Tree searches for Sequential Ordering Problem

Contradicting conventional wisdom

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

January, 30, 2020 - RealOpt Seminar

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

Before starting

Please feel free to ask me questions at any time!

About the subtitle

Reference to [2]

Closing the open shop: Contradicting conventional wisdom (2009)

Diarmuid Grimes, Emmanuel Hebrard, and Arnaud Malapert

About the subtitle

Reference to [2]

Closing the open shop: Contradicting conventional wisdom (2009)

Diarmuid Grimes, Emmanuel Hebrard, and Arnaud Malapert

- solves the open shop (by the time, known to be hard)
- uses a simple CP model (weighted degree)
- learning which task is "hard" outperform sophisticated inferences.

About the subtitle

Reference to [2]

Closing the open shop: Contradicting conventional wisdom (2009)

Diarmuid Grimes, Emmanuel Hebrard, and Arnaud Malapert

- solves the open shop (by the time, known to be hard)
- uses a simple CP model (weighted degree)
- learning which task is “hard” outperform sophisticated inferences.

we identify some simple components that outperform state-of-the-art (on the SOPLIB).

Table of contents

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, evolutionary algorithms …

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

Tree searches: usually constructive algorithms (explore a tree)

Anytime tree searches - some (trivial) definitions

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

Many presented in AI/planning conferences

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

Anytime tree searches

Many presented in AI/planning conferences

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

Sometimes in Operations Research (successful in Cutting & Packing)

Anytime tree searches

Many presented in AI/planning conferences

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

Sometimes in Operations Research (successful in Cutting & Packing)

Not used much compared to classical meta-heuristics (tabu search, genetic algorithms ...)

why anytime tree searches are not used more?

two hypothesis (on SOP):

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

We believe the latter is true

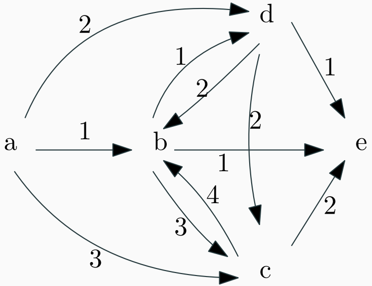
Our experiment

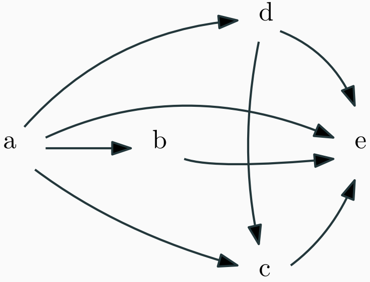
- We consider a well known benchmark (SOP)
- Apply anytime tree searches

Sequential Ordering Problem

SOP - problem definition

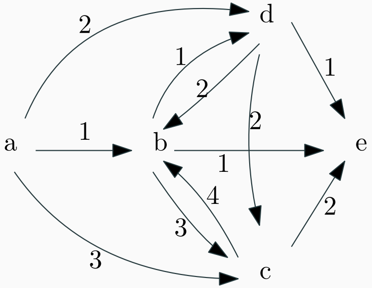
Asymmetric Traveling Salesman Problem with precedence constraints

