Tree search algorithms for the sequential ordering problem

Luc Libralesso - Abdel-Malik Bouhassoun
Hadrien Cambazard - Vincent Jost

September 2 2020 - ECAI 2020

Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France
email: luc.libralesso@grenoble-inp.fr

Context & Methodology

Conventional wisdom - Exact Methods and Metaheuristics

Two ways to solve a **Operations Research** problem

- **Exact methods:** MIPs, CP, Branch and Price ...

Conventional wisdom - Exact Methods and Metaheuristics

Two ways to solve a **Operations Research** problem

- **Exact methods:** MIPs, CP, Branch and Price ...
- **(Meta-)heuristics:** local-search, genetic algorithms, ant colony ...

Conventional wisdom - about Tree Search

Mathematical Programming Solver based on Local Search ([1]):

> *“ Tree search approaches like branch-and-bound are in essence* ***designed to prove optimality*** *[...] Moreover, tree search has an exponential behavior which makes it* ***not scalable*** *faced with real-world combinatorial problems inducing millions of binary decisions. ”*

Conventional wisdom - about Tree Search

Mathematical Programming Solver based on Local Search ([1]):

> *“ Tree search approaches like branch-and-bound are in essence* ***designed to prove optimality*** *[...] Moreover, tree search has an exponential behavior which makes it* ***not scalable*** *faced with real-world combinatorial problems inducing millions of binary decisions. ”*

We believe it is false considering **anytime tree searches**

Anytime tree searches - some (trivial) definitions and facts

Tree searches: usually constructive algorithms (explore a tree)

Anytime tree searches - some (trivial) definitions and facts

Tree searches: usually constructive algorithms (explore a tree)

Anytime: find quickly good solutions, then later try to improve them (similar to meta-heuristics)

Anytime tree searches - some (trivial) definitions and facts

Tree searches: usually constructive algorithms (explore a tree)

Anytime: find quickly good solutions, then later try to improve them (similar to meta-heuristics)

Many presented in AI/planning conferences

- Some famous ones: **LDS, Beam Search, wA* ...**
- Some recent: **Anytime pack search, Anytime Focal Search**

Anytime tree searches - some (trivial) definitions and facts

Tree searches: usually constructive algorithms (explore a tree)

Anytime: find quickly good solutions, then later try to improve them (similar to meta-heuristics)

Many presented in AI/planning conferences

- Some famous ones: **LDS, Beam Search, wA* ...**
- Some recent: **Anytime pack search, Anytime Focal Search**

Similar in purpose and definitions to meta-heuristics

Still not used much compared to classical meta-heuristics

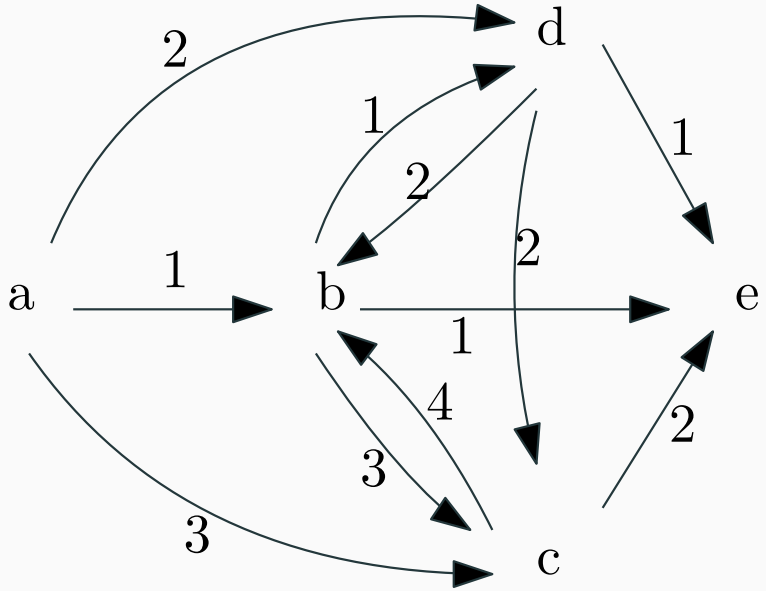
Our experiment

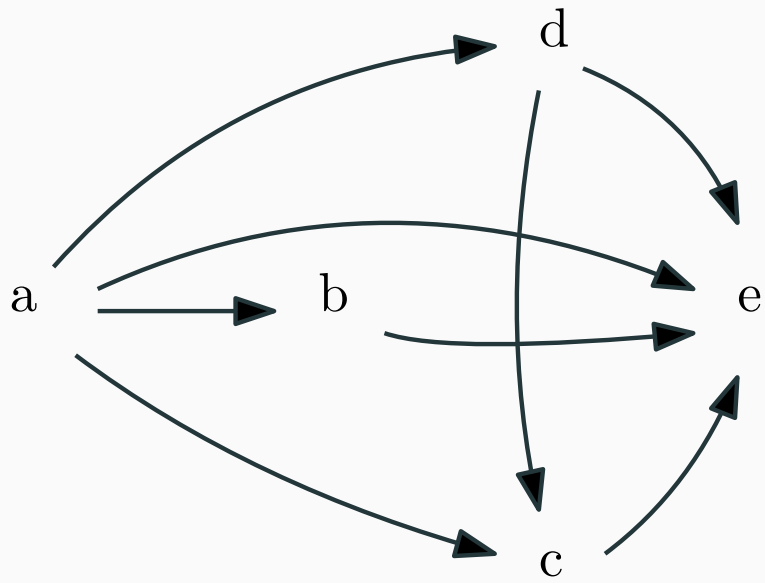
- We consider a well known benchmark (SOP)
- Apply anytime tree searches

Sequential Ordering Problem

SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints





The benchmark: SOPLIB

- Standard benchmark, proposed in 2006 (“large” instances)

The benchmark: SOPLIB

- Standard benchmark, proposed in 2006 (“large” instances)
- Some instances are almost precedence free
- Some are heavily constrained
- “in the middle” instances remain open (7 instances)

Literature

Many methods implemented during the 30 last years to solve SOP

Exact methods:

- Branch and cuts
- Decision diagrams + CP
- Branch & Bounds with advanced bounds/prunings

Literature

Many methods implemented during the 30 last years to solve SOP

Exact methods:
- Branch and cuts
- Decision diagrams + CP
- Branch & Bounds with advanced bounds/prunings

Meta-heuristics:
- Local search (3-opt)
- Ant Colony Optimization (using a 3-opt move)
- others (GA, ABC, parallel roll-out, LKH ...)

Literature

Many methods implemented during the 30 last years to solve SOP

Exact methods:
- Branch and cuts
- Decision diagrams + CP
- Branch & Bounds with advanced bounds/prunings

Meta-heuristics:
- Local search (3-opt)
- Ant Colony Optimization (using a 3-opt move)
- others (GA, ABC, parallel roll-out, LKH ...)

- Exact methods tend to build stronger bounds
- meta-heuristics strongly rely on 3-opt (local search)

Our anytime Branch-and-Bound

Tree Search

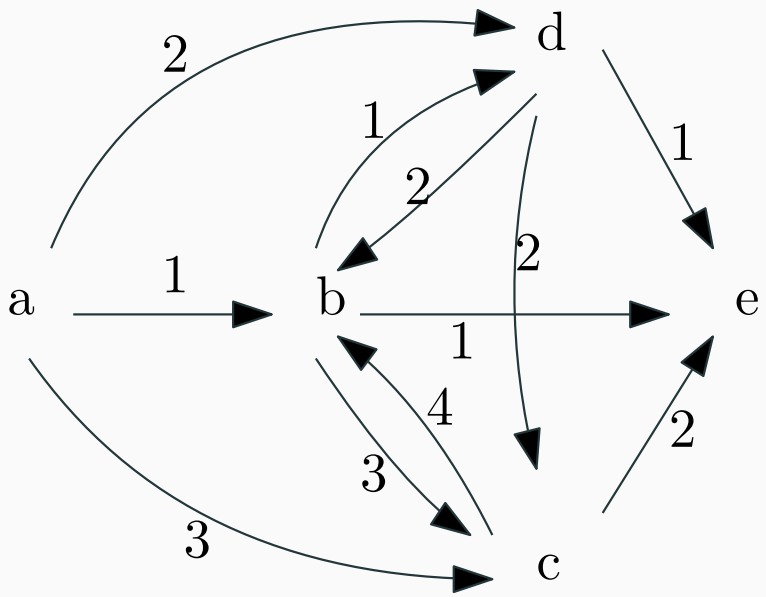
Two parts:

Implicit tree: how to branch, bounds ...

Search strategy: DFS, best-first, Beam Search ...

