Why branch-and-bounds are meta-heuristics

state-of-the-art heuristics from branch-and-bounds

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

February, 20, 2020 - ROADEF 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 problem

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

Conventional wisdom - Exact Methods and Metaheuristics

Two ways to solve a 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

why anytime tree searches are not used more?

two hypothesis:

1. They are not efficient?
2. They are underestimated?

We believe the latter is true

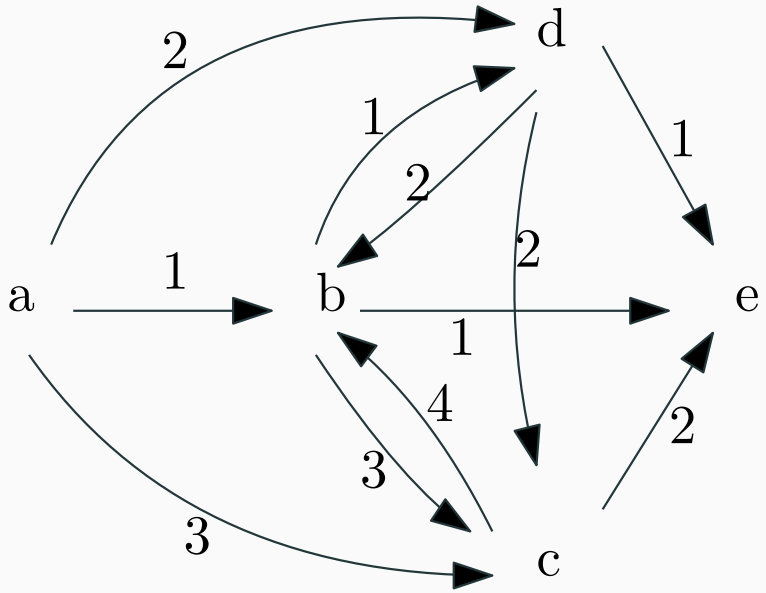
Our experiment

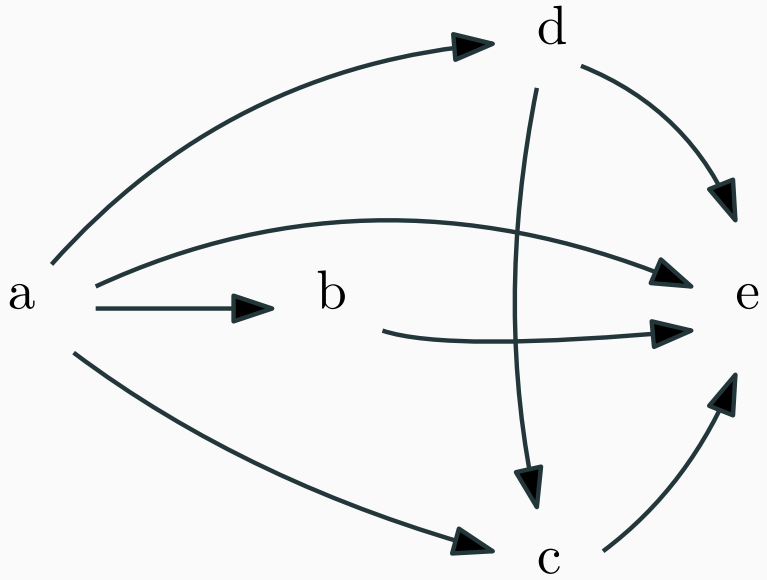
- We consider a well known benchmark (SOP)
- Apply anytime tree searches

Sequential Ordering Problem

SOP - problem definition

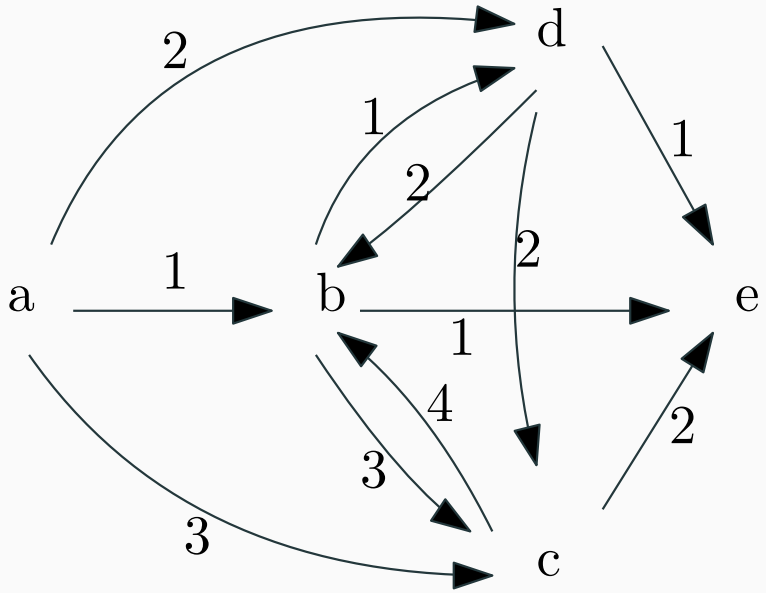
Asymmetric Traveling Salesman Problem with precedence constraints

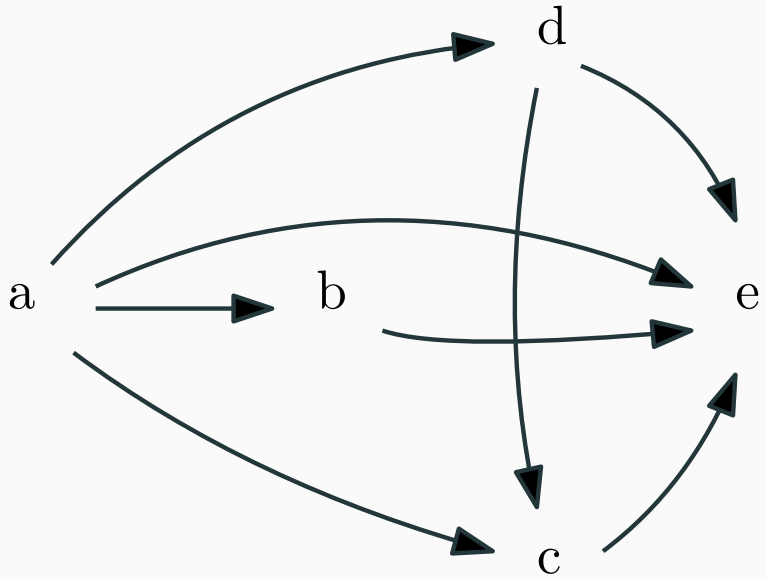




SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints

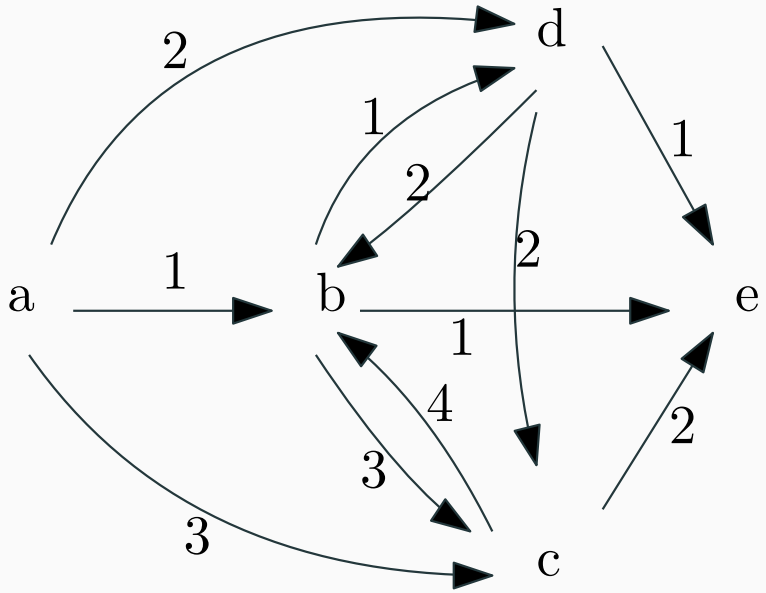


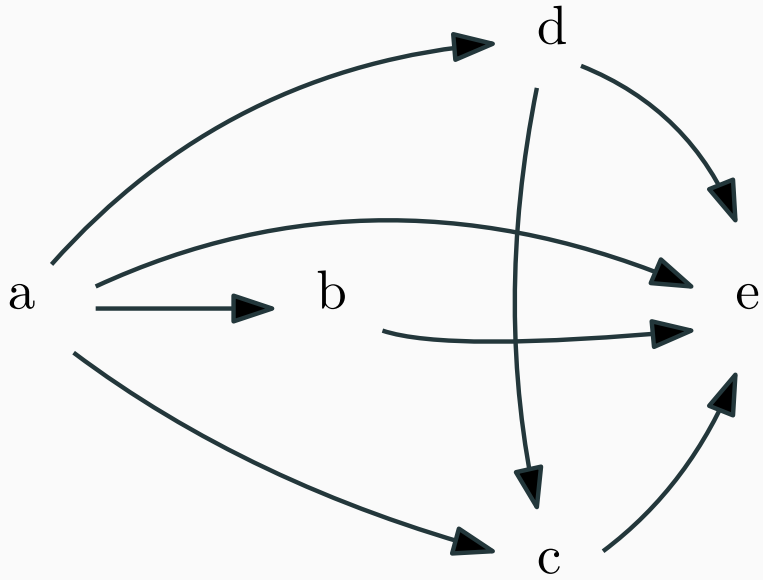


- **a,d,c,b,e** is a feasible and costs 9

SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints





- **a,d,c,b,e** is a feasible and costs 9
- **a,b,c,d,e** is not feasible

The benchmark: SOPLIB

- Published in 2006
- Standard for meta-heuristics

The benchmark: SOPLIB

- Published in 2006
- Standard for meta-heuristics

- “Large” instances (200 to 700 cities)
- Different densities (1, 15, 30, 60) % precedence constraints