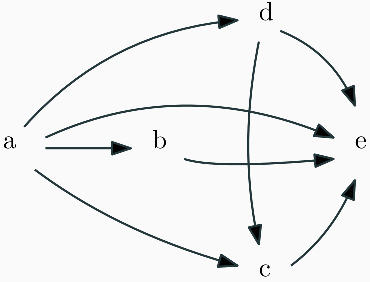




SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints

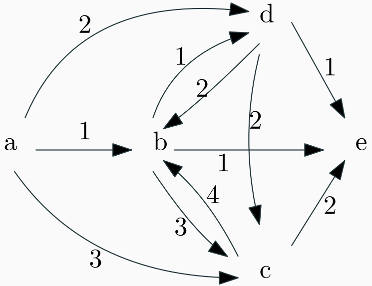


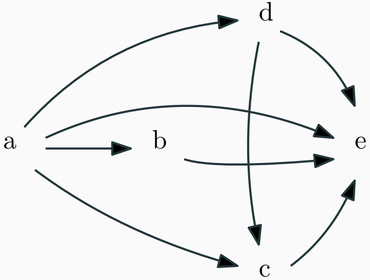


- **a,d,c,b,e** is a feasible and costs 10

SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints

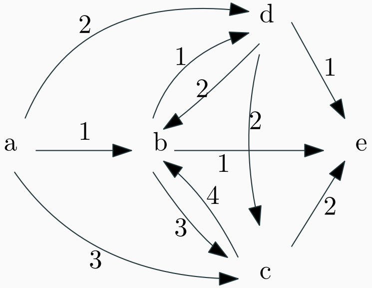


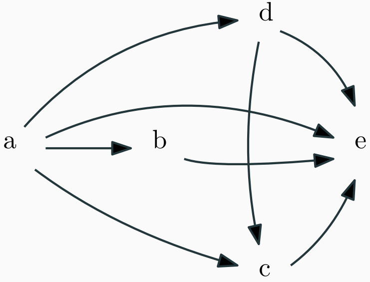


- **a,d,c,b,e** is a feasible and costs 10
- **a,b,c,d,e** is not feasible

SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints





- a,d,c,b,e is a feasible and costs 10
- a,b,c,d,e is not feasible
- a,b,d,c,e is optimal and costs 6

The benchmark: SOPLIB

- proposed 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)

The benchmark: SOPLIB

- proposed 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)

- **1% precedences** are easy (using MIP + lazy constraints)
- **15% precedences** are “hard”
- **30% and 60% precedences** are easy (solved by dynamic programming)

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

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 searches (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 searches (GA, ABC, parallel roll-out, LKH ...)

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

Tree Search

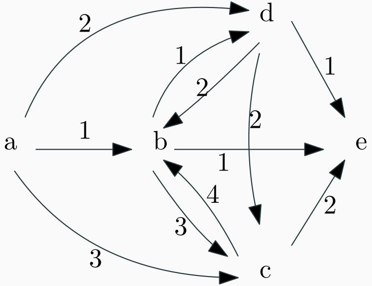
Two parts:

Implicit tree: how to branch, bounds ...