Implicit tree - Branching

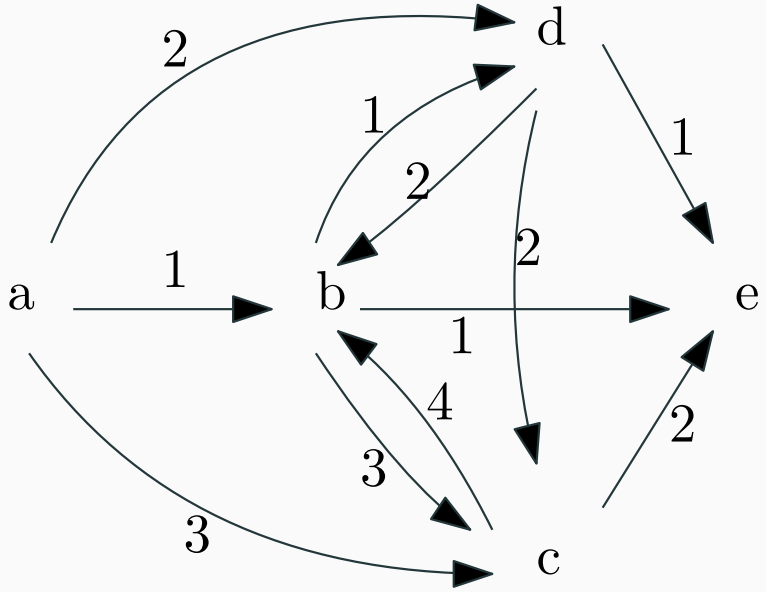
Forward branching + Prefix bounds

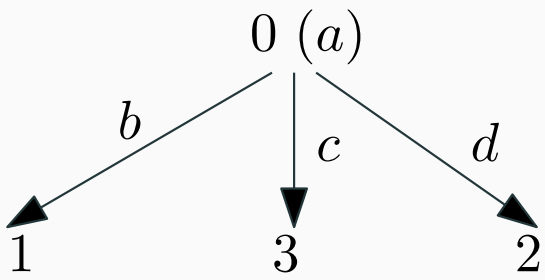


$0\ (a)$

Implicit tree - Branching

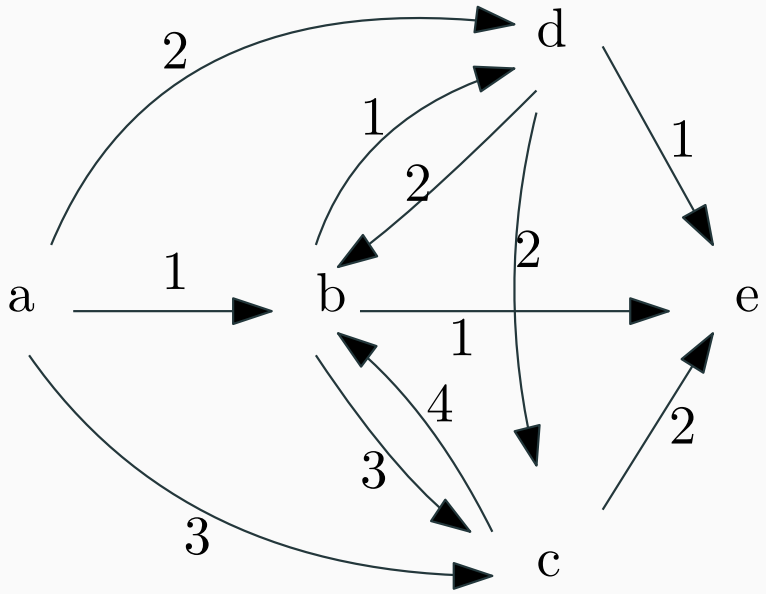
Forward branching + Prefix bounds

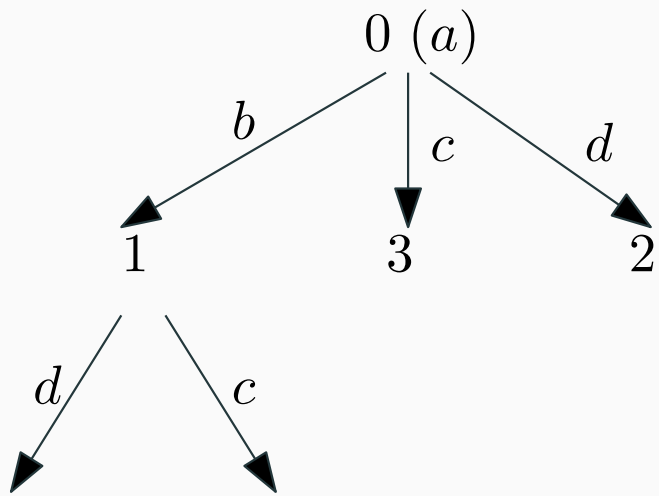




Implicit tree - Branching

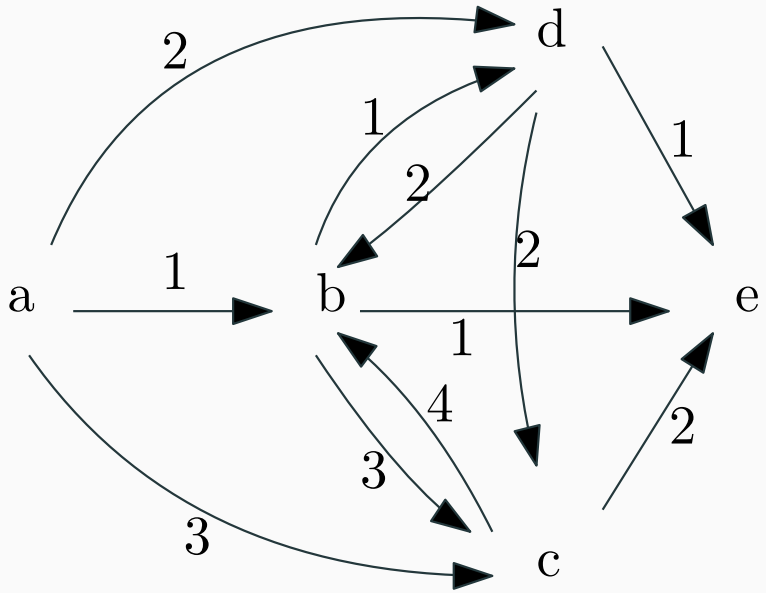
Forward branching + Prefix bounds