The benchmark: SOPLIB

- Published in 2006
- Standard for meta-heuristics
- “Large” instances (200 to 700 cities)
- Different densities (1, 15, 30, 60) % precedence constraints
- 15% precedence-dense 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/fathomings

Meta-heuristics:

- Local searches (3-opt)
- Ant Colony Optimization
- Various heuristics (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/fathomings

Meta-heuristics:

- Local searches (3-opt)
- Ant Colony Optimization
- Various heuristics (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

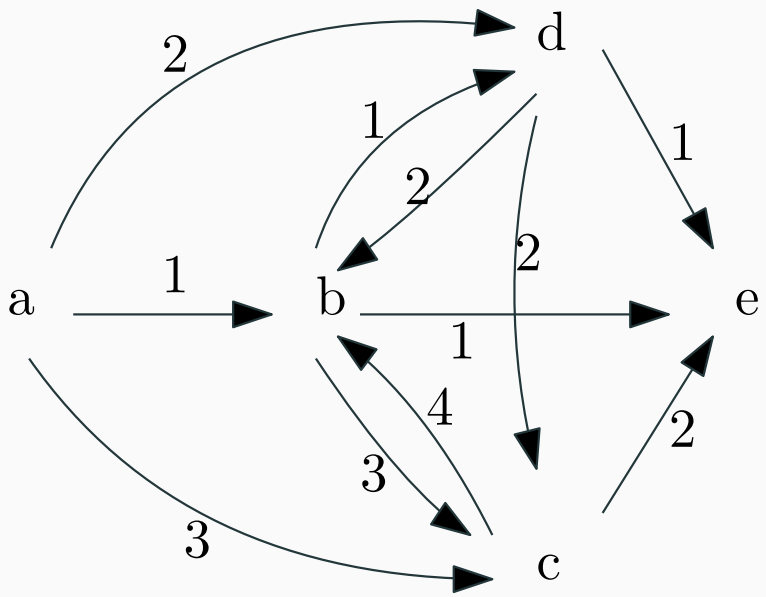
Two parts:

Implicit tree: how to branch, bounds ...