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

Implicit tree - Branching

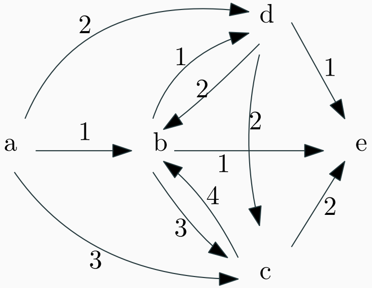
Forward branching

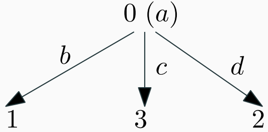


$0\ (a)$

Implicit tree - Branching

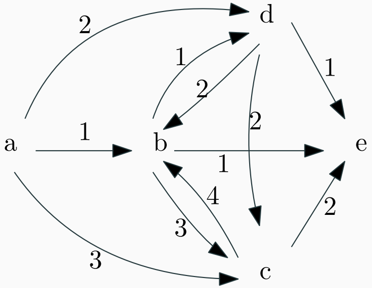
Forward branching

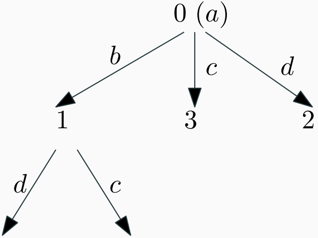




Implicit tree - Branching

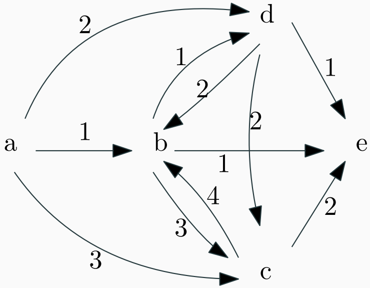
Forward branching

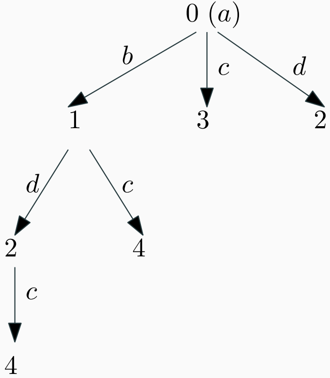




Implicit tree - Branching

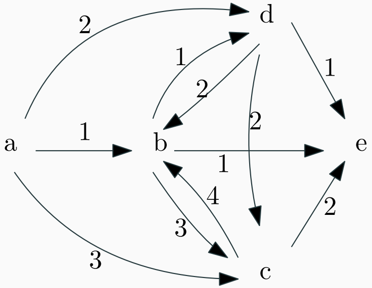
Forward branching

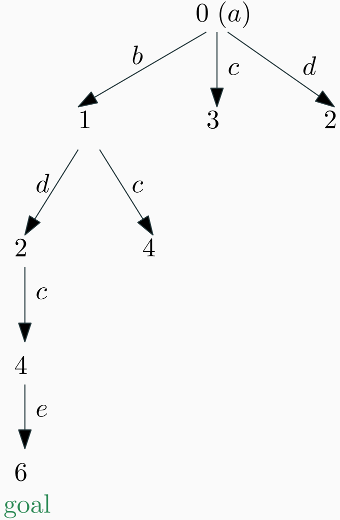




Implicit tree - Branching

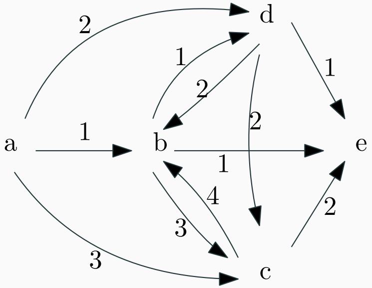
Forward branching

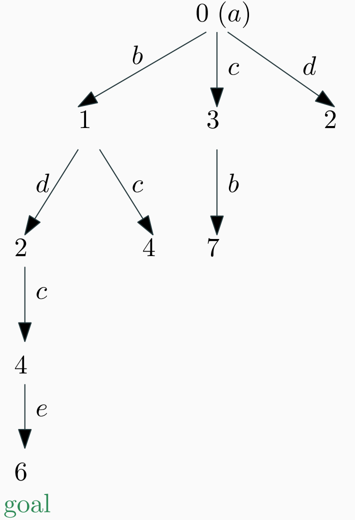




Implicit tree - Branching

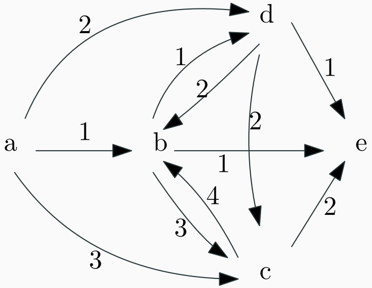
Forward branching

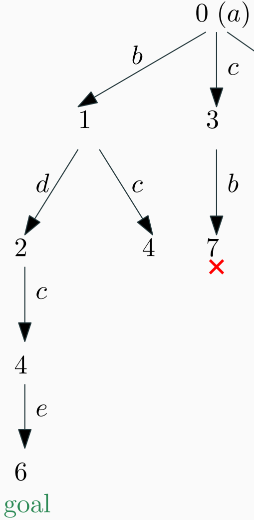




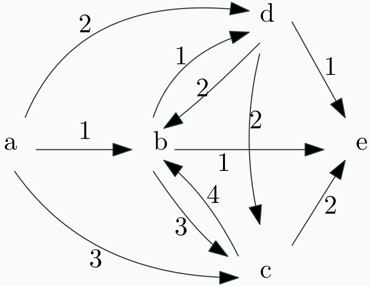
Implicit tree - Branching

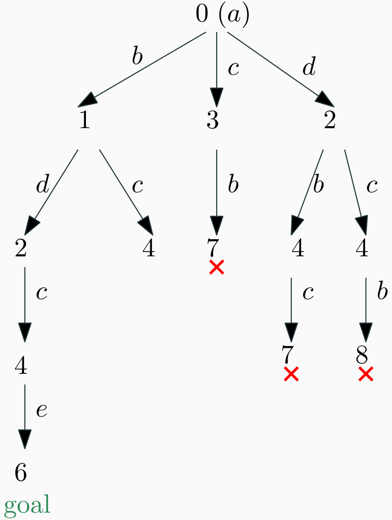
Forward branching





Forward branching





Bounds

We consider 3 (simple) bounds:

Prefix bound: only arcs between selected vertices (previous example)

ingoing/outgoing: adding minimum ingoing/outgoing arc for not-selected vertices

Bounds

We consider 3 (simple) bounds:

Prefix bound: only arcs between selected vertices (previous example)

ingoing/outgoing: adding minimum ingoing/outgoing arc for not-selected vertices

MST: compute a MST using remaining vertices

Bounds

We consider 3 (simple) bounds:

Prefix bound: only arcs between selected vertices (previous example)

ingoing/outgoing: adding minimum ingoing/outgoing arc for not-selected vertices

MST: compute a MST using remaining vertices

Bounds

We consider 3 (simple) bounds:

Prefix bound: only arcs between selected vertices (previous example)

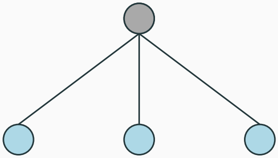
ingoing/outgoing: adding minimum ingoing/outgoing arc for not-selected vertices

MST: compute a MST using remaining vertices

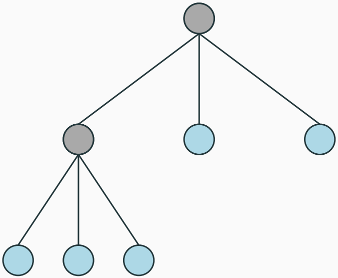
Out of our experiments, the prefix bound (despite its simplicity) provides the same guiding quality as other bounds. For simplicity, we only show results using it.

Depth First Search

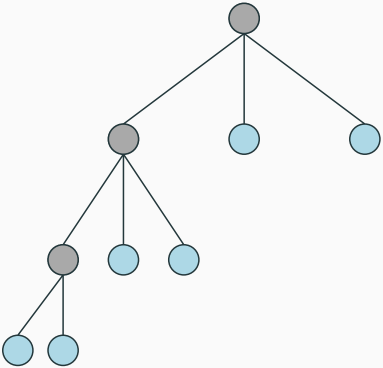
Depth First Search



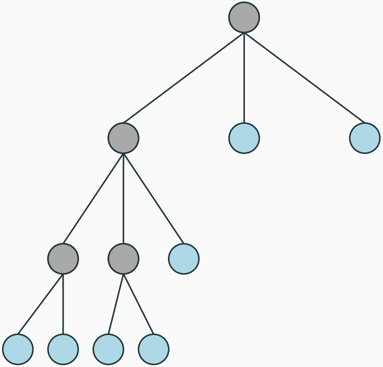
Depth First Search



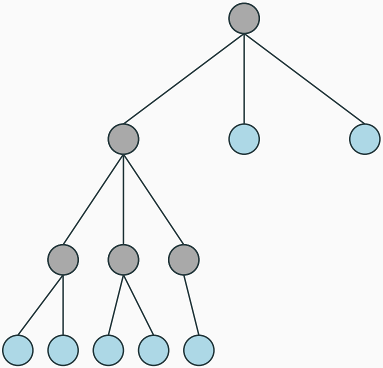
Depth First Search



Depth First Search



Depth First Search

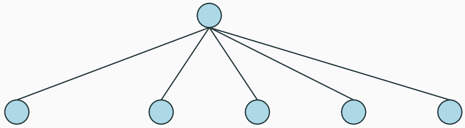


Depth First Search - Drawbacks

Stuck in early sub-optimal choices near the root

Limited Discrepancy Search (LDS) - key idea

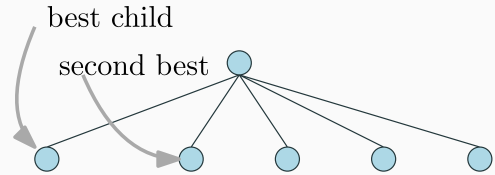
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

Limited Discrepancy Search (LDS) - key idea

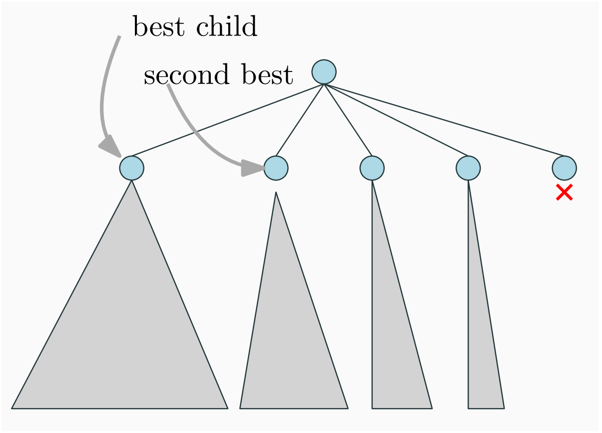
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

