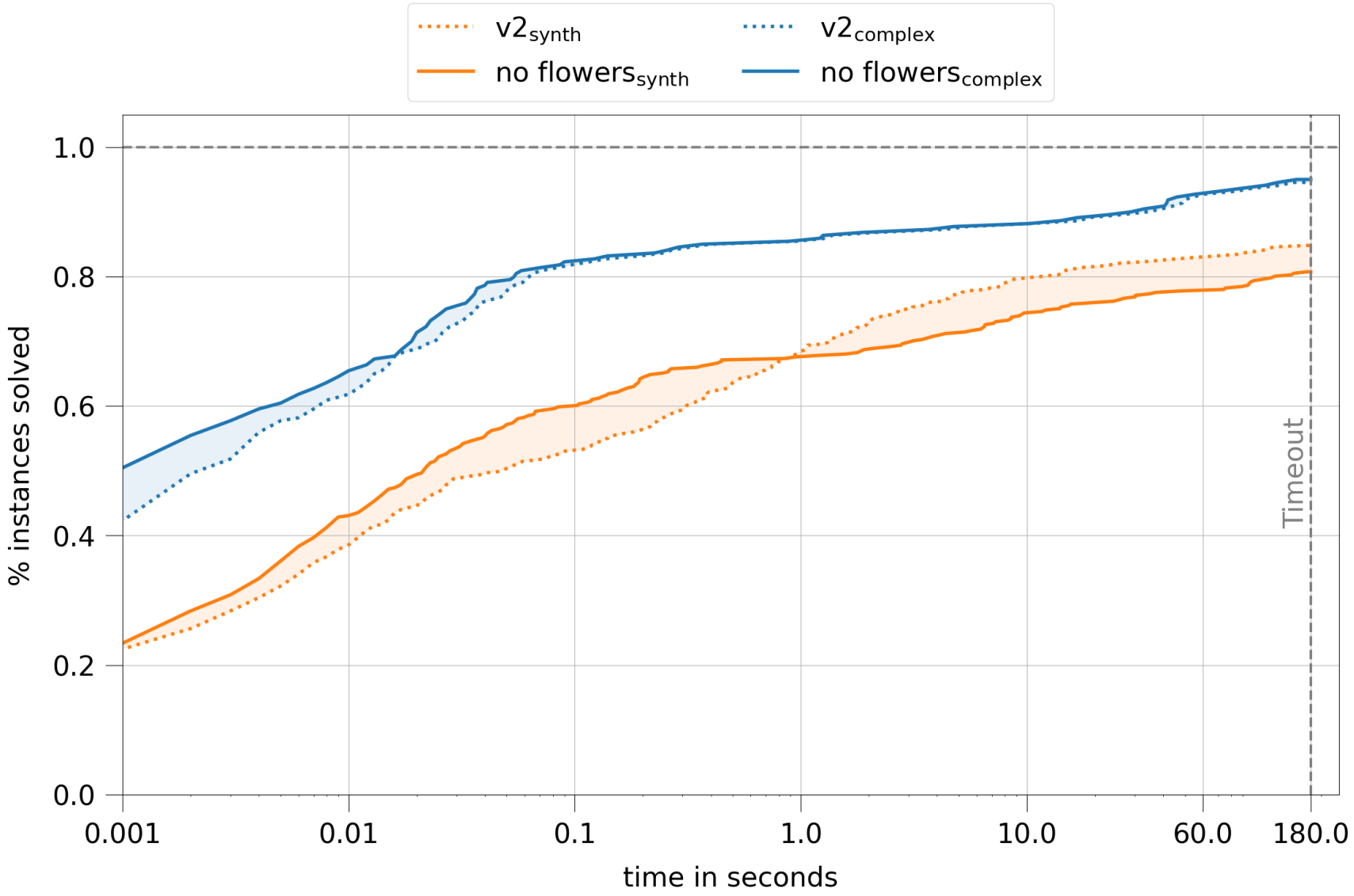
# Flowers

- General idea: petal(V) = number of disjoint cycles connected in V
- Approach from lecture:
  1. Create graph
  2. Calculate Max-Flow/Min-Cut with Ford-Fulkerson
- Optimization: Don't create a new graph for every node