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

Implicit tree - Branching

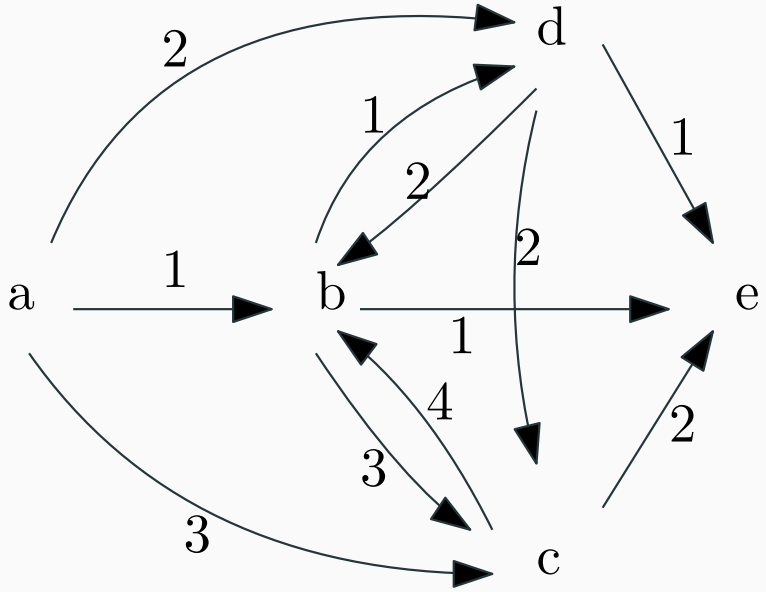
Forward branching + Prefix bounds

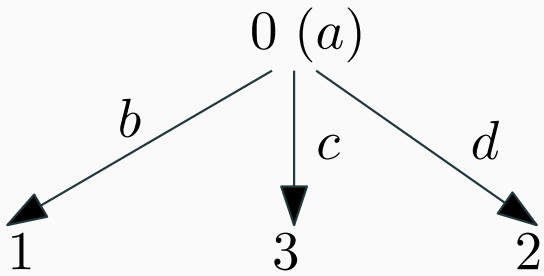


$0\ (a)$

Implicit tree - Branching

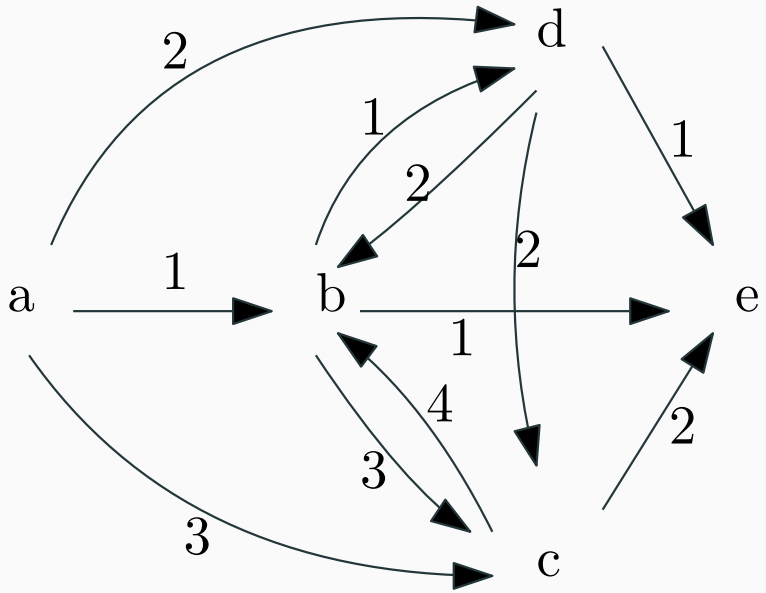
Forward branching + Prefix bounds

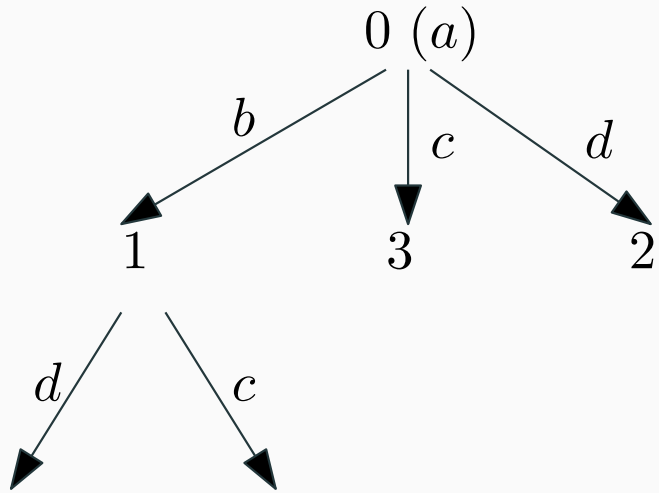




Implicit tree - Branching

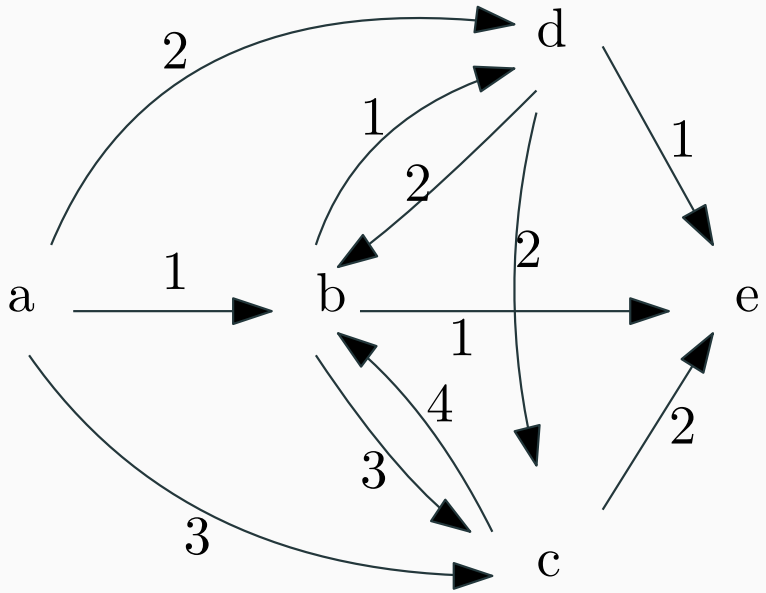
Forward branching + Prefix bounds

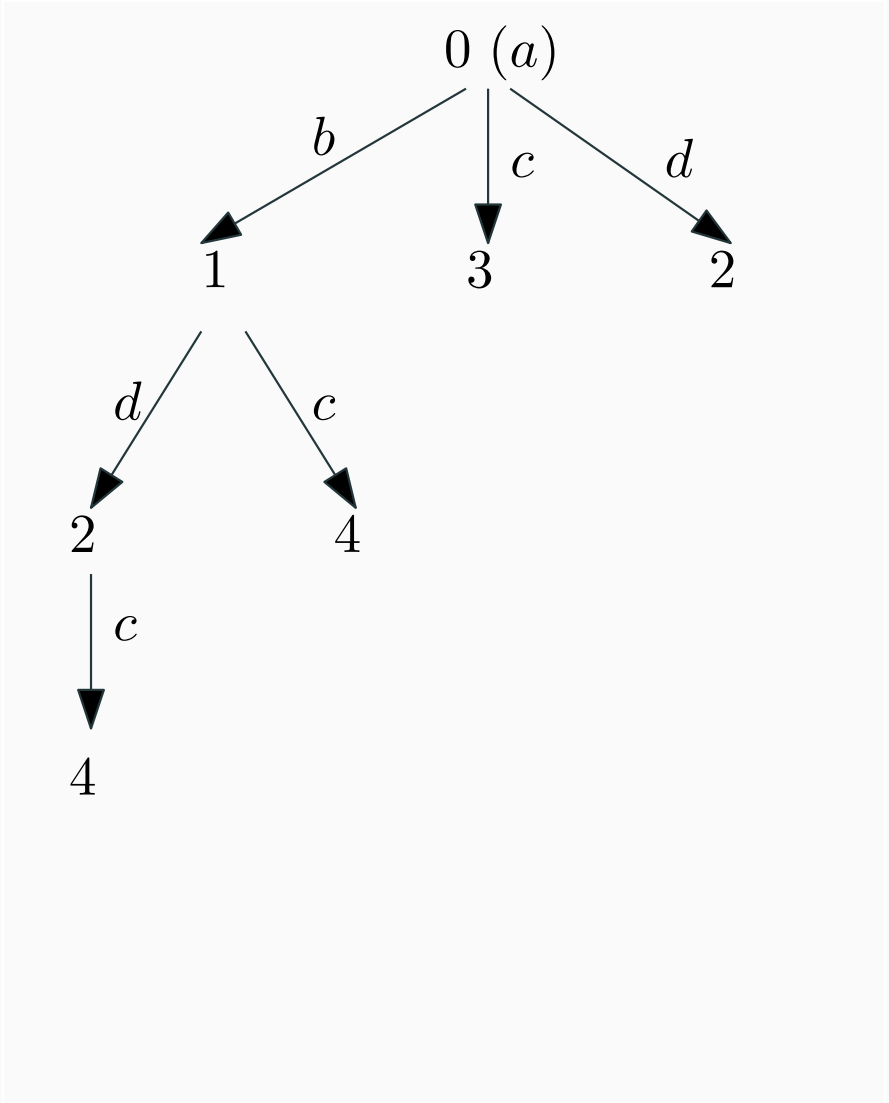