Limited Discrepancy Search (LDS) - key idea

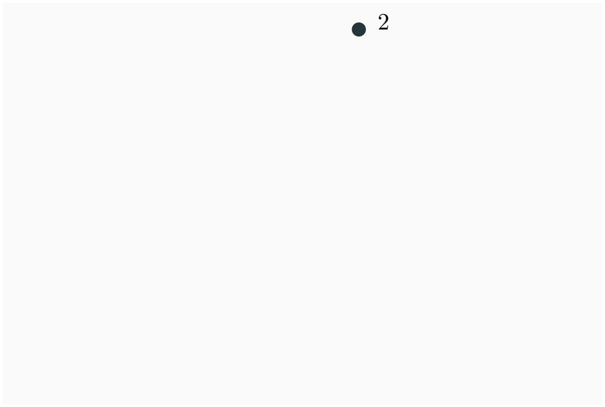
Correct DFS drawback: early bad decisions



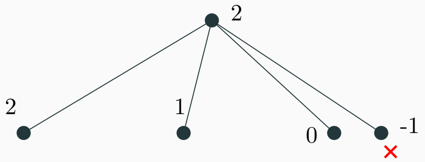
Explore more the most promising but still keep exploring others

Limited Discrepancy Search (LDS) - An example: D=2

Count the number of deviations from the best child (discrepancies)

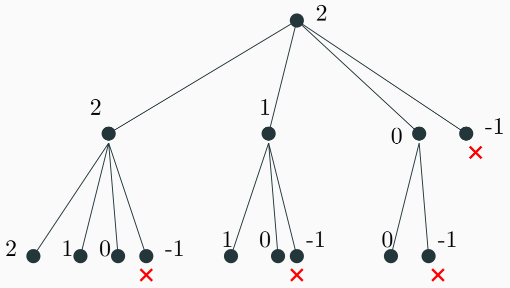


Count the number of deviations from the best child (discrepancies)



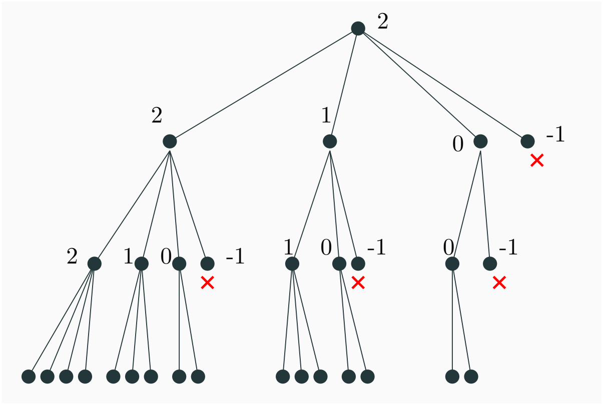
Limited Discrepancy Search (LDS) - An example: D=2

Count the number of deviations from the best child (discrepancies)

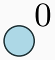


Limited Discrepancy Search (LDS) - An example: D=2

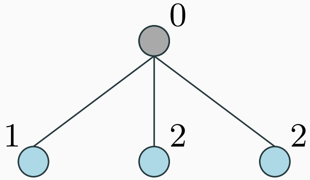
Count the number of deviations from the best child (discrepancies)



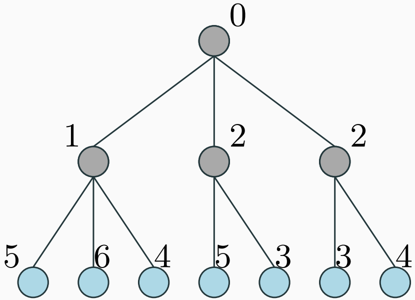
Beam Search ($D = 3$)