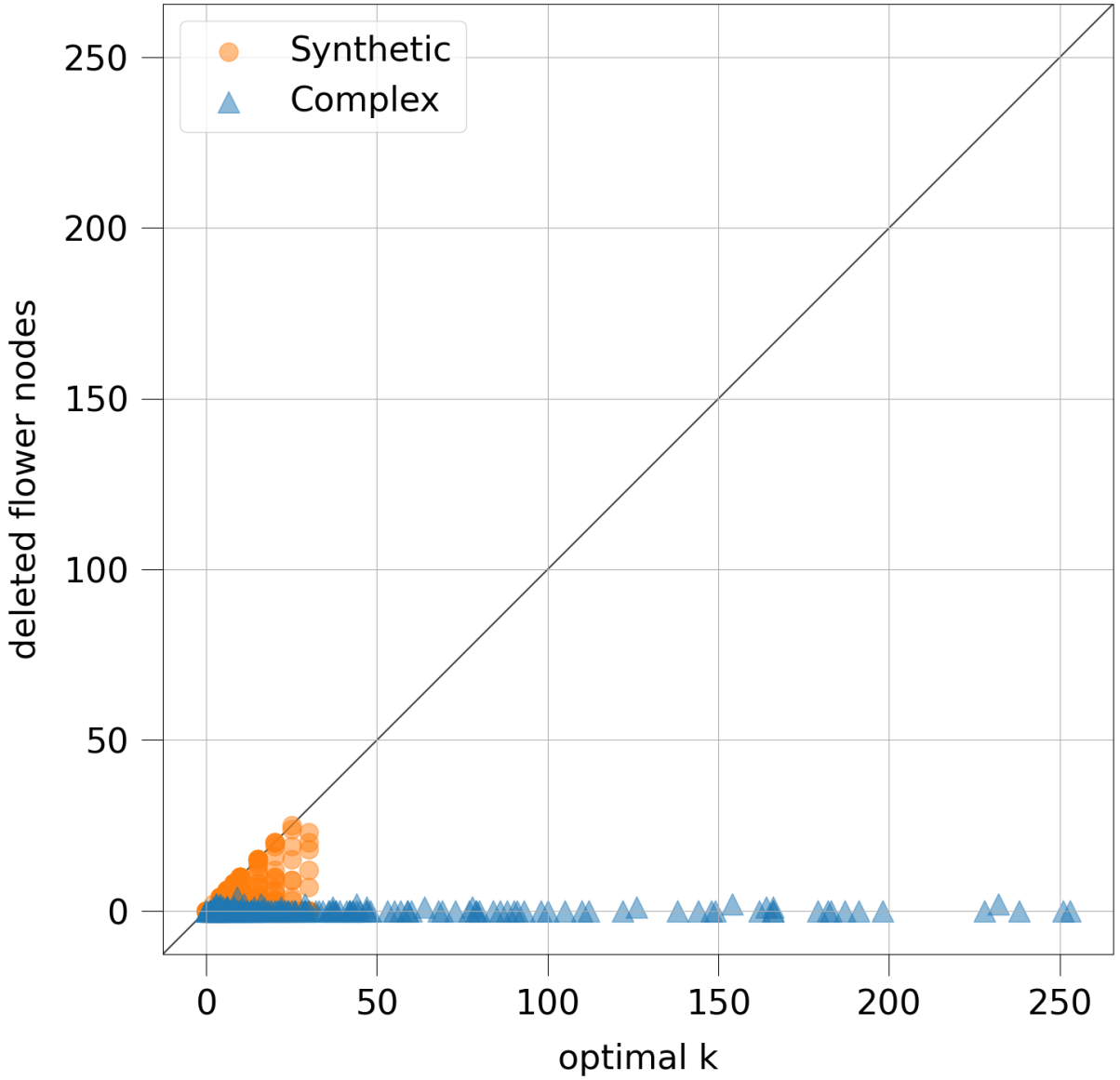
# Flowers

- Petal rule: when petal(V) > k, delete V and increase k by one
- How do we use it:
  1. Calculate petal-values for all nodes
  2. Calculate for current k: are there nodes, that can be deleted with the petal rule? ➞ If yes, delete one and recalculate petal-values
  3. When the amount of deleted nodes is higher that our current k, there is no solution for this k
- Optimization: reduce petal-values by 1 instead of recalculating
- When do we use it? For each k before we start branching