Implicit tree - Branching

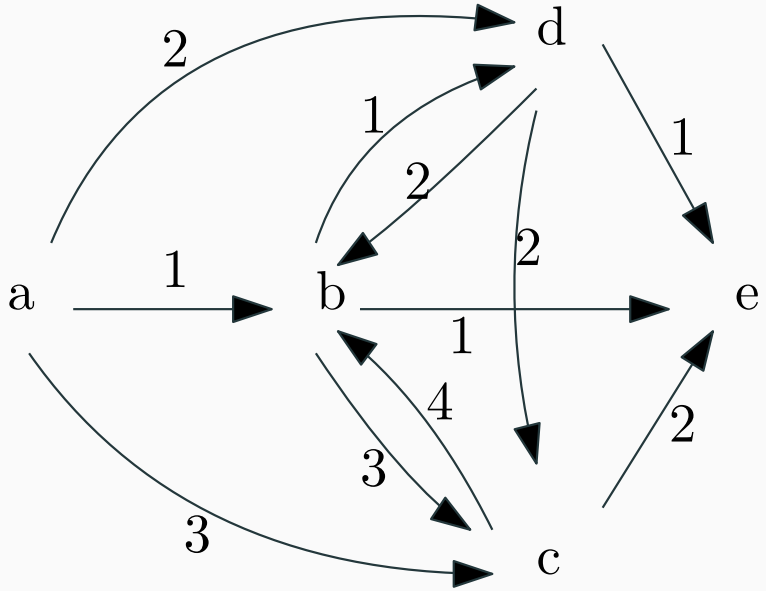
Forward branching + Prefix bounds

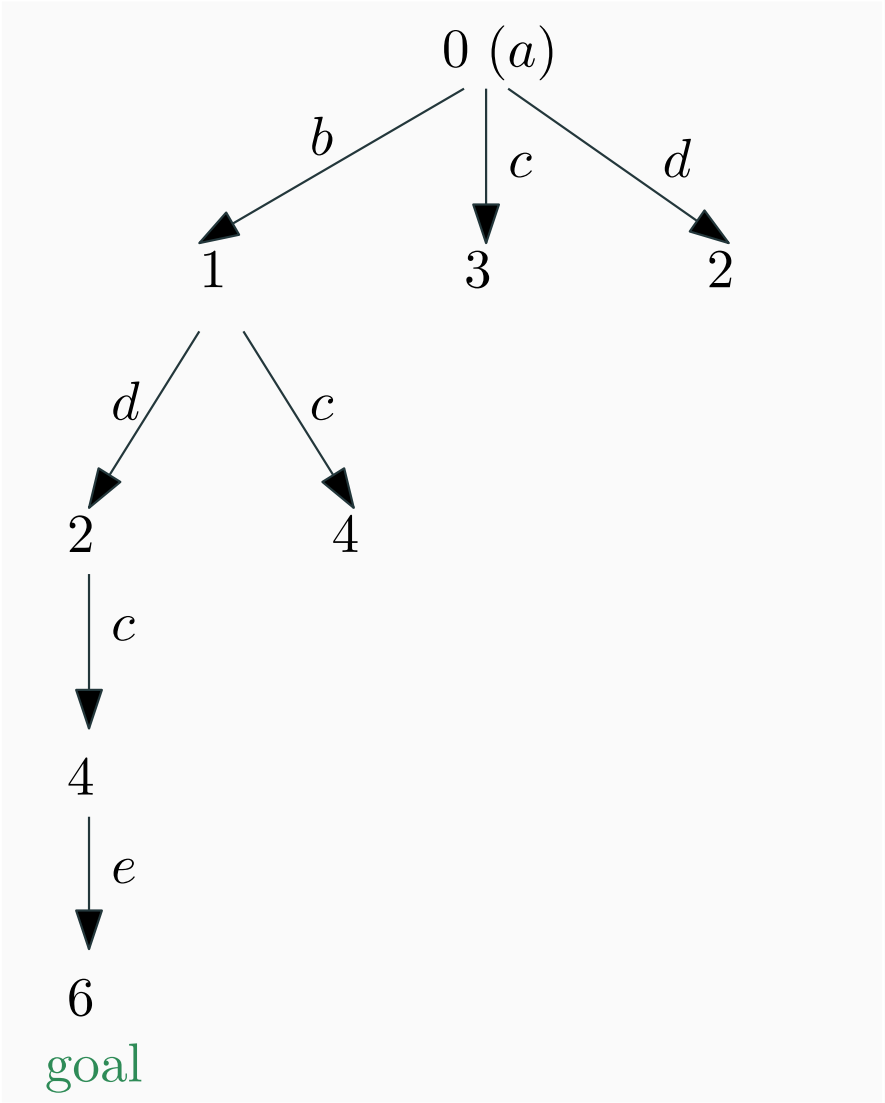




Implicit tree - Branching

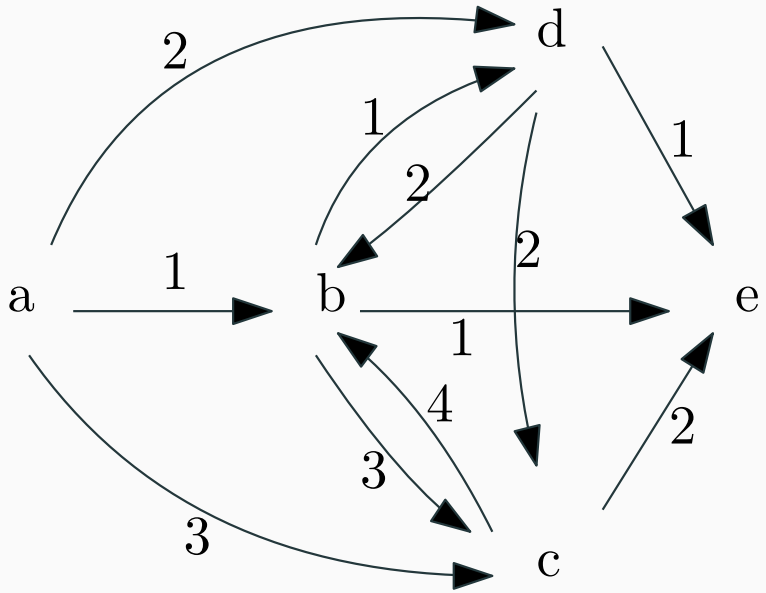
Forward branching + Prefix bounds

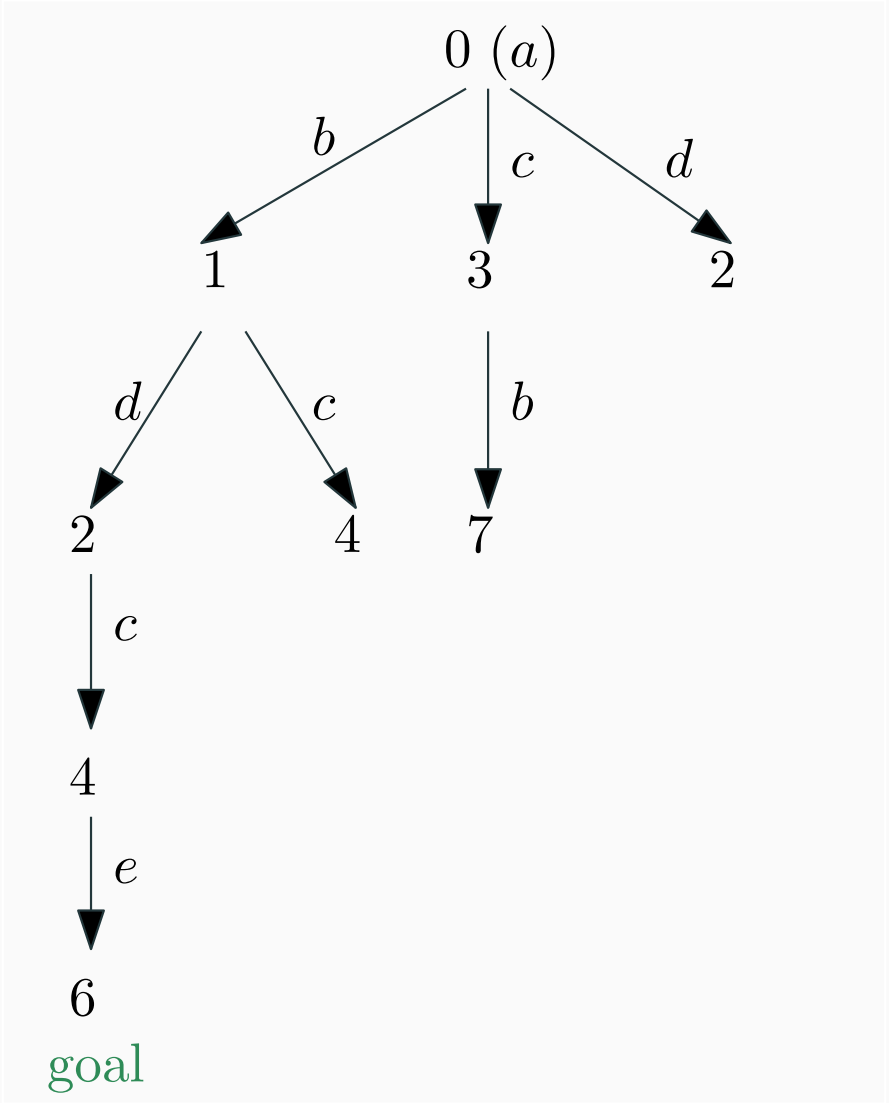




Implicit tree - Branching

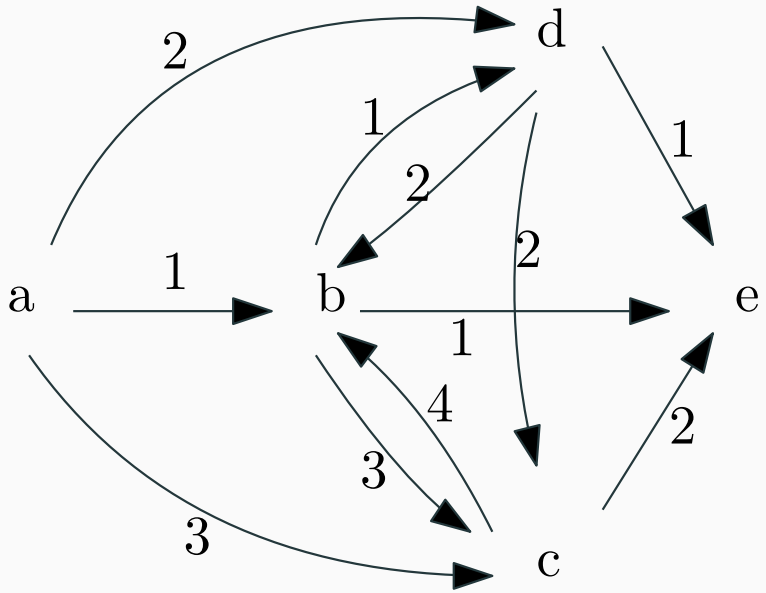
Forward branching + Prefix bounds

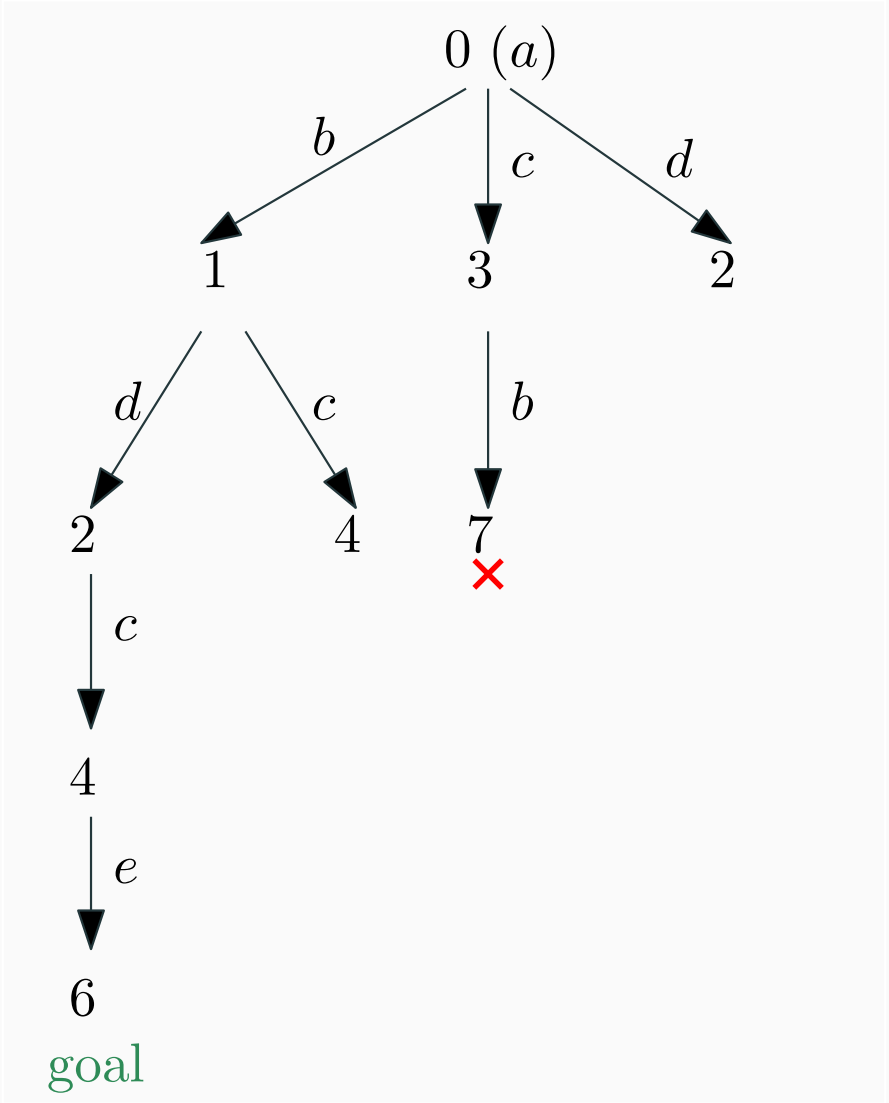




Implicit tree - Branching

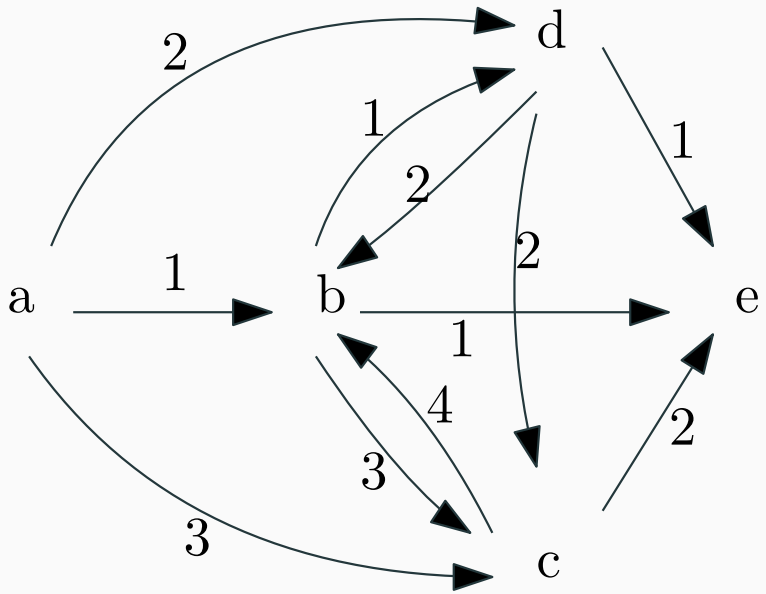