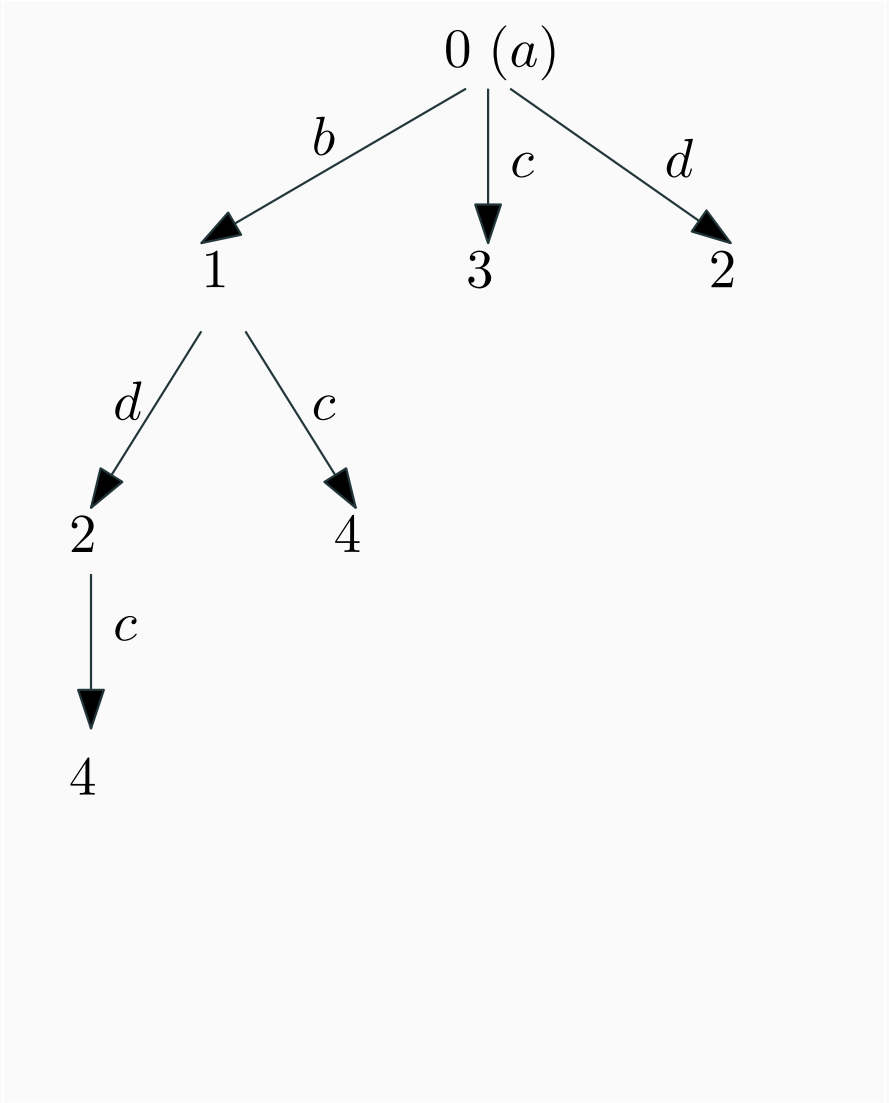




Implicit tree - Branching

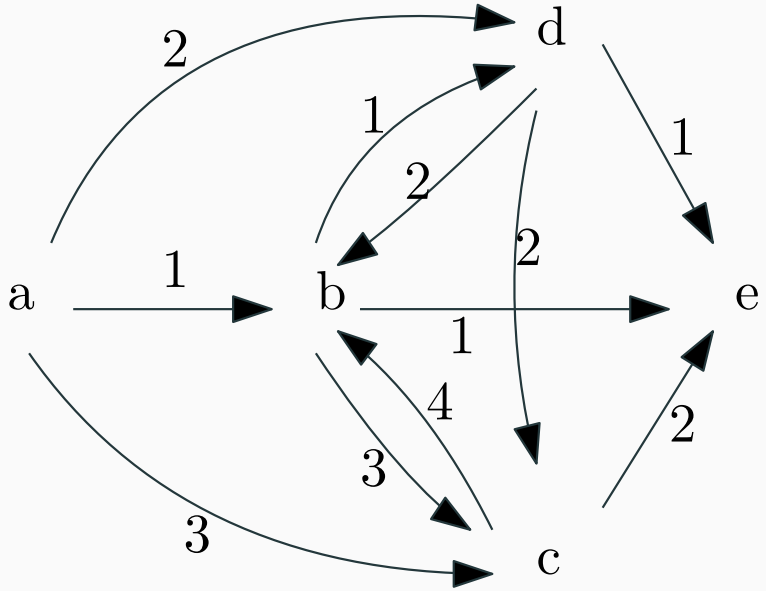
Forward branching + Prefix bounds

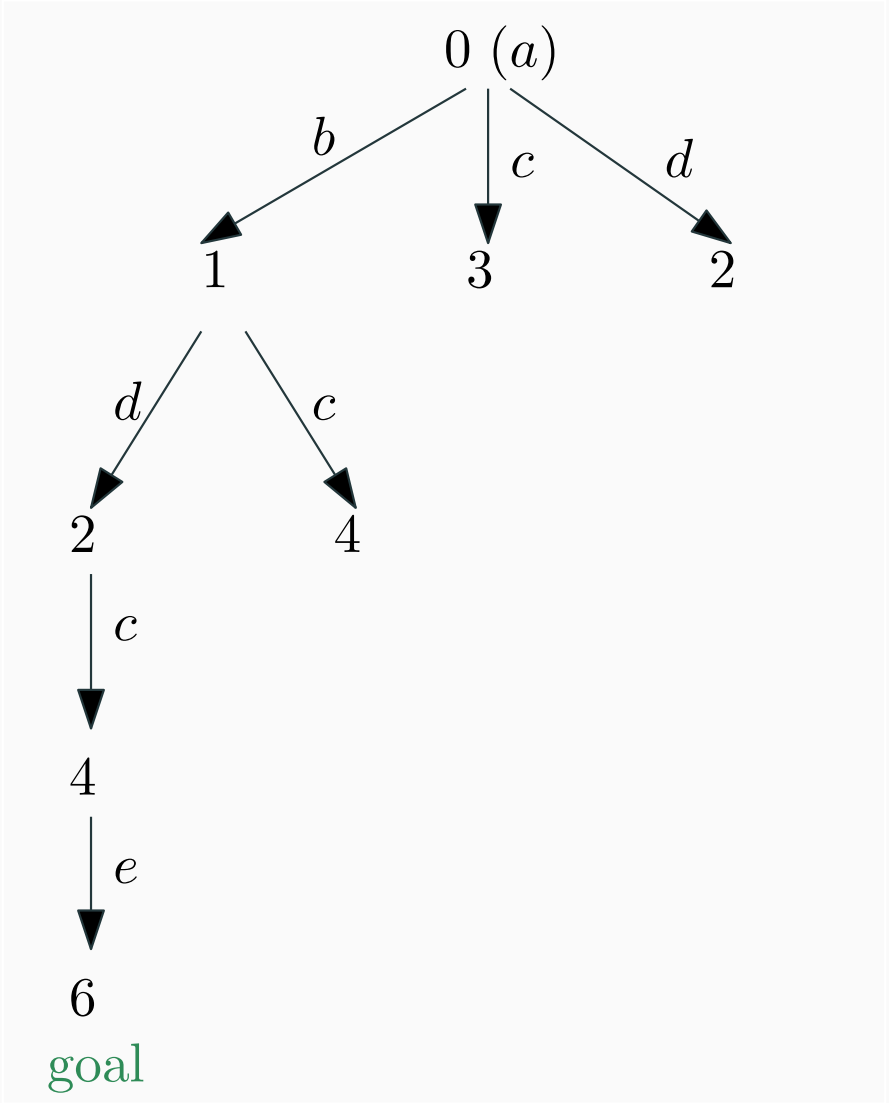




Implicit tree - Branching

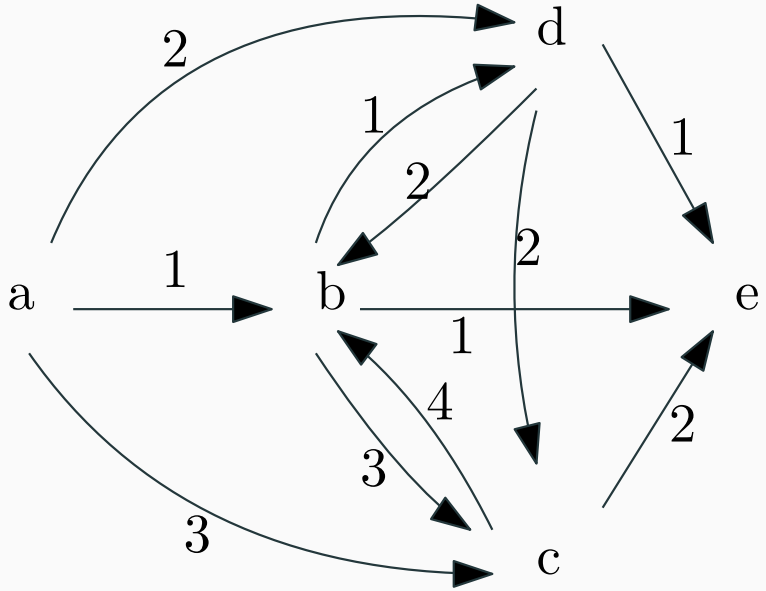
Forward branching + Prefix bounds

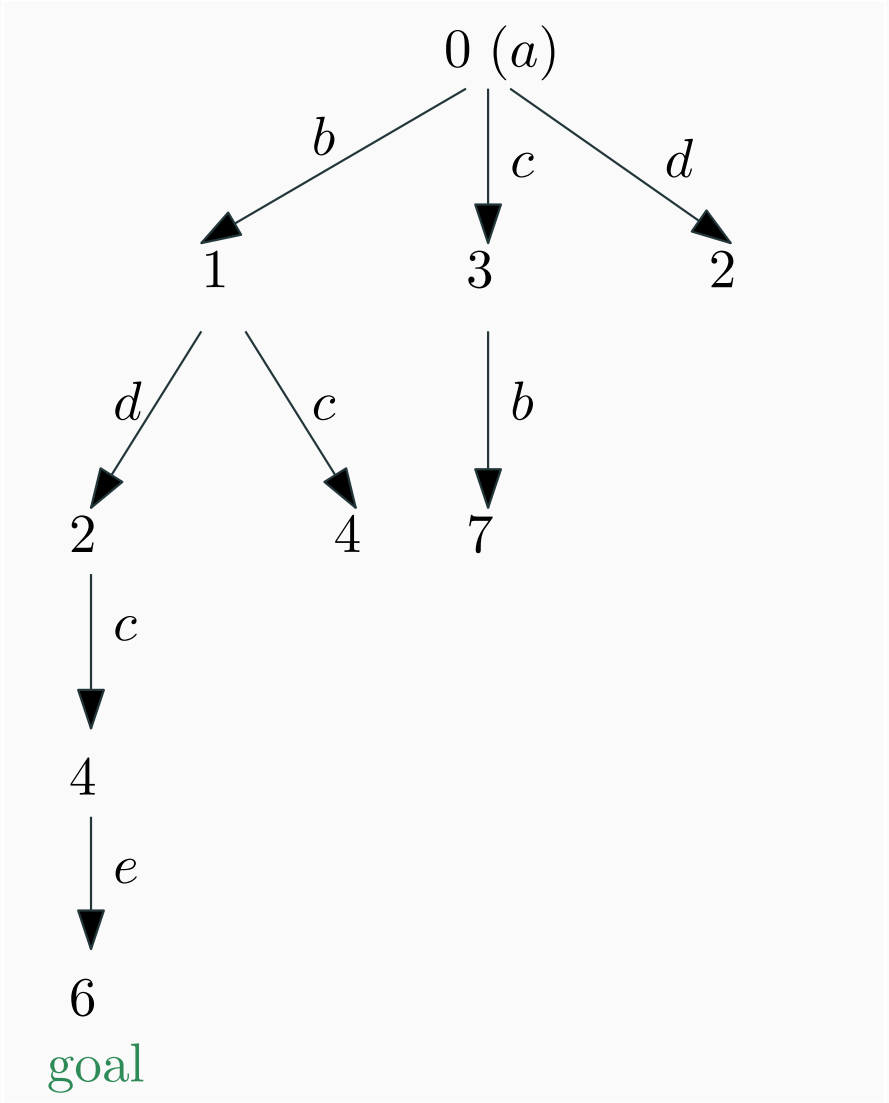




Implicit tree - Branching

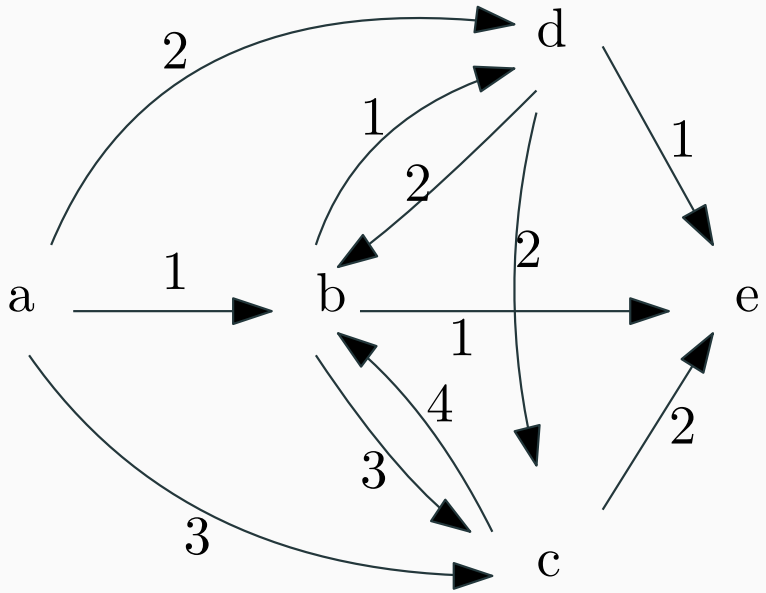