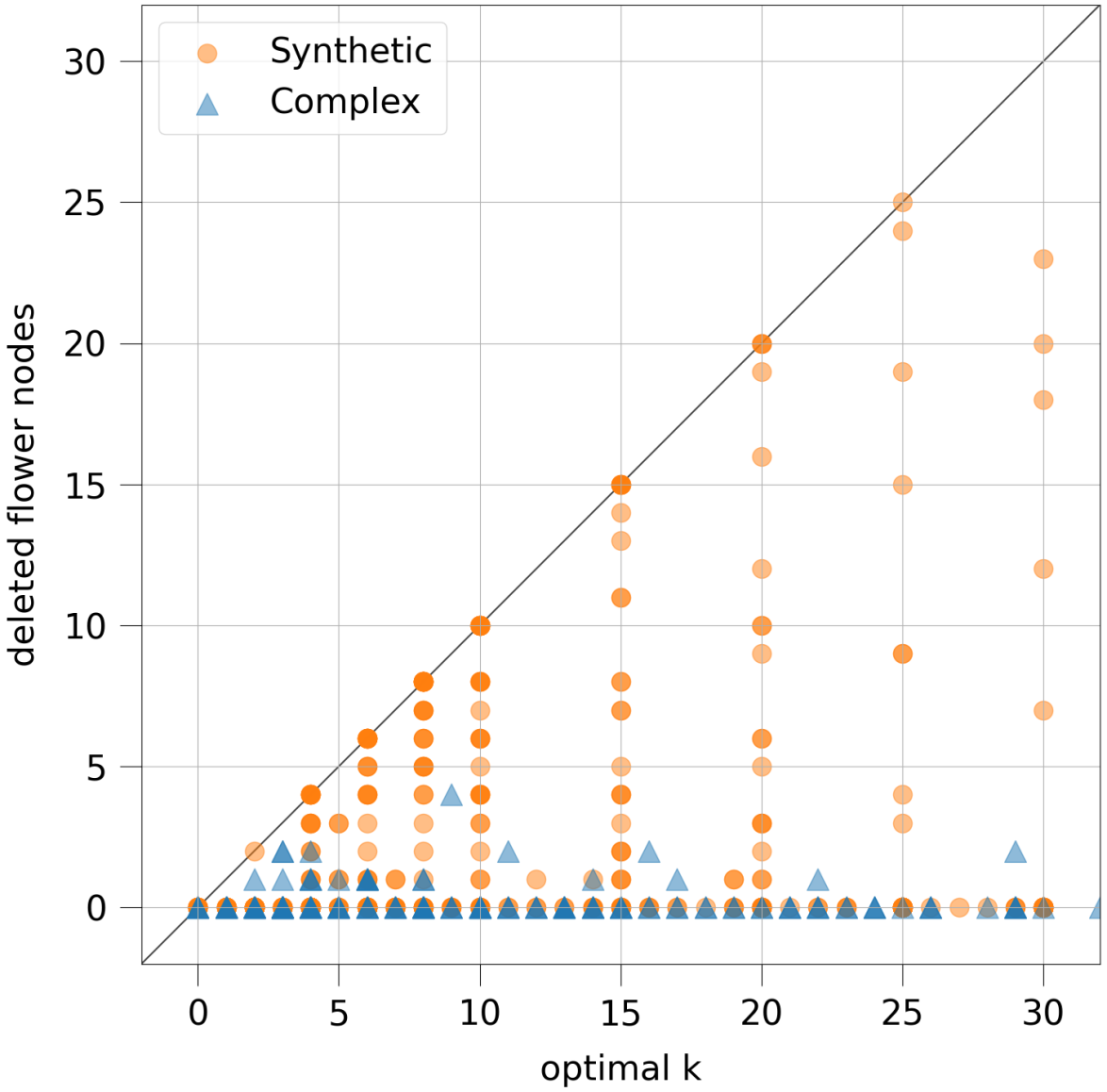
# Flowers


v2synth
v2complex
no flowerssynth
no flowerscomplex
1.0
0.8
0.6
0.4
0.2
0.0
% instances solved
Timeout
0.001
0.01
0.1
1.0
10.0
60.0
180.0
time in seconds