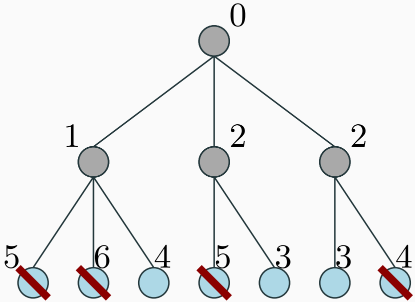
Beam Search ($D = 3$)



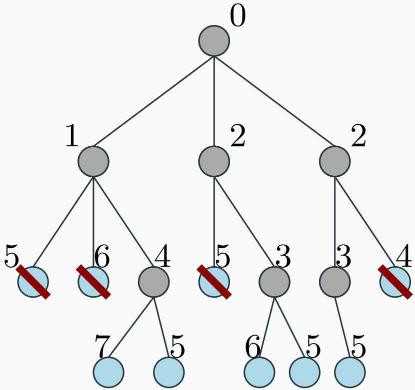
Beam Search ($D = 3$)



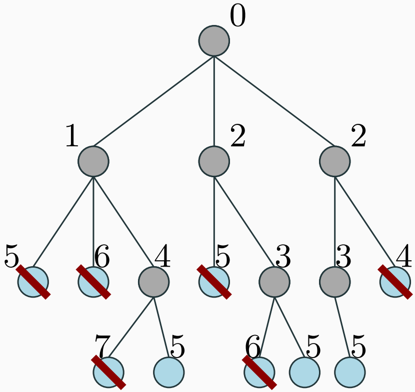
Beam Search ($D = 3$)



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)

Prefix equivalence fathomings

Inspired from dynamic programming

Example, two partial equivalent[1] solutions:

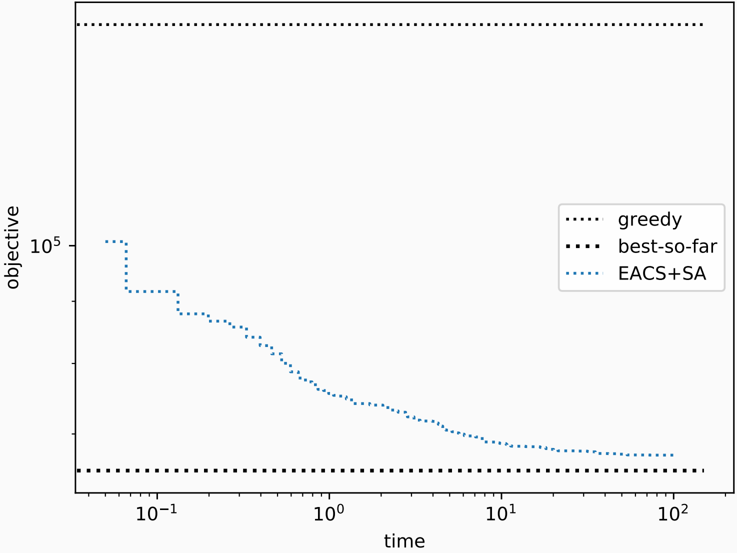
1. **a,b,c,d** cost 10
2. **a,c,b,d** cost 12

Discard (2) as it is "dominated" by (1).

[1] have the same sub-trees. The node contains the same subset of visited nodes and the same last vertex

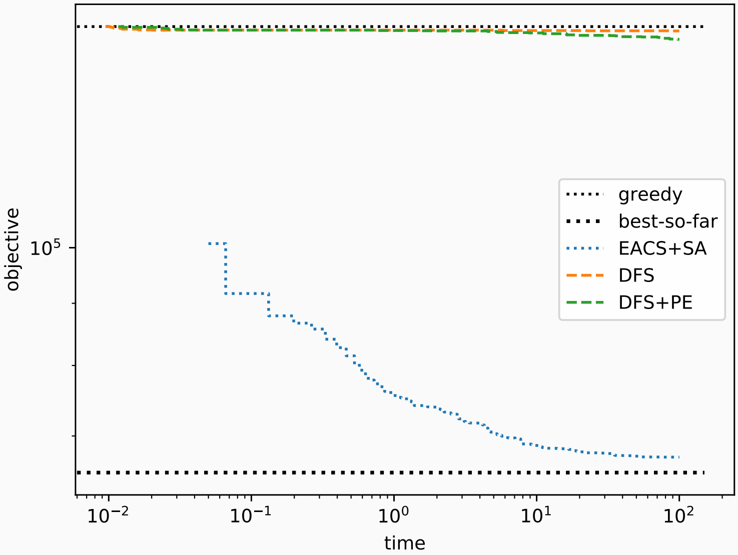
Results - Performance profiles on R.700.1000.15

best-so-far LKH3 with 100.000 seconds run ($\approx$ 27h)