Forward branching + Prefix bounds

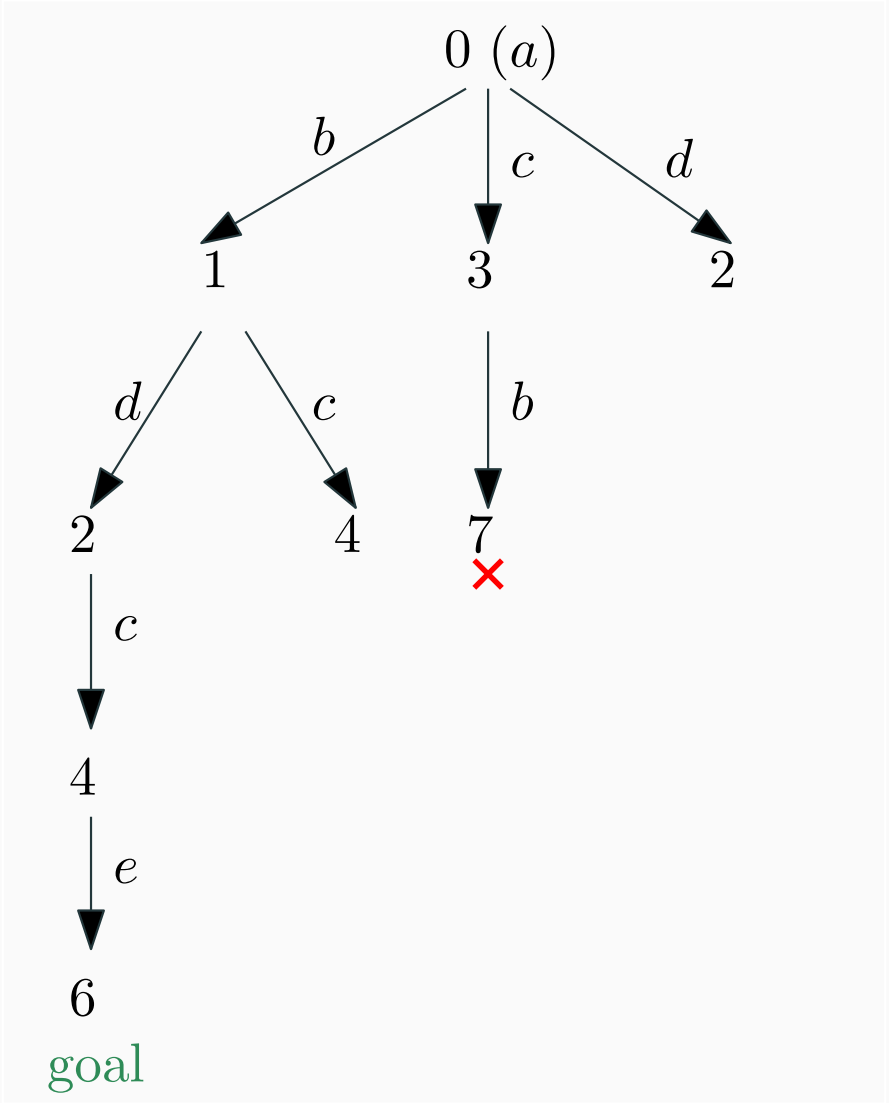




Implicit tree - Branching

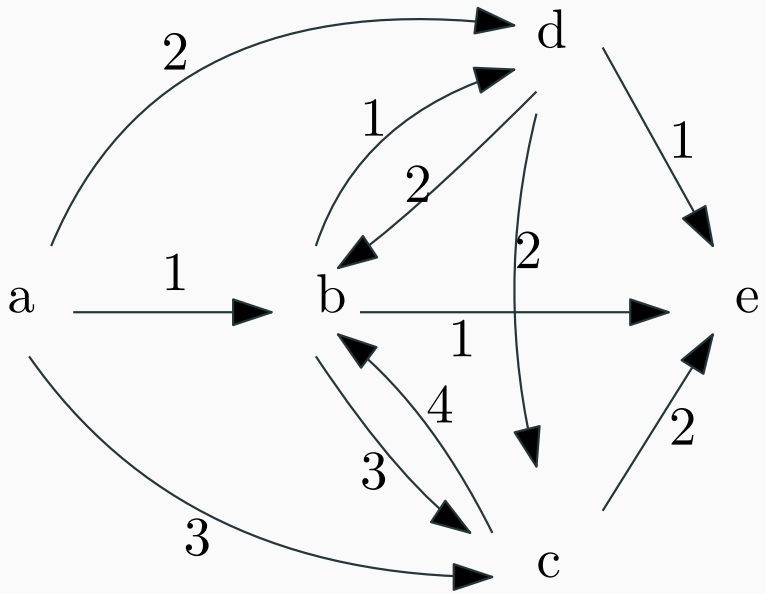
Forward branching + Prefix bounds

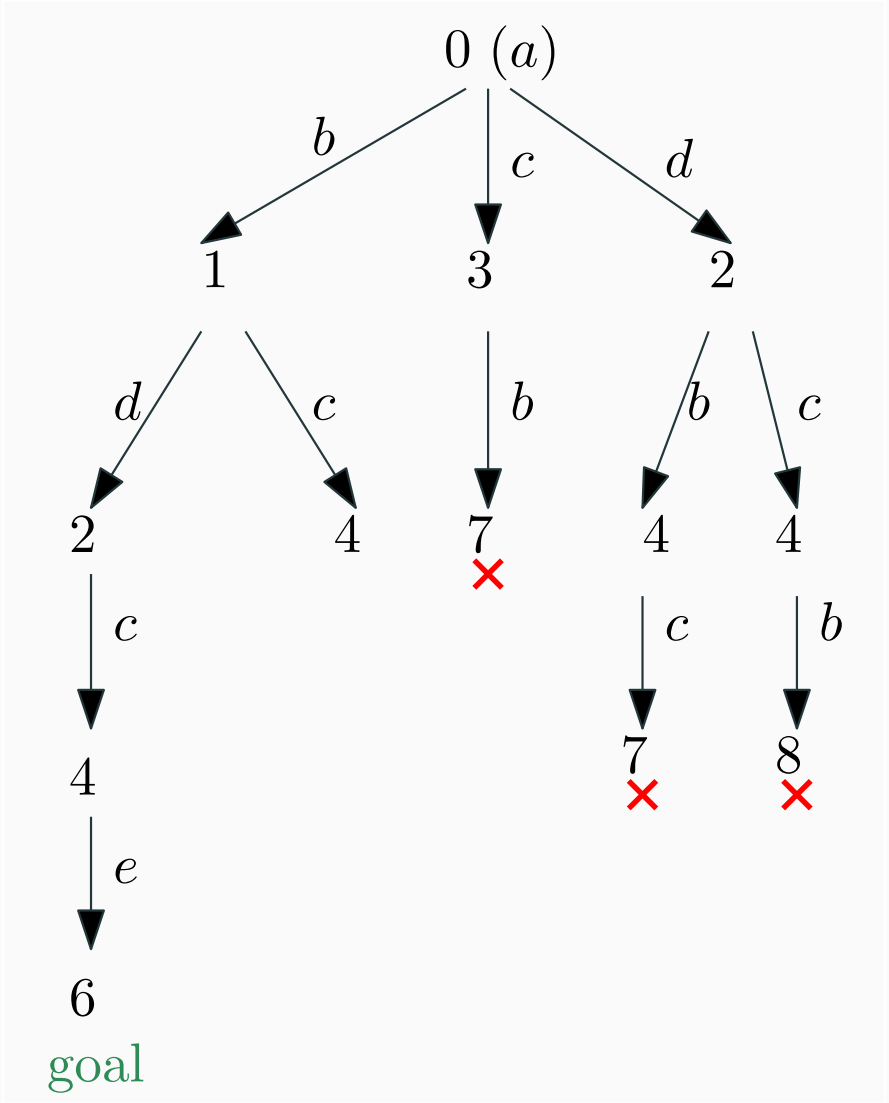




Implicit tree - Branching

Forward branching + Prefix bounds





Search

- **DFS** classical in Branch-and-bounds. Usually stuck in early sub-optimal choices