# Flowers

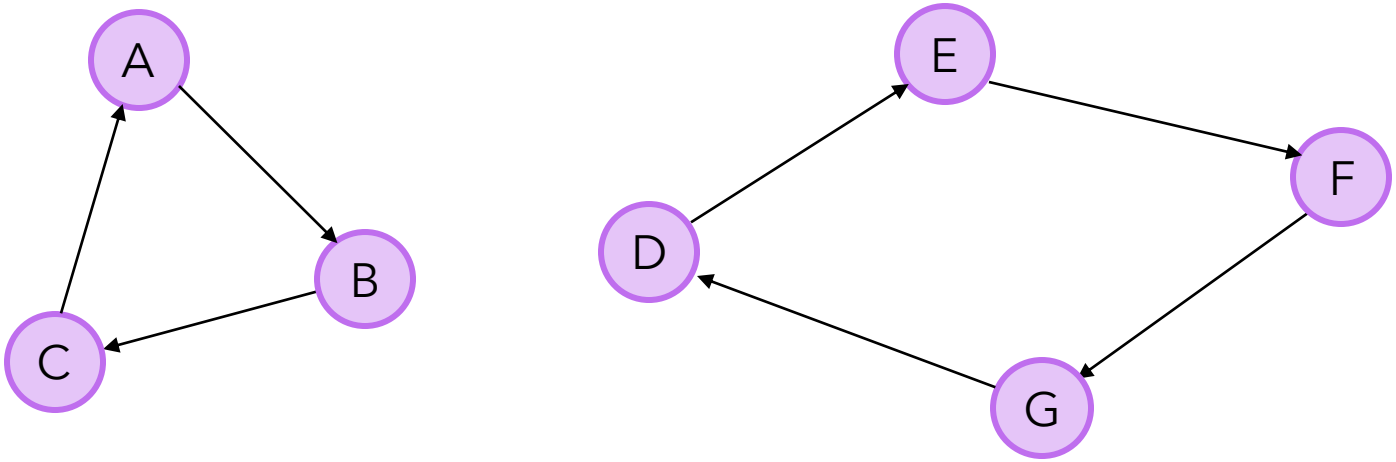

Synthetic
Complex
deleted flower nodes
optimal k
0
50
100
150
200
250

# Flowers


Synthetic
Complex
deleted flower nodes
optimal k
0
5
10
15
20
25
30

# Finding cycles

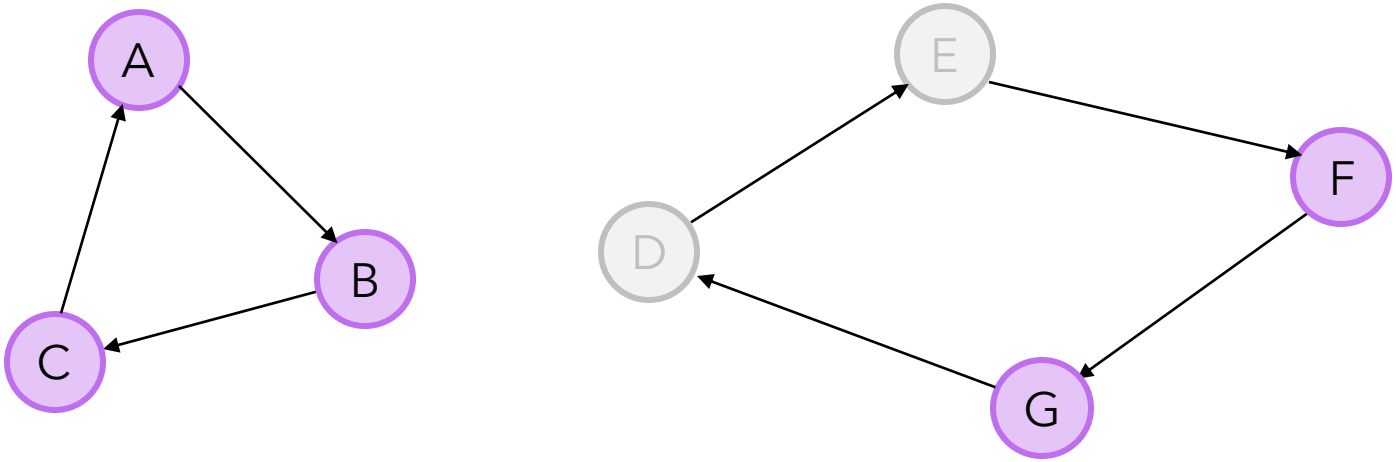
- Forbidden nodes do affect the cycles in the graph:




A-B-C is the shortest cycle, but is it necessarily the best to branch on?

