Anytime tree search for discrete optimization

A tutorial

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

April, 28, 2020 - ROSP Seminar

G-SCOP, Grenoble, France
email: luc.libralesso@grenoble-inp.fr

Before starting

Please feel free to ask me questions at any time!

About the title

Anytime: provides good solutions fast and is able to improve them with more time. Similar to meta-heuristics.

About the title

Anytime: provides good solutions fast and is able to improve them with more time. Similar to meta-heuristics.

Tree search: explores a tree (more on it later)

Anytime tree search for discrete optimization

About the title

Anytime: provides good solutions fast and is able to improve them with more time. Similar to meta-heuristics.

Tree search: explores a tree (more on it later)

Discrete optimization: all variables are discrete (Integer or Boolean)

Goal of this tutorial

- Quickly present search algorithms for optimization

Goal of this tutorial

- Quickly present search algorithms for optimization
- With a focus on anytime tree search

Goal of this tutorial

- Quickly present search algorithms for optimization
- With a focus on anytime tree search
- Some applications (academic and industrial)

Table of contents

Optimization and search

We want to find the best possible solution out of a finite and **huge** number of solutions.

Example: (Asymmetric) Traveling Salesman Problem

INPUT:

- graph $G = (V, A)$
- distance function $w : A \to \mathbb{R}$

Example: (Asymmetric) Traveling Salesman Problem

INPUT:

- graph $G = (V, A)$
- distance function $w : A \to \mathbb{R}$

GOAL:

- Find a tour that visit all n cities
- Minimize the distance of selected arcs
- $n!$ possible solutions

Resolution Methods (you may already know)

- Brute Force (very bad)

Resolution Methods (you may already know)

- Brute Force (very bad)
- Branch and Bound (better)

Resolution Methods (you may already know)

- Brute Force (very bad)
- Branch and Bound (better)
- Mixed Integer Programming (LP + Branch and Bound)

Resolution Methods (you may already know)

- Brute Force (very bad)
- Branch and Bound (better)
- Mixed Integer Programming (LP + Branch and Bound)
- Meta-heuristics:
    - Local Search
    - Simulated Annealing
    - Genetic Algorithms
    - Ant Colony Optimization
    - *etc.*

Classification

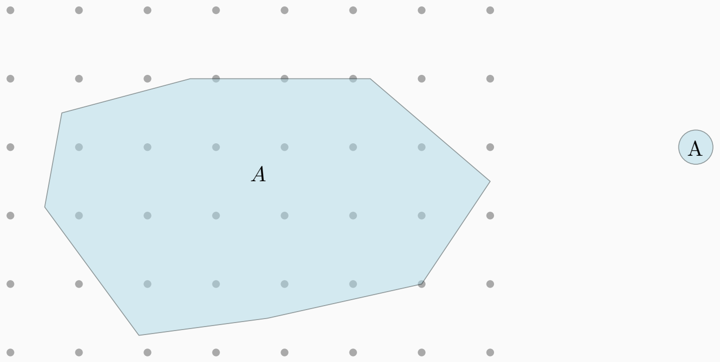
Search procedures are often labeled as:

- Tree Search
- Local Search
- Population Based Search

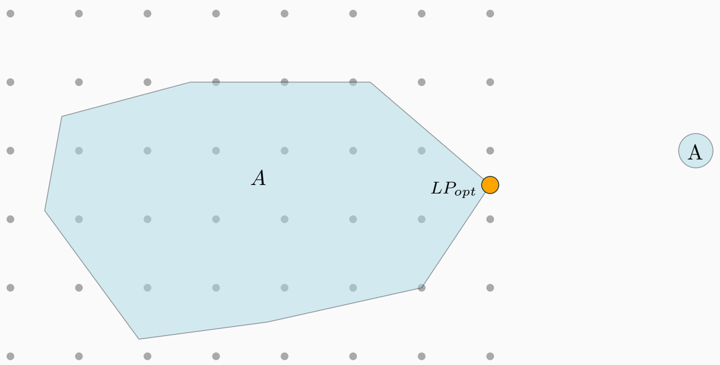
Tree Search

- usually ”constructs” solutions
- Models the problem as a tree
- Explores this tree

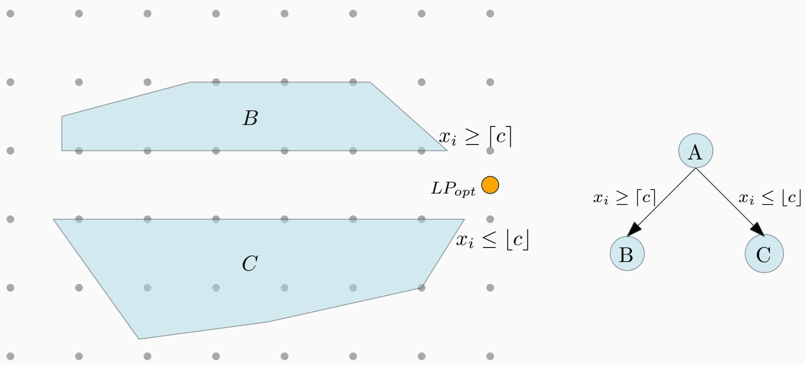
Example: Mixed Integer Programming



Example: Mixed Integer Programming