Forward branching + Prefix bounds

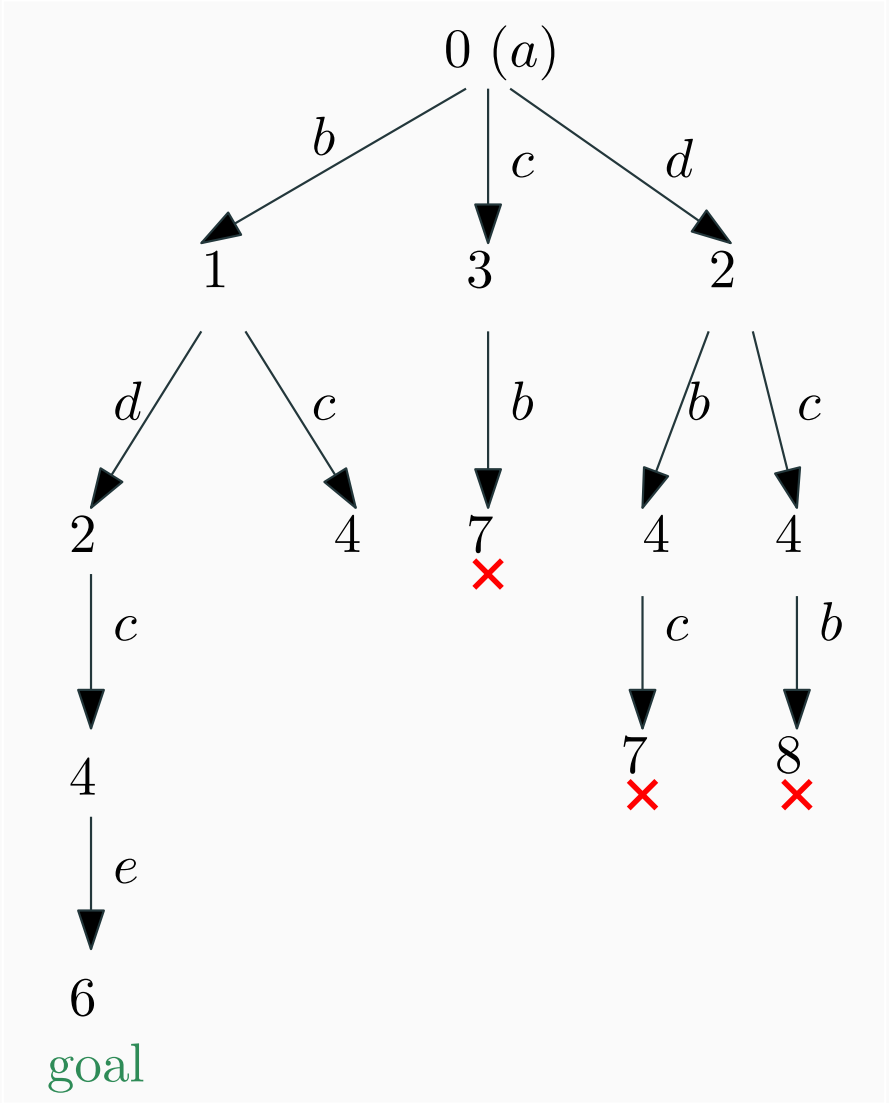




Implicit tree - Branching

Forward branching + Prefix bounds





Search

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

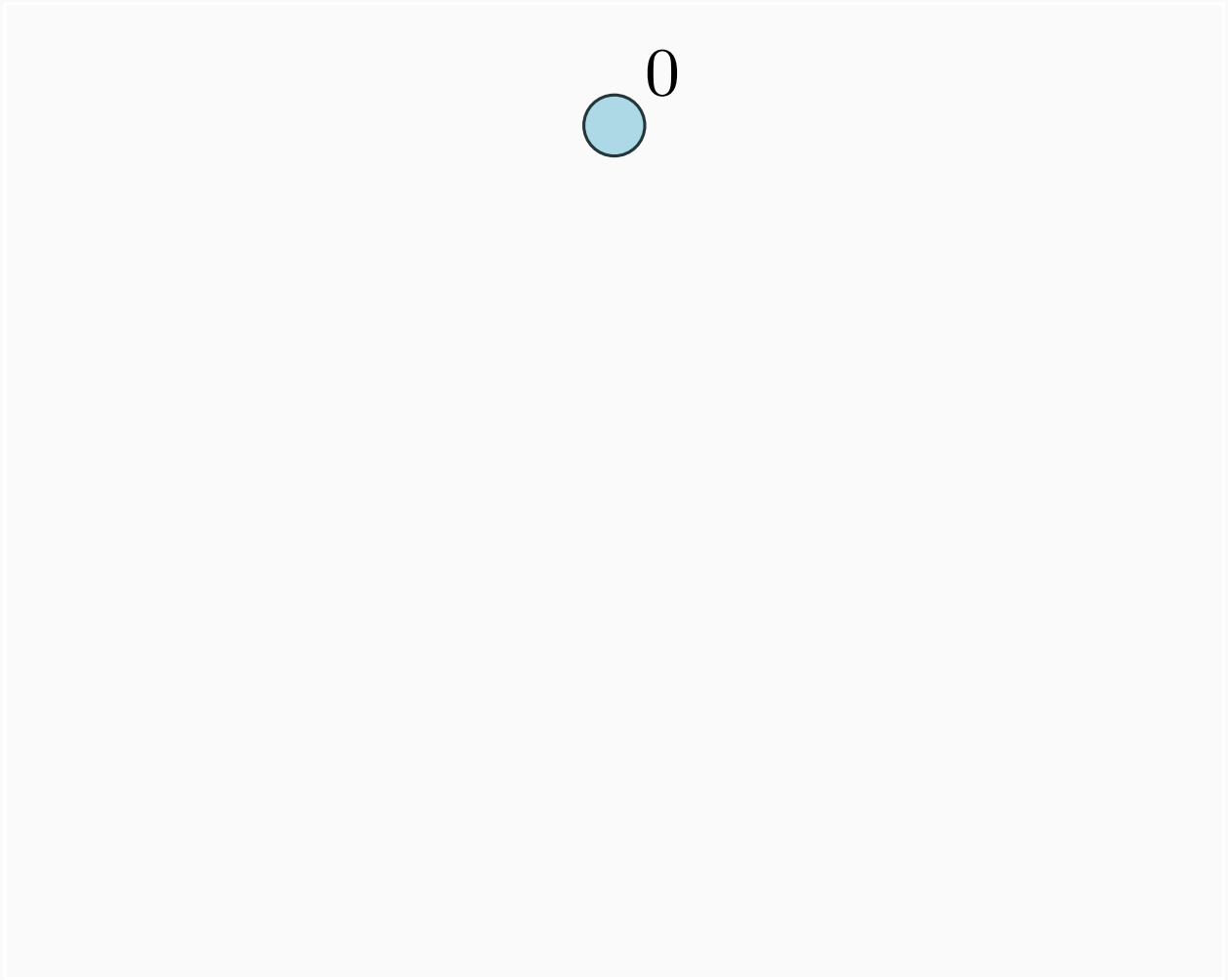
Search

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

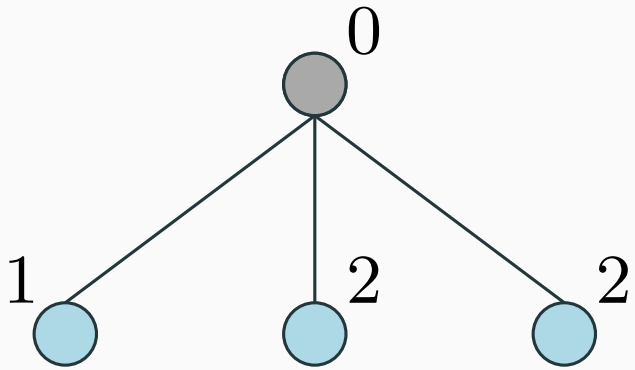
Search

- **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)

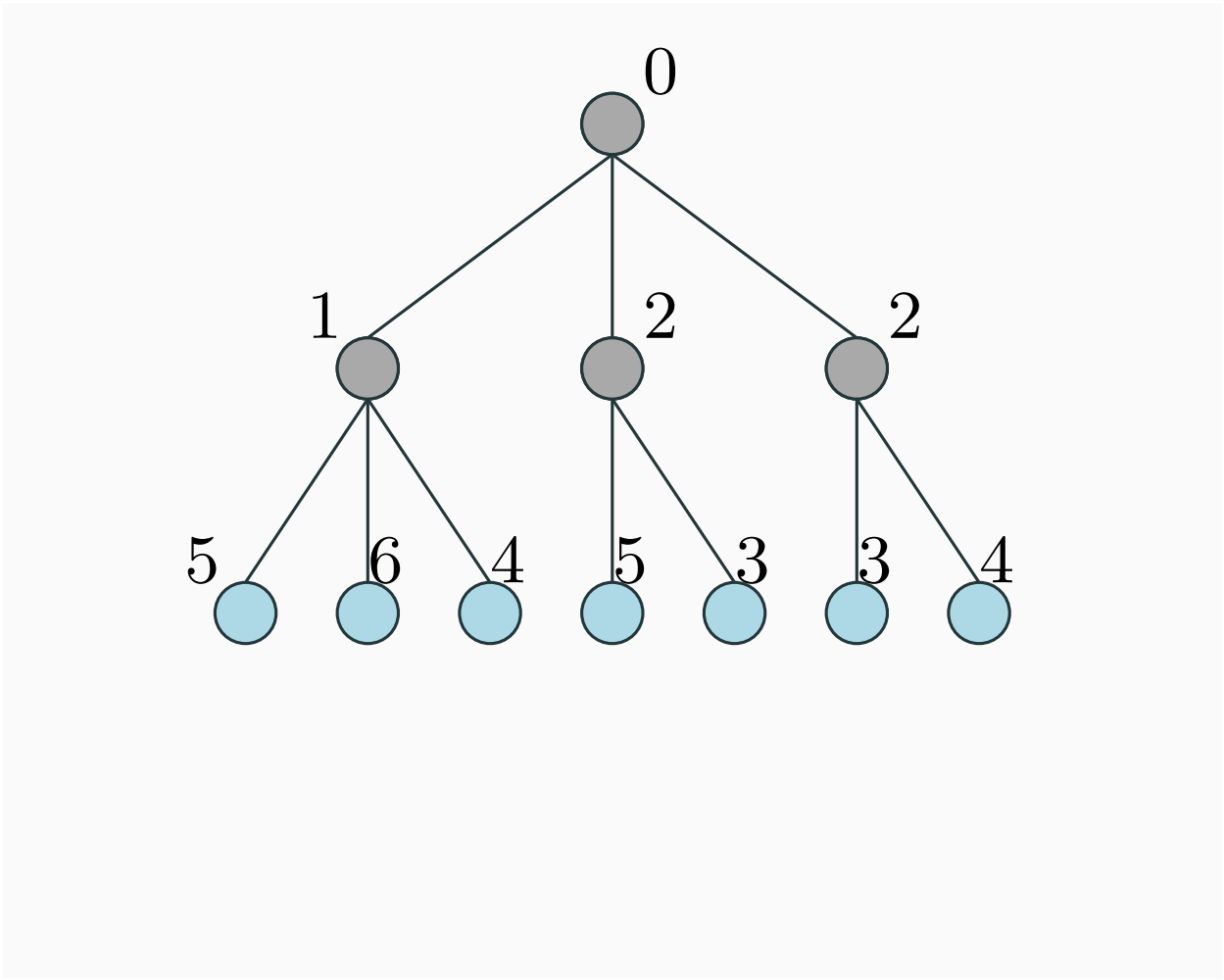
Beam Search ($D = 3$)