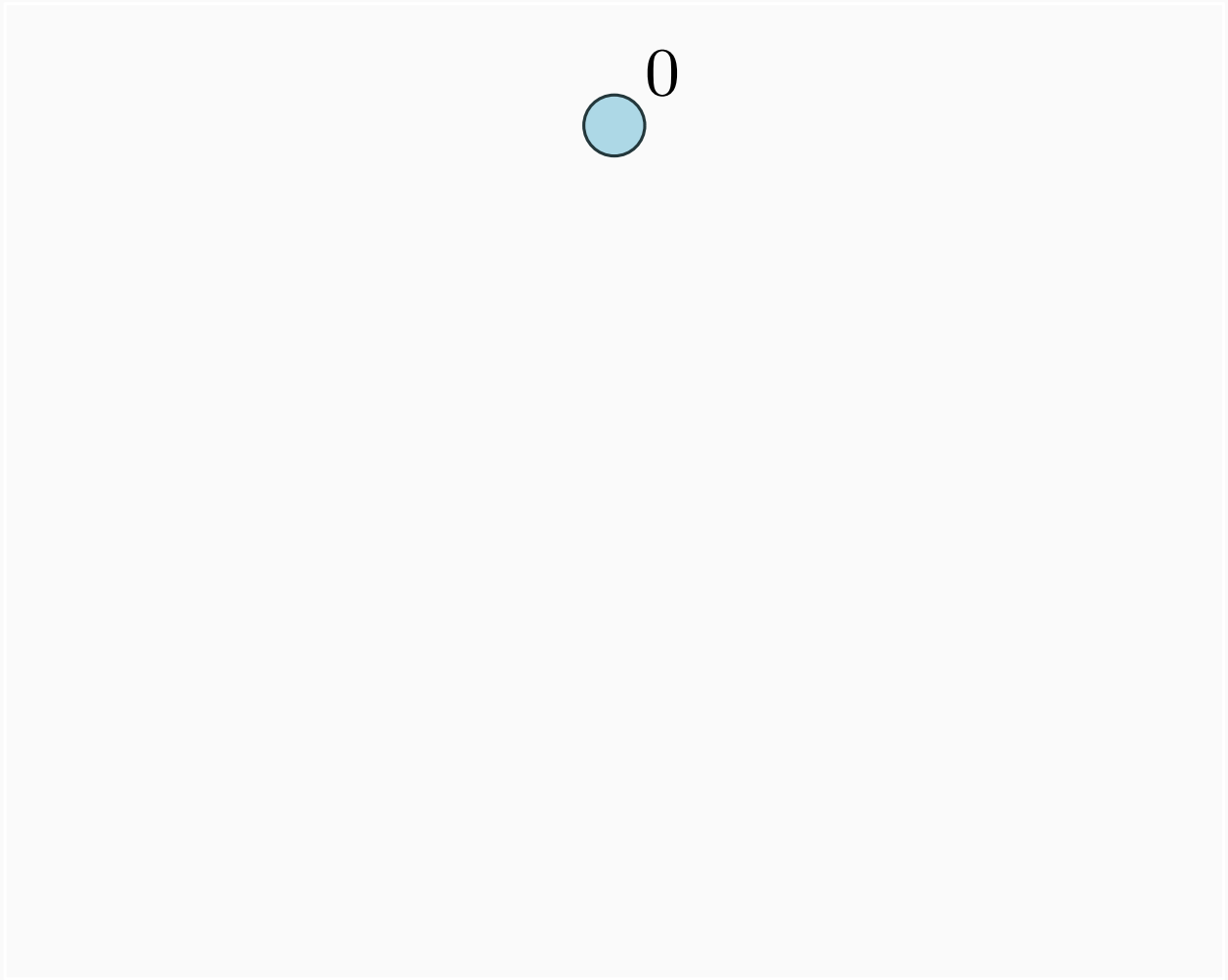
Search

- **DFS** classical in Branch-and-bounds. Usually stuck in early sub-optimal choices
- **LDS** allows DFS to recover from early sub-optimal choices

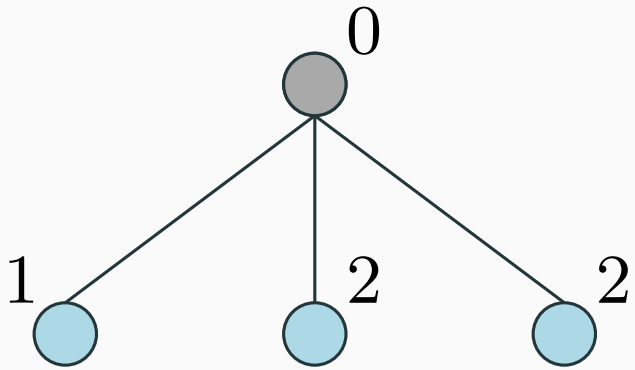
Search

- **DFS** classical in Branch-and-bounds. Usually stuck in early sub-optimal choices
- **LDS** allows DFS to recover from early sub-optimal choices
- **Iterative Beam Search** (next slide)

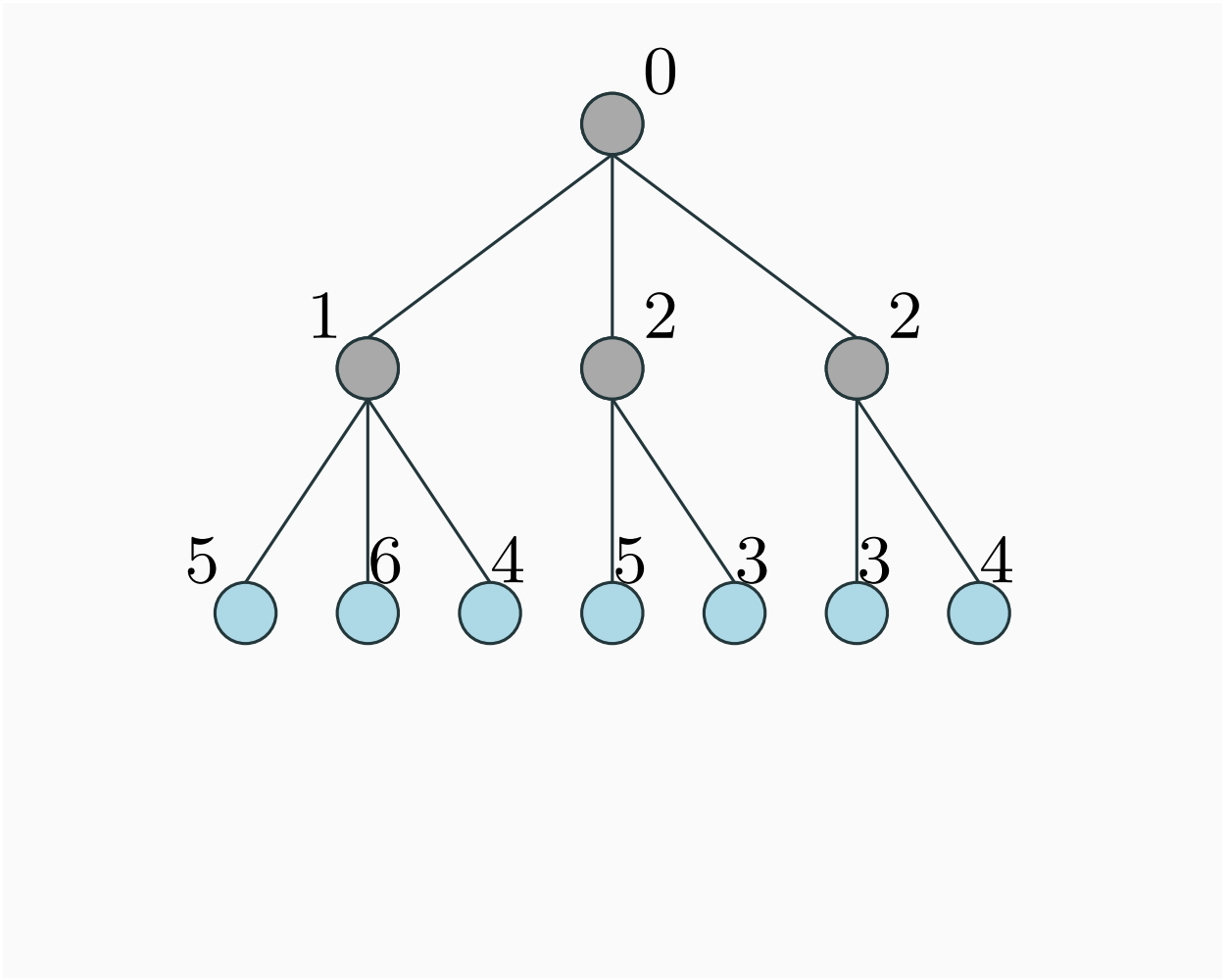
Beam Search ($D = 3$)