# Finding cycles

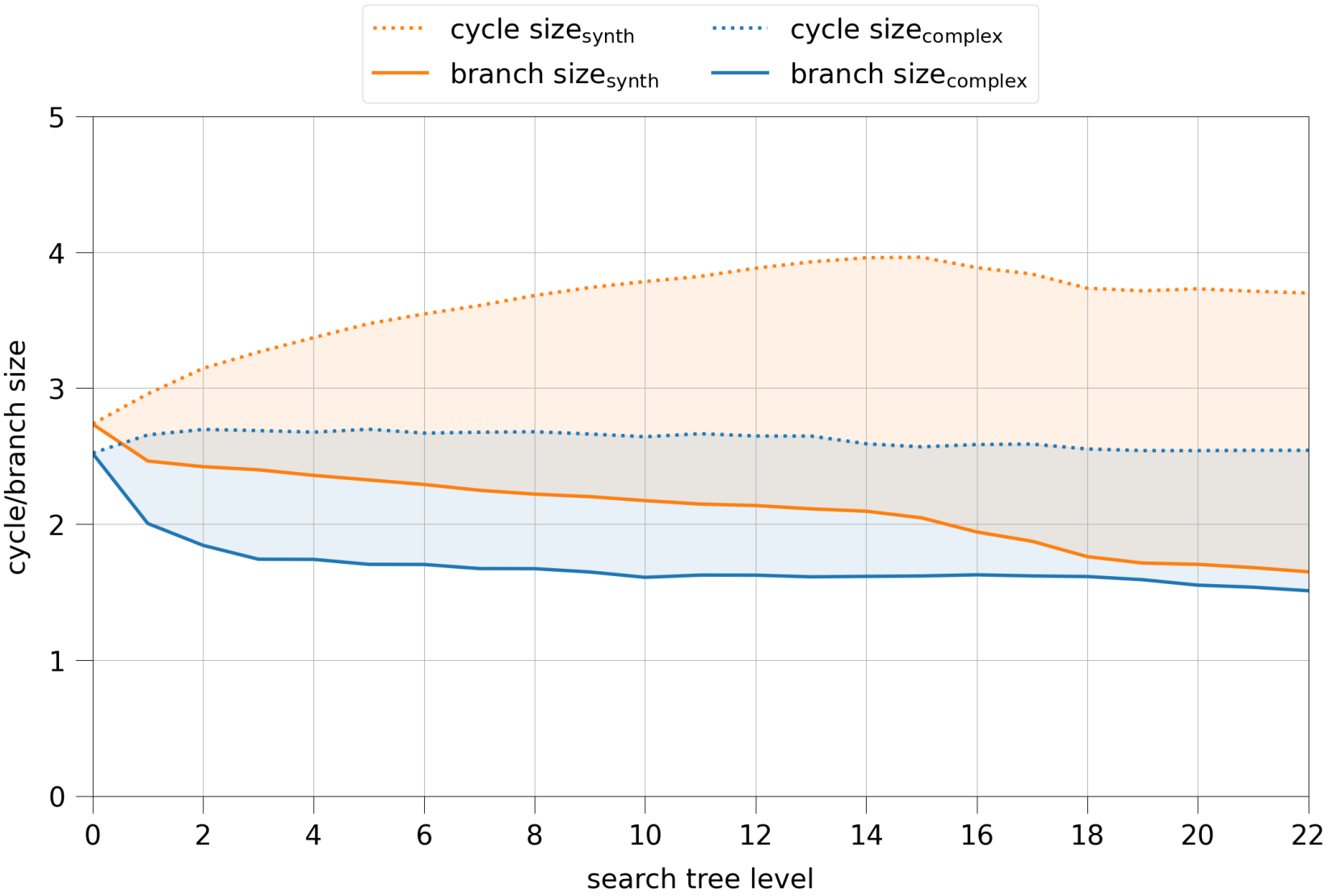
- Forbidden nodes do affect the cycles in the graph:




A-B-C is the shortest cycle, but is it necessarily the best to branch on?