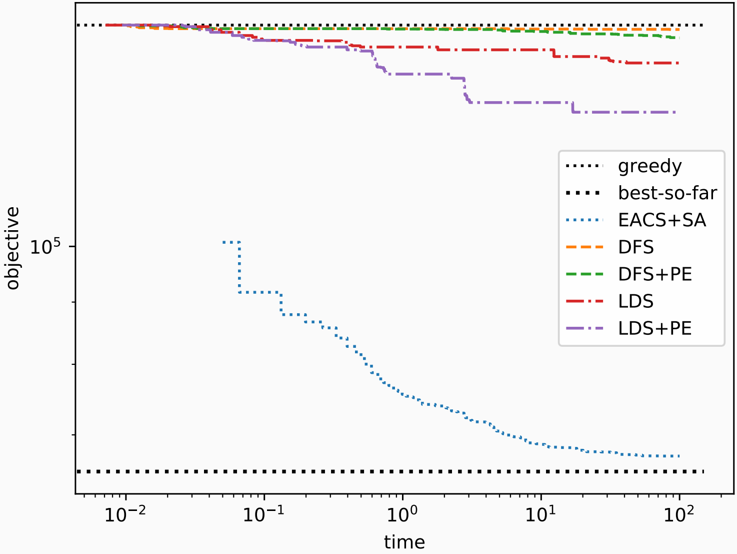
Results - Performance profiles on R.700.1000.15

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



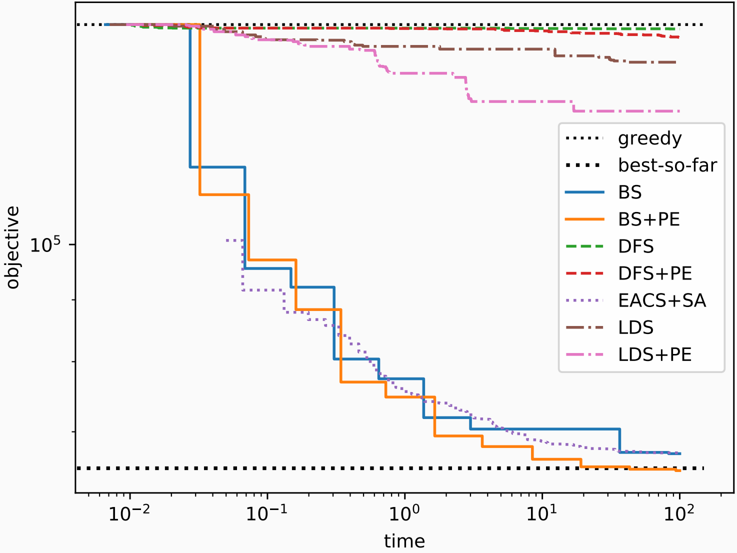
Results - Performance profiles on R.700.1000.15

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 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: 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: depletes the search tree (proves optimality) in a few seconds. 10 to 100 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: depletes the search tree (proves optimality) in a few seconds. 10 to 100 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: depletes the search tree (proves optimality) in a few seconds. 10 to 100 faster than other exact methods.

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

Conclusions

- At least on the SOPLIB, anytime tree searches outperform classical meta-heuristics
- The search-strategy choice is crucial
- An analysis of the impact of each algorithmic component leads to interesting insights

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
- 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
- cutting & packing (EURO/ROADEF challenge and variants)
- Generic tree search framework

Tree searches for Sequential Ordering Problem

Contradicting conventional wisdom

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

January, 30, 2020 - RealOpt Seminar

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] Diarmuid Grimes, Emmanuel Hebrard, and Arnaud Malapert. Closing the open shop: Contradicting conventional wisdom. In *International Conference on Principles and Practice of Constraint Programming*, pages 400–408. Springer, 2009.