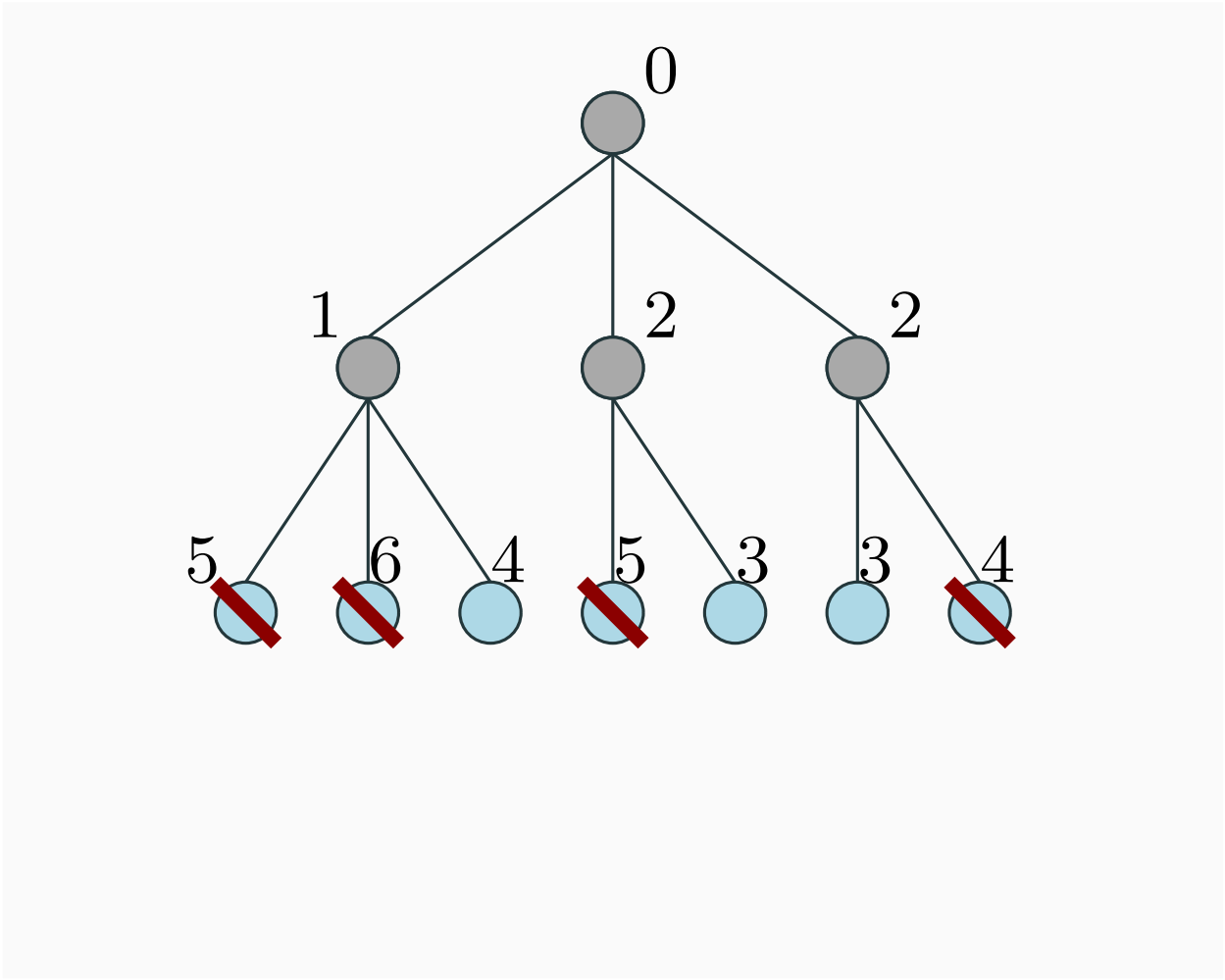
Beam Search ($D = 3$)



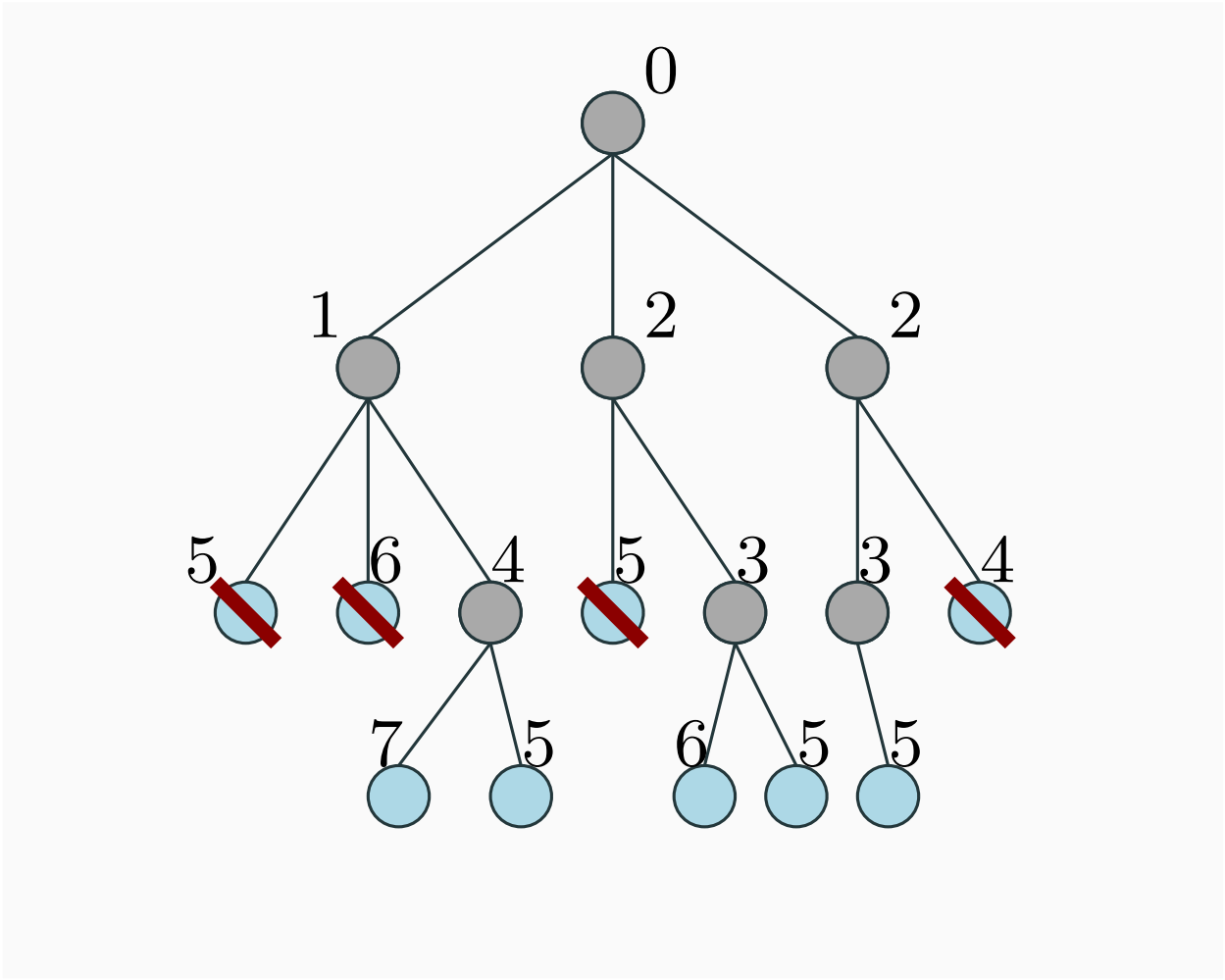
Beam Search ($D = 3$)



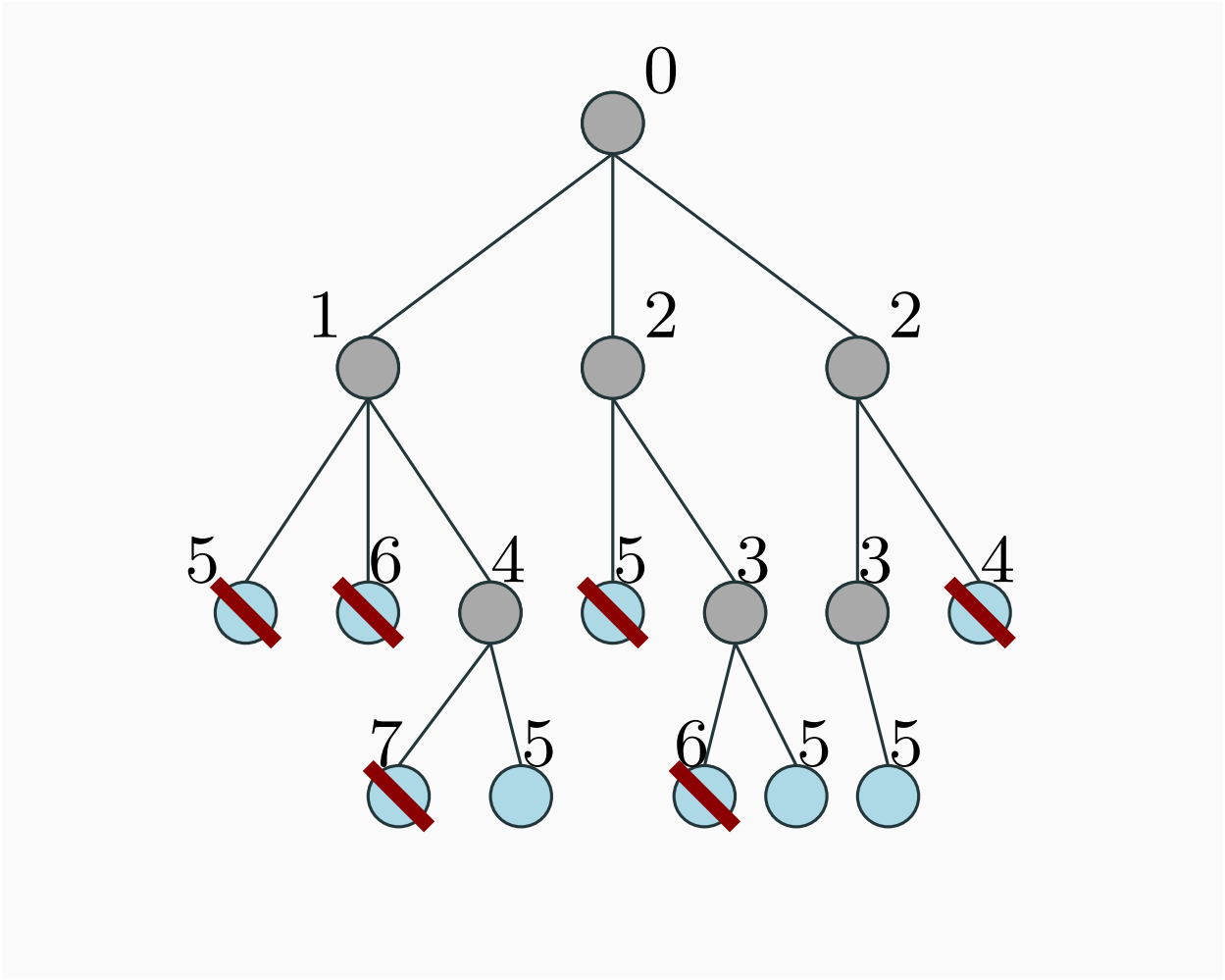
Beam Search ($D = 3$)