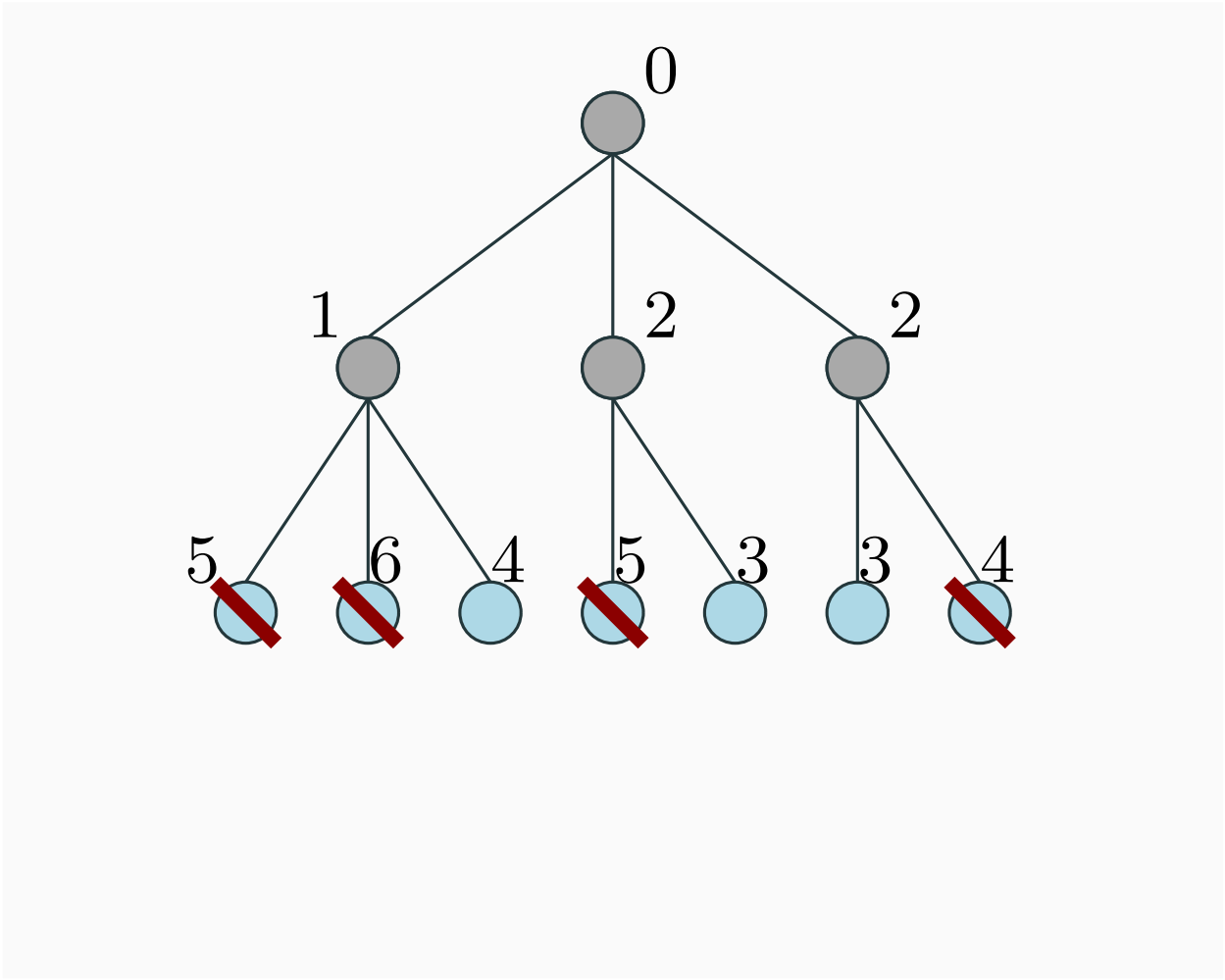
Beam Search ($D = 3$)



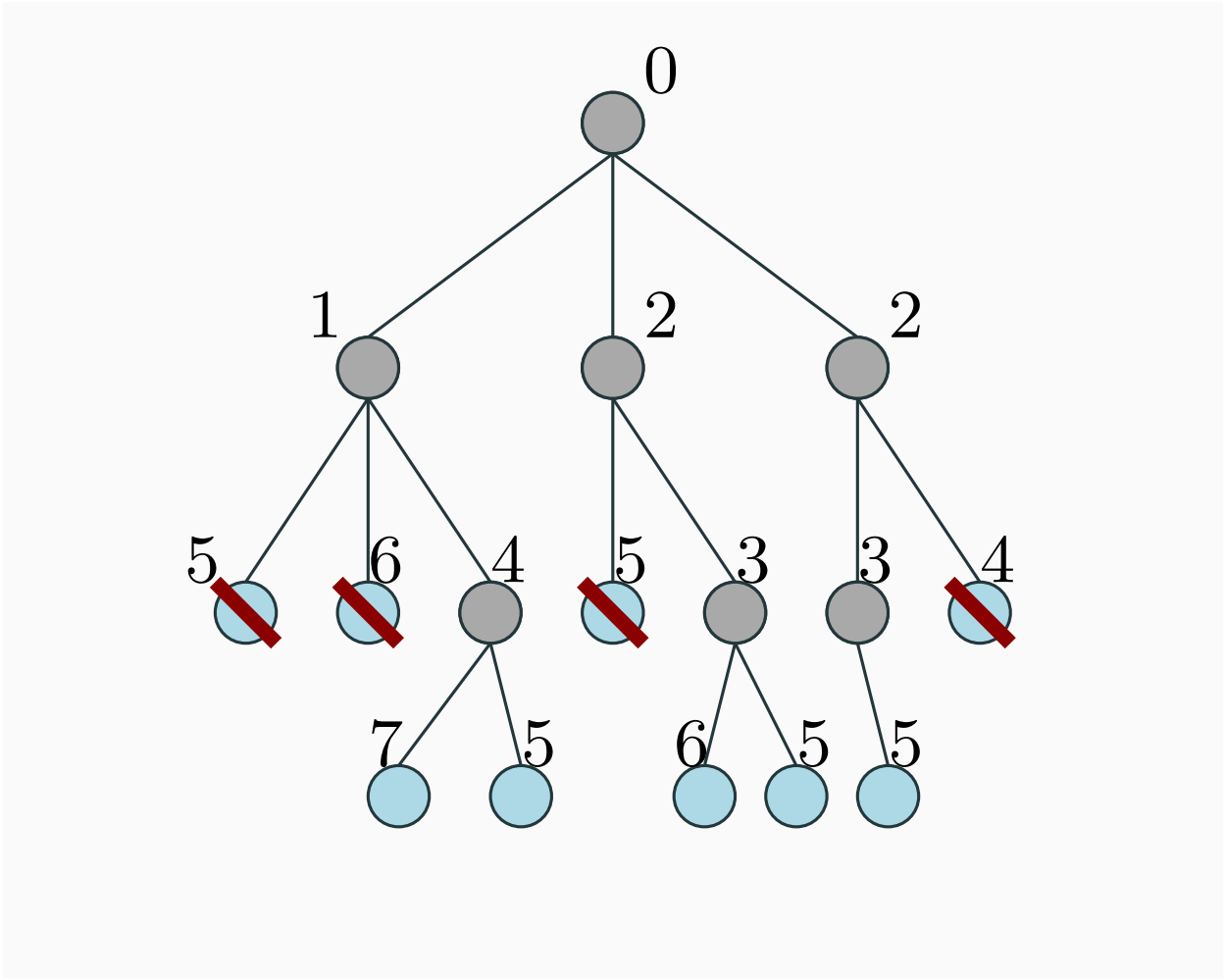
Beam Search ($D = 3$)



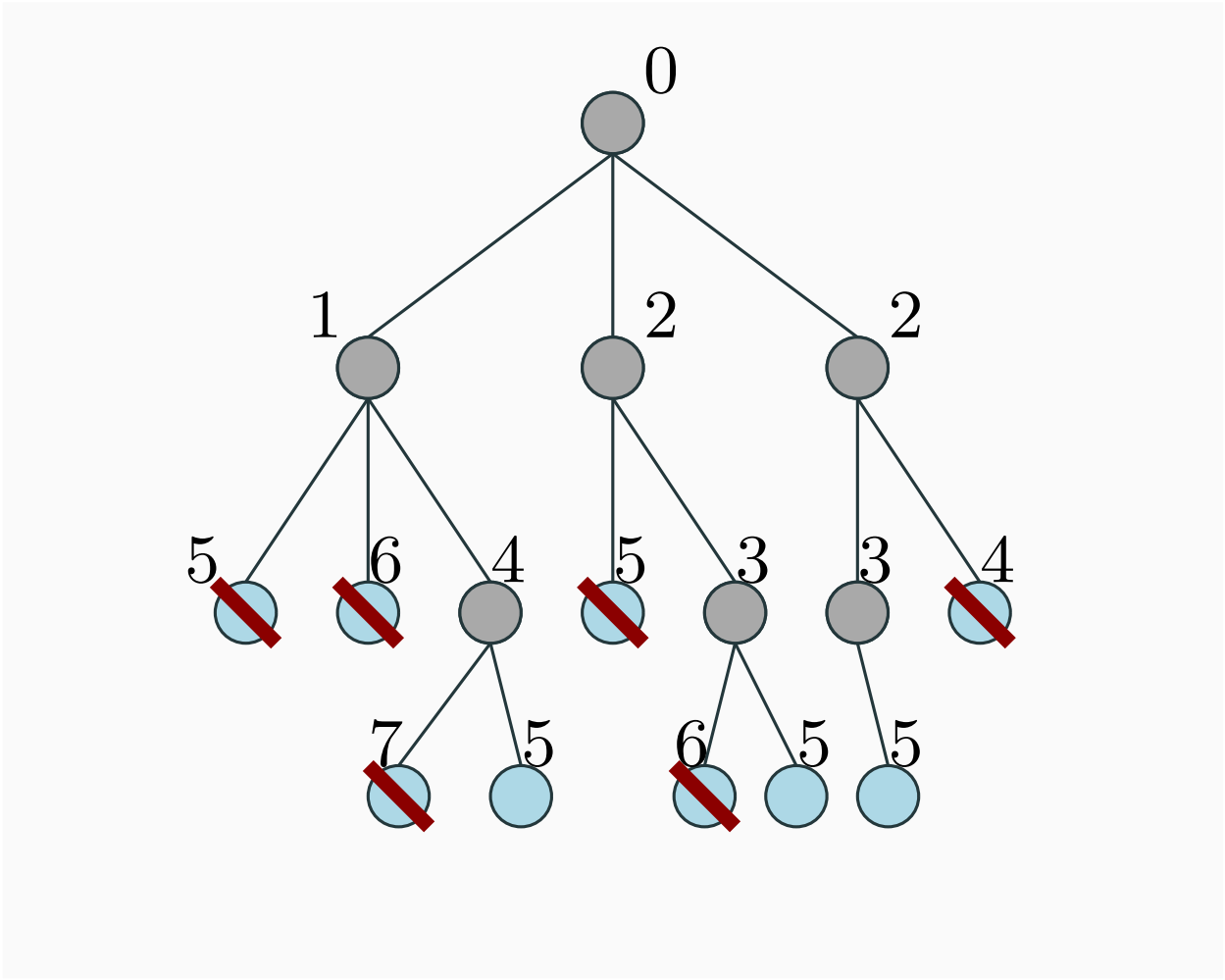
Beam Search ($D = 3$)