➔ No, D-E-F-G branches only 2 instead of 3 times

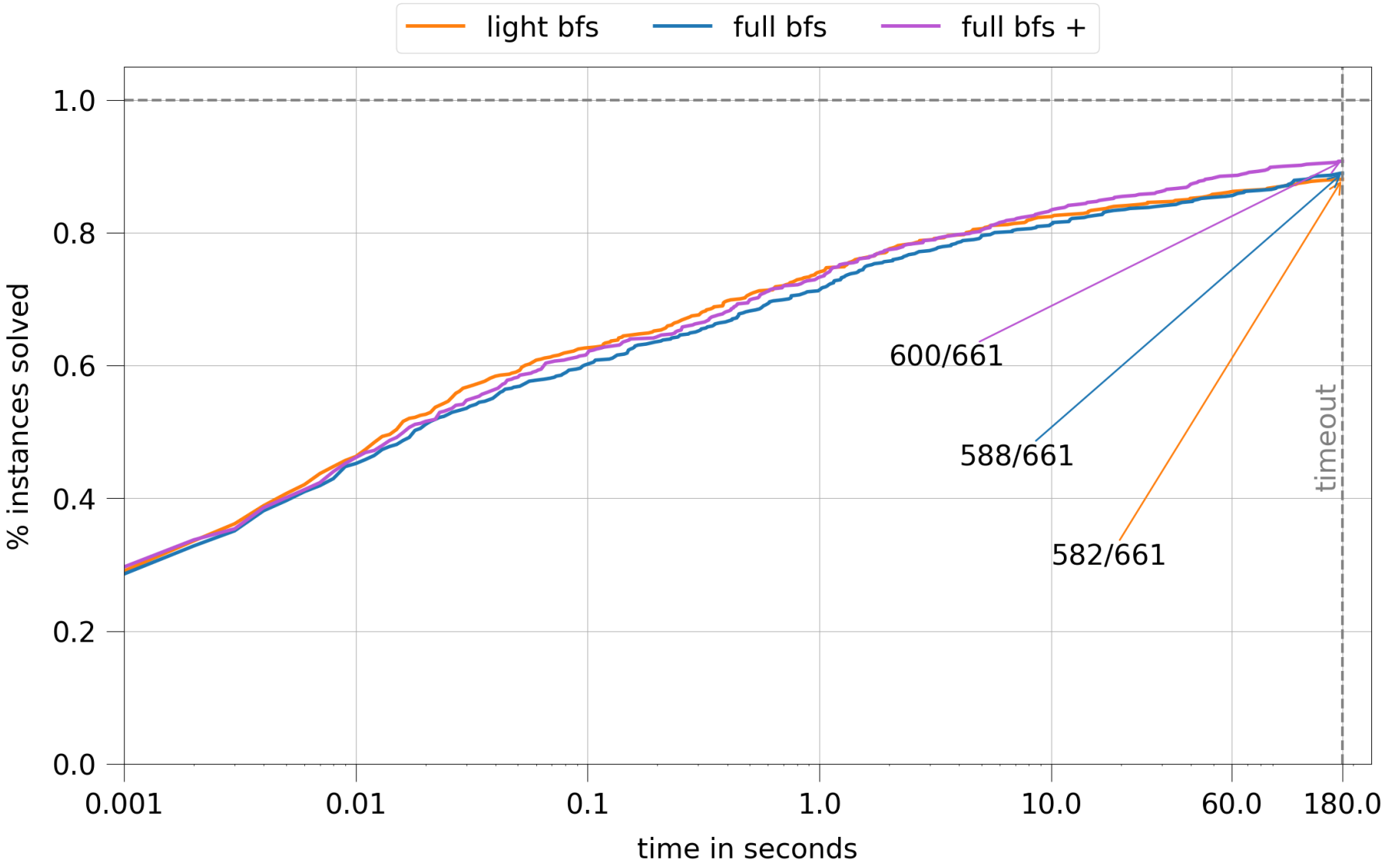
# Finding cycles



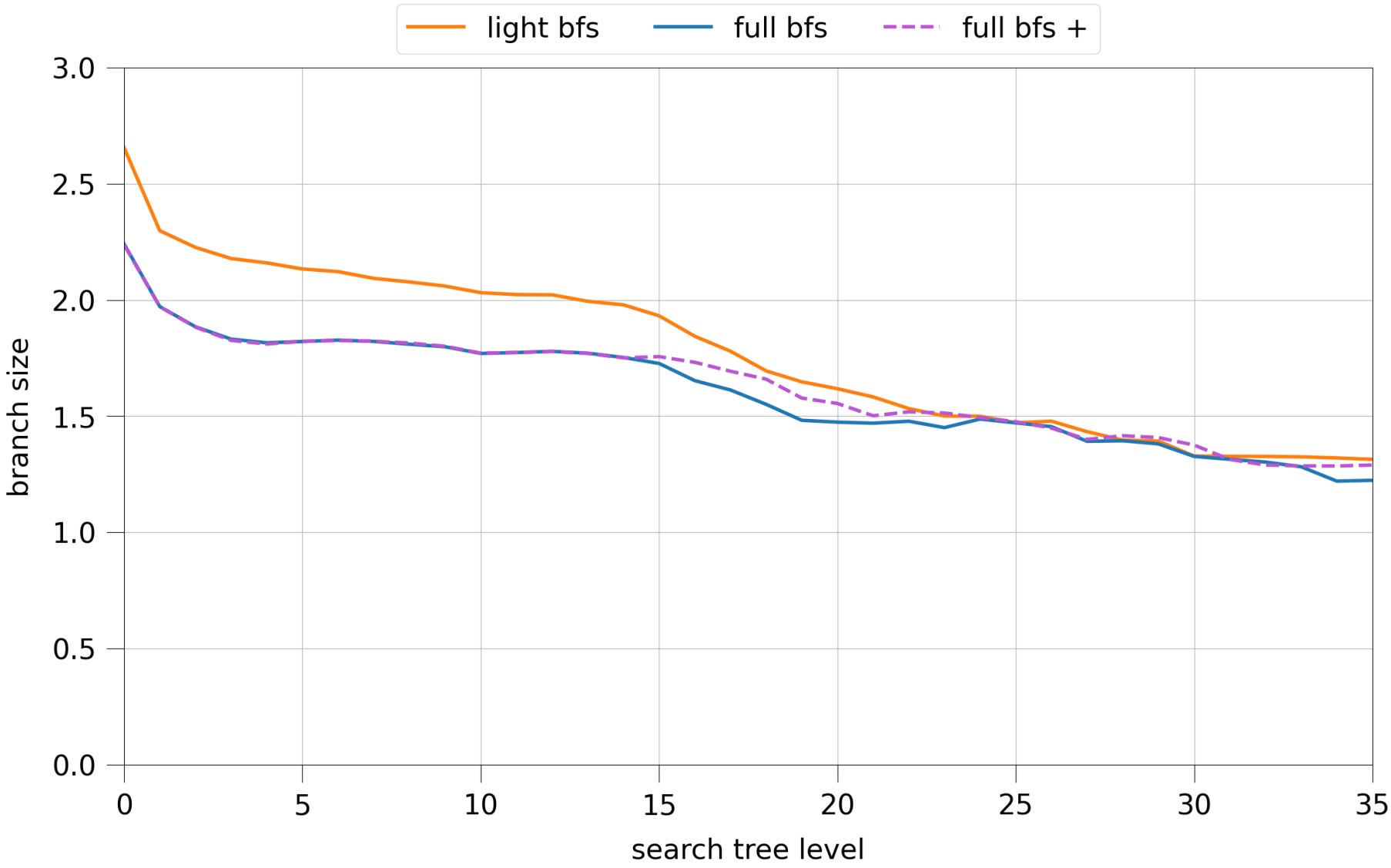
# Finding cycles with BFS

- They can be found with breadth-first search
- We tried out three approaches:
  1. Full BFS: in every node of the search tree, run a BFS from every node of the graph and pick the best cycle ➔ always finds the best cycle, high runtime
  2. Full BFS + (with caching): calculate the best cycle for every node only once in the beginning and update it, when their cycle is destroyed ➔ mostly finds the best cycle, medium runtime
  3. Light BFS: In every node of the search tree, run a modified BFS on the graph that returns the first found cycle ➔ not always finds the best cycle, fast runtime
- Light BFS was the best for a long time until we added 1 line of code for the improvement that I just showed you