Beam Search ($D = 3$)



Beam Search ($D = 3$)



Iterative Beam Search

- Runs a beam of size 1 (greedy)
- Then runs a beam of size 2, then 4, then 8 ...

Stops when no heuristic fathoming is done (proves optimality)

Dynamic Programming inspired prunings

Example, two equivalent partial solutions:

1. **a,b,c,d** cost 10
2. **a,c,b,d** cost 12

Dynamic Programming inspired prunings

Example, two equivalent partial solutions:

1. **a,b,c,d** cost 10
2. **a,c,b,d** cost 12

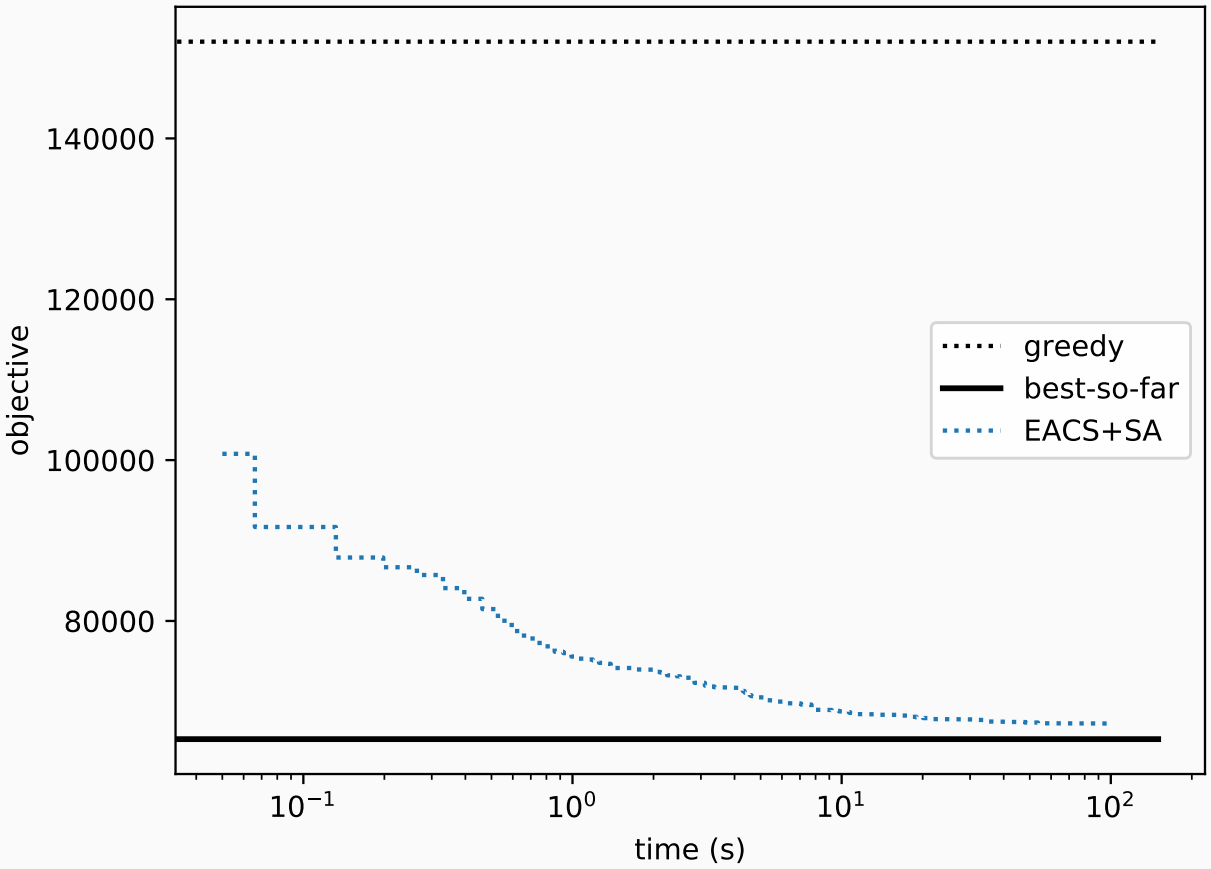
Discard (2) as it is “dominated” by (1).

Numerical Results

Results - Performance profiles on R.700.1000.15

state-of-the-art:

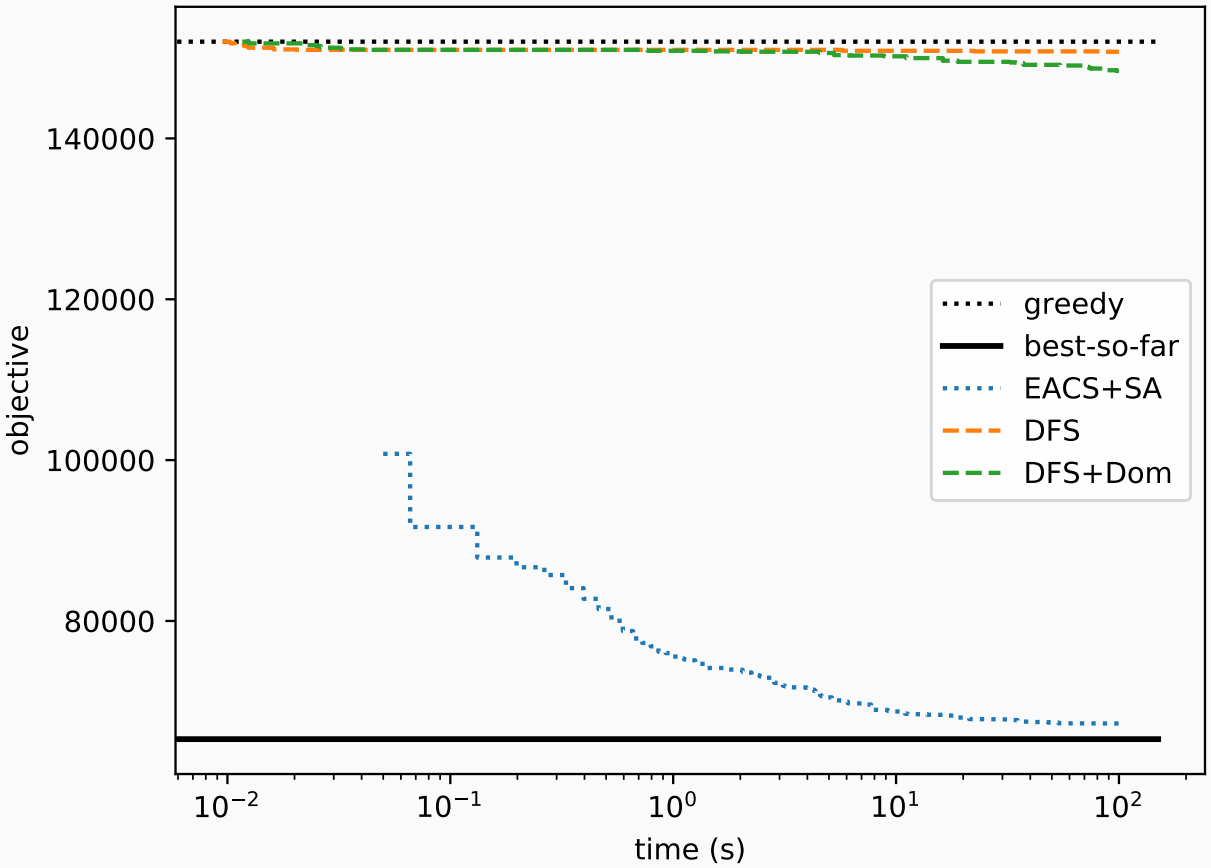
- Enhanced Ant Colony System and Simulated Annealing (EACS+SA)
- best-so-far LKH3 with 100.000 seconds run (≈ 27h)