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)

Integrating exact-method techniques within heuristics

Dominance fathomings. Inspired from dynamic programming

Example, two partial equivalent solutions:

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

Integrating exact-method techniques within heuristics

Dominance fathomings. Inspired from dynamic programming

Example, two partial equivalent solutions:

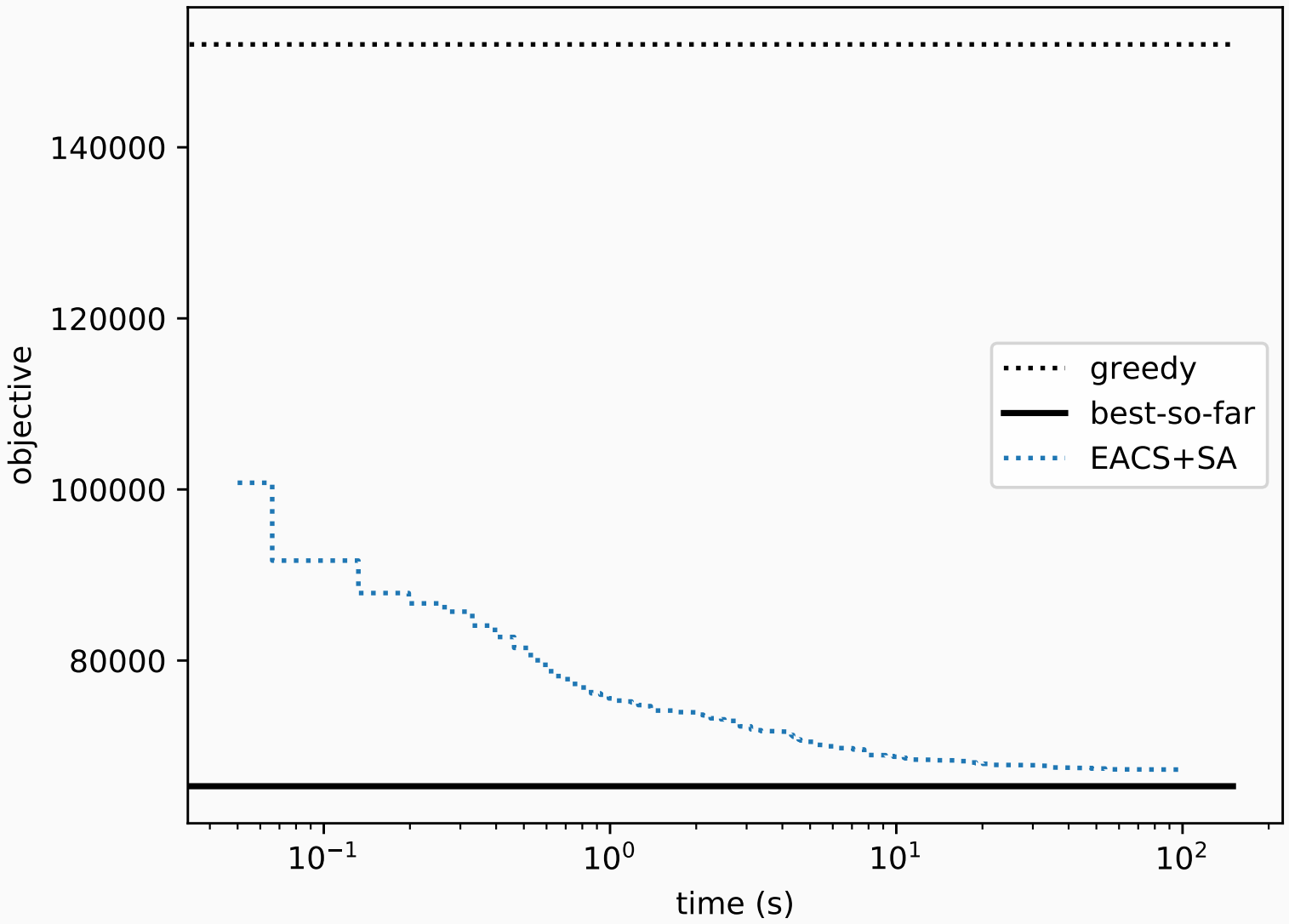
1. **a,b,c,d** cost 10
2. **a,c,b,d** cost 12

Discard (2) as it is “dominated” by (1).
Maintain each entry in a *Hash table*

Numerical Results

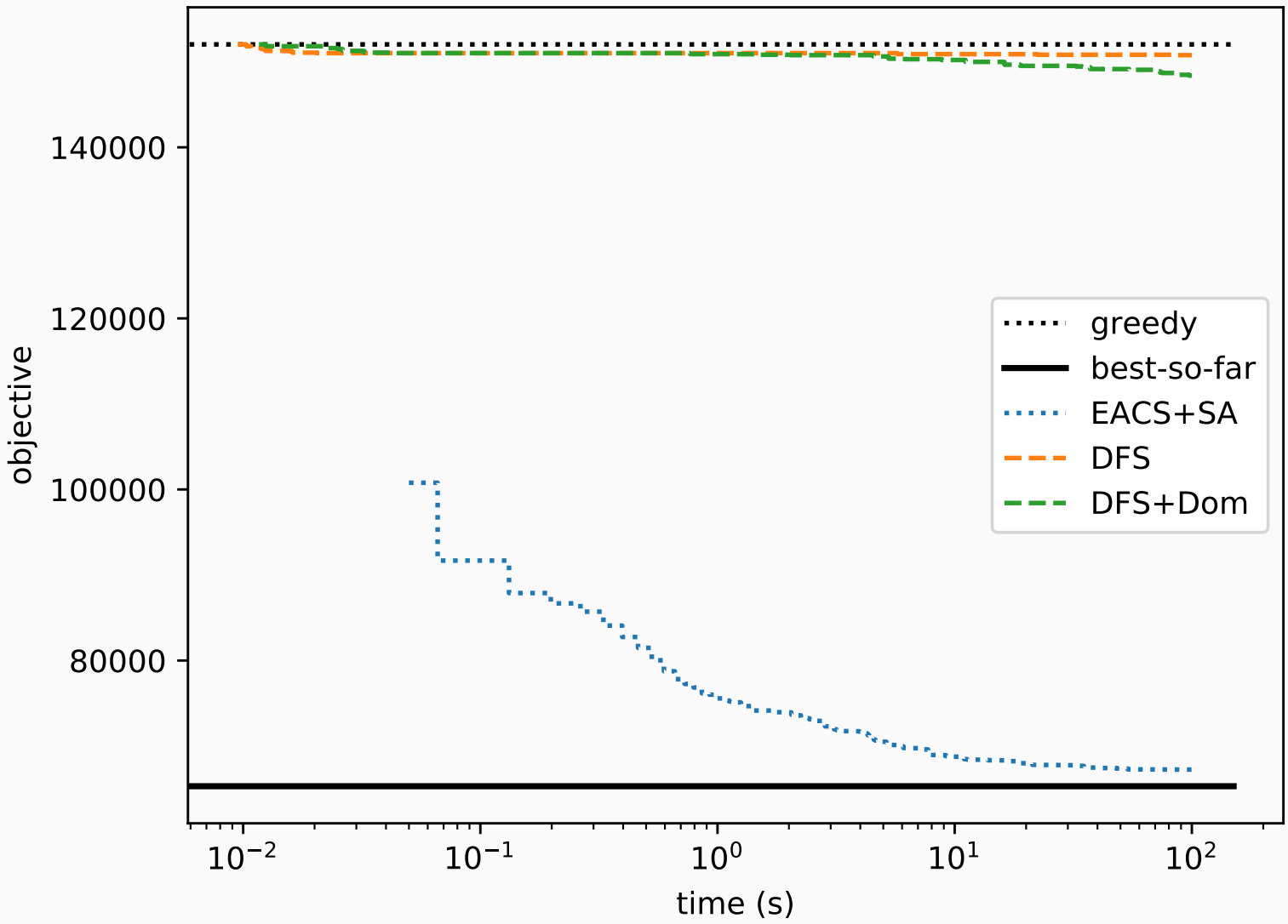
Results - Performance profiles on R.700.1000.15

best-so-far LKH3 with 100.000 seconds run (≈ 27h)