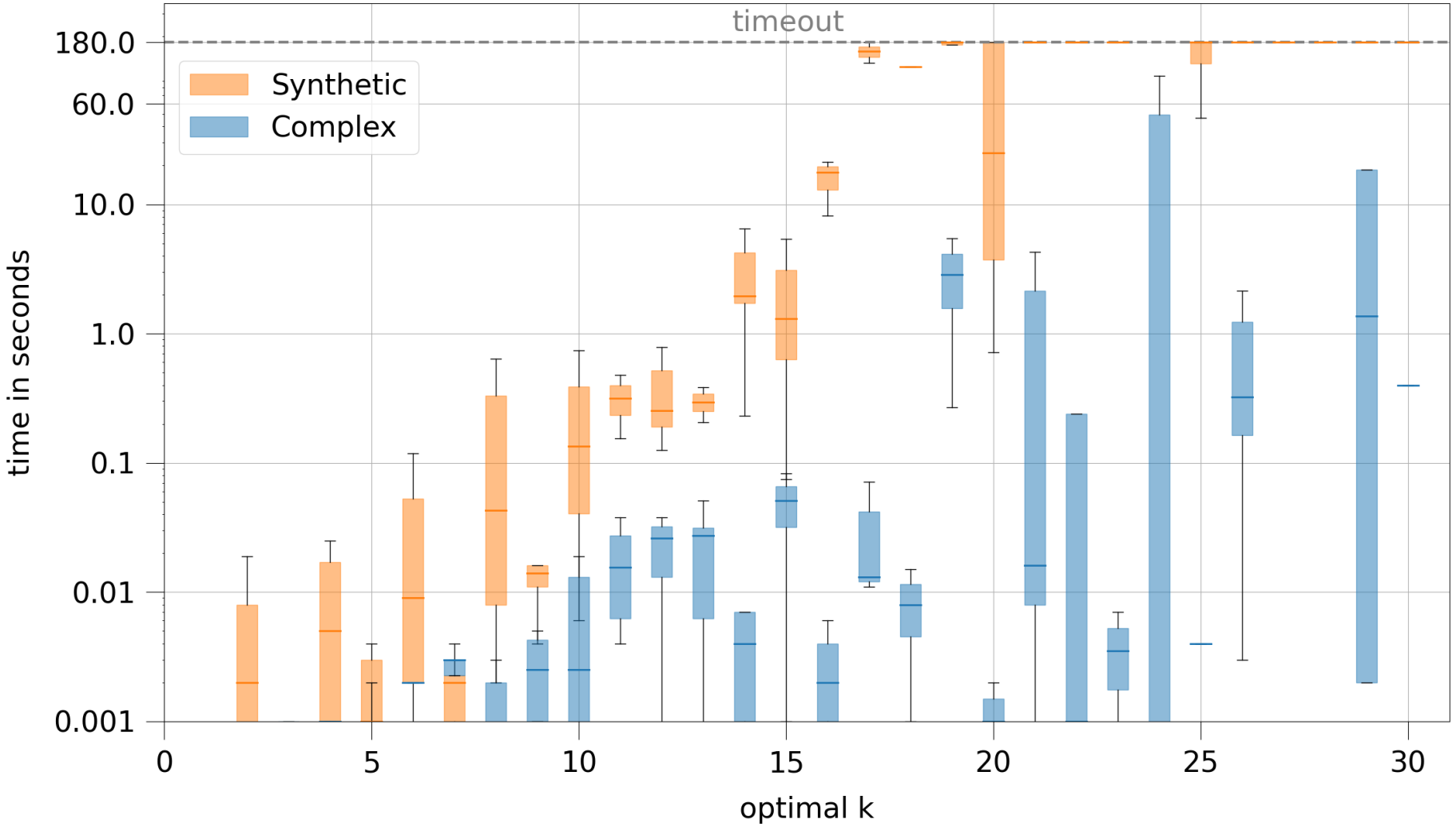
# Finding cycles with BFS



# Finding cycles with BFS



# Summary


timeout
Synthetic
Complex
180.0
60.0
10.0
1.0
0.1
0.01
0.001
time in seconds
0
5
10
15
20
25
30
optimal k

# Summary & Outlook

- Percentage solved: 88.1% (90.8%)
  - Synthetic: 84.8% (88.7%)
  - Complex: 94.6% (95.0%)
- Goals for next milestone:
  1. Use reductions, cyclic decomposition and flowers in branching
  2. Improve and use the best BFS approach
  3. Use lower bounds (from last lecture)

# Do you have questions?