Results - Performance profiles on R.700.1000.15

state-of-the-art:

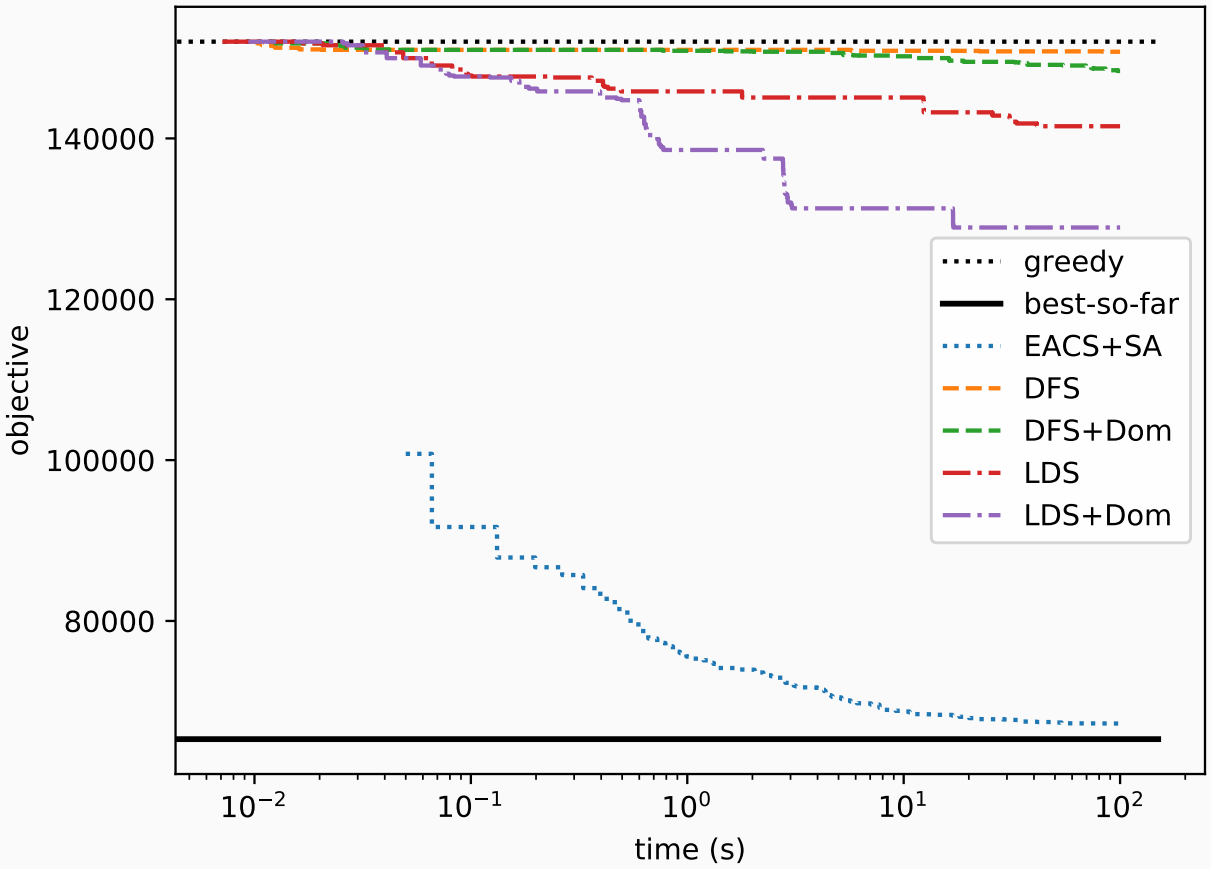
- Enhanced Ant Colony System and Simulated Annealing (EACS+SA)
- best-so-far LKH3 with 100.000 seconds run (≈ 27h)



Results - Performance profiles on R.700.1000.15

state-of-the-art:

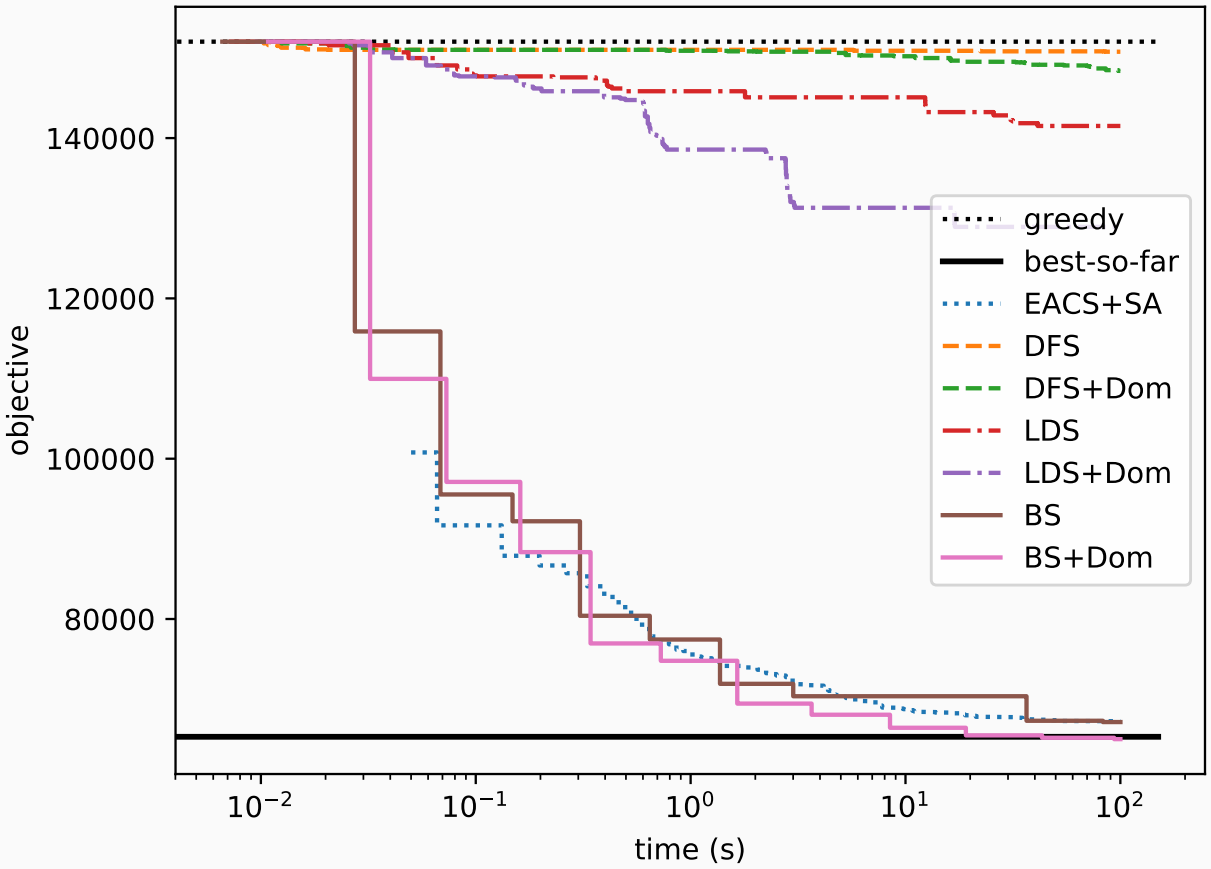
- Enhanced Ant Colony System and Simulated Annealing (EACS+SA)
- best-so-far LKH3 with 100.000 seconds run (≈ 27h)



Results - Performance profiles on R.700.1000.15

state-of-the-art:

- Enhanced Ant Colony System and Simulated Annealing (EACS+SA)
- best-so-far LKH3 with 100.000 seconds run (≈ 27h)



Results - New best-so-far solutions

6 over 7 new-best-so-far solutions
(the other one is probably optimal)

Instance	best known	BS+PE (600s)
R.500.100.15	5.284	**5.261**
R.500.1000.15	49.504	**49.366**
R.600.100.15	5.472	**5.469**
R.600.1000.15	55.213	**54.994**
R.700.100.15	7.021	**7.020**
R.700.1000.15	65.305	**64.777**

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: large search space and poor guidance

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: large search space and poor guidance

15% precedence constraints: 5 children in average

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: large search space and poor guidance

15% precedence constraints: 5 children in average

30% ,60% precedence constraints: proves optimality in a few milliseconds.

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: large search space and poor guidance

15% precedence constraints: 5 children in average

30% ,60% precedence constraints: proves optimality in a few milliseconds.

The SOPLIB mainly contains heavily constrained instances:

- hard for MIPs and local searches
- but (relatively) easy for constructive algorithms
- **thus the need to consider anytime tree searches**

Wrapping-up

Conclusions

- At least on the SOPLIB, anytime tree searches outperform classical meta-heuristics

Conclusions

- At least on the SOPLIB, anytime tree searches outperform classical meta-heuristics
- The search-strategy choice is crucial

Tree search algorithms for the sequential ordering problem

Luc Libralesso - Abdel-Malik Bouhassoun
Hadrien Cambazard - Vincent Jost

September 2 2020 - ECAI 2020

Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France
email: luc.libralesso@grenoble-inp.fr

References

[1] Frédéric Gardi, Thierry Benoist, Julien Darlay, Bertrand Estellon, and Romain Megel. *Mathematical programming solver based on local search*. Wiley Online Library, 2014.