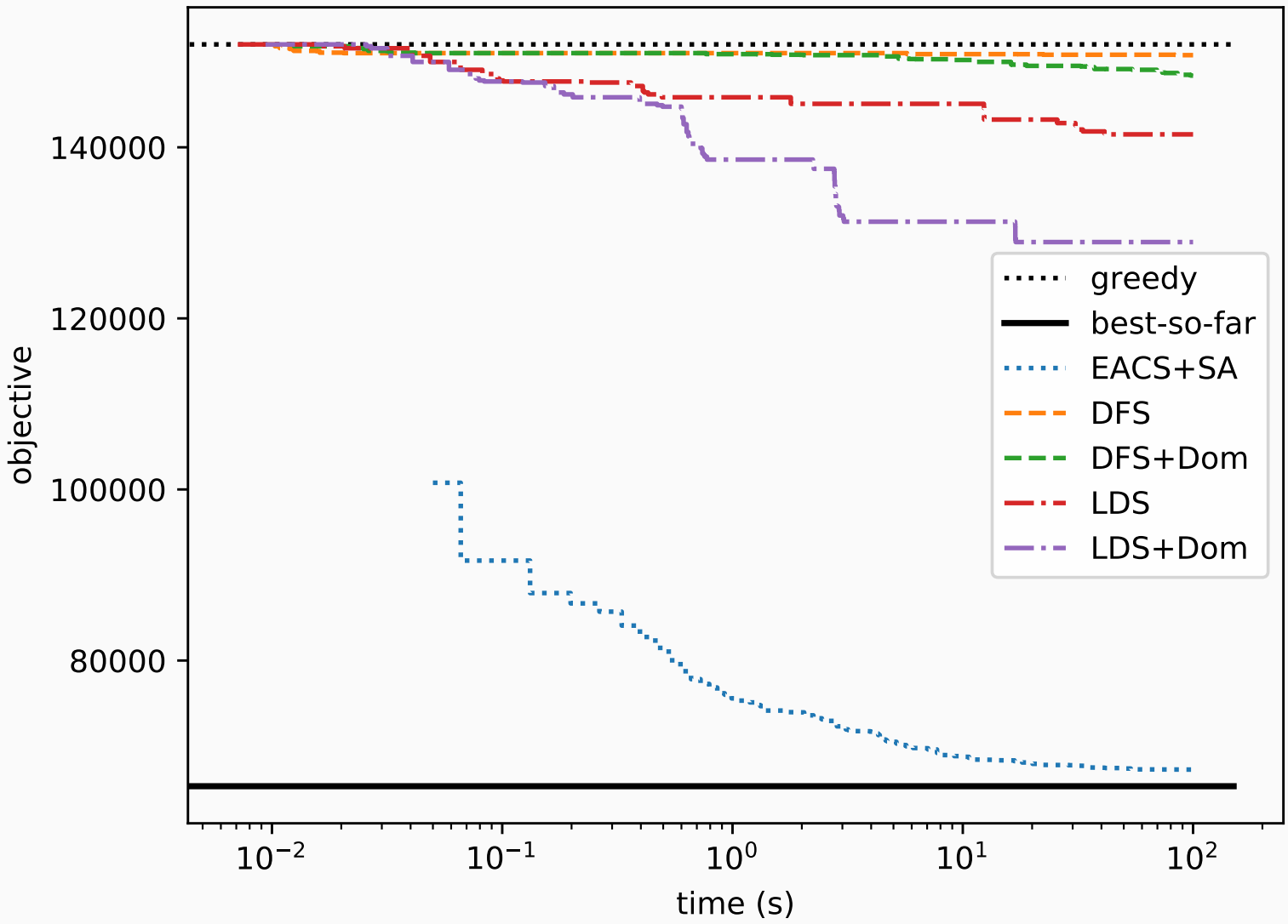
Results - Performance profiles on R.700.1000.15

best-so-far LKH3 with 100.000 seconds run (≈ 27h)



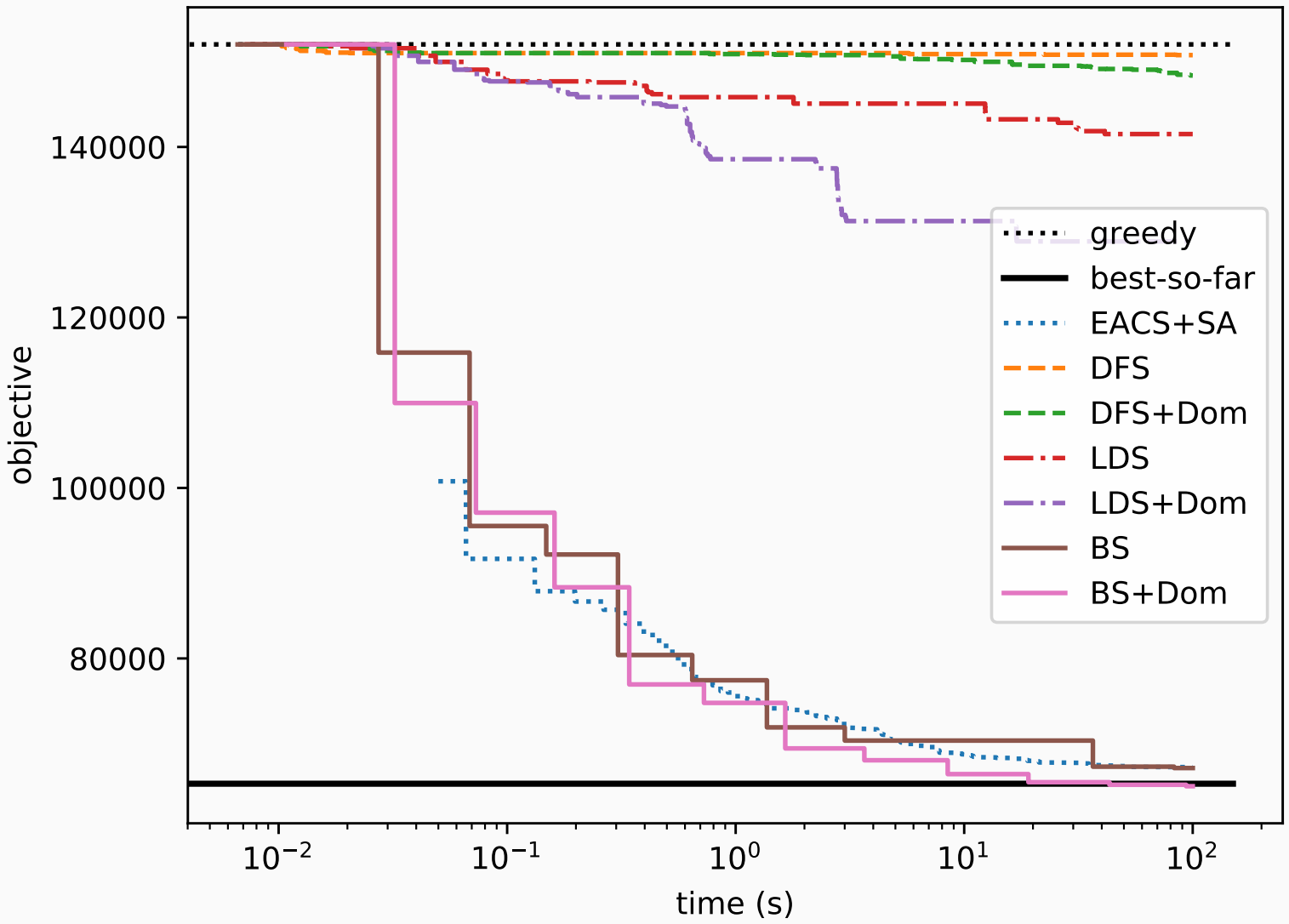
Results - Performance profiles on R.700.1000.15

best-so-far LKH3 with 100.000 seconds run (≈ 27h)



Results - Performance profiles on R.700.1000.15

best-so-far LKH3 with 100.000 seconds run (≈ 27h)



Results - New best-so-far solutions

6 out of 7 new-best-so-far solutions
(the best known solution for the 7th is maybe already optimal)

Instance	best known	BS+Dom (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: poor results. too many choices and too simple guides

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: poor results. too many choices and too simple guides

15% precedence constraints: state-of-the-art. small number of choices (5 children per node in average)

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: poor results. too many choices and too simple guides

15% precedence constraints: state-of-the-art. small number of choices (5 children per node in average)

30%,60% precedence constraints: state-of-the-art. Depletes the search tree (proves optimality) in a few seconds. 10 to 100 times faster than other exact methods.

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: poor results. too many choices and too simple guides

15% precedence constraints: state-of-the-art. small number of choices (5 children per node in average)

30%,60% precedence constraints: state-of-the-art. Depletes the search tree (proves optimality) in a few seconds. 10 to 100 times faster than other exact methods.

The SOPLIB mainly contains heavily constrained instances:

- Hard for MIPs and local searches
- But (relatively) easy for constructive algorithms

Results - Overview

How this simple tree search behaves on the SOPLIB:

1% precedence constraints: poor results. too many choices and too simple guides

15% precedence constraints: state-of-the-art. small number of choices (5 children per node in average)

30%,60% precedence constraints: state-of-the-art. Depletes the search tree (proves optimality) in a few seconds. 10 to 100 times faster than other exact methods.

The SOPLIB mainly contains heavily constrained instances:

- Hard for MIPs and local searches
- But (relatively) easy for constructive algorithms

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

My thesis (goal)

- Study tree searches (from AI/planning, OR, and heuristics)
- Combine them and study their efficiency on OR problems

My thesis (goal)

- Study tree searches (from AI/planning, OR, and heuristics)
- Combine them and study their efficiency on OR problems

- **Results on SOP** (accepted at ECAI2020 [2])
- Cutting & packing (EURO/ROADEF challenge and variants)

My thesis (goal)

- Study tree searches (from AI/planning, OR, and heuristics)
- Combine them and study their efficiency on OR problems

- **Results on SOP** (accepted at ECAI2020 [2])
- Cutting & packing (EURO/ROADEF challenge and variants)

- Generic tree search framework

Why branch-and-bounds are meta-heuristics

state-of-the-art heuristics from branch-and-bounds

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

February, 20, 2020 - ROADEF 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.

[2] Luc Libralesso, Abdel-Malik Bouhassoun, Hadrien Cambazard, and Vincent Jost. Tree searches for the Sequential Ordering Problem. working paper or preprint, January 2020. URL *https://hal.archives-ouvertes.fr/hal-02374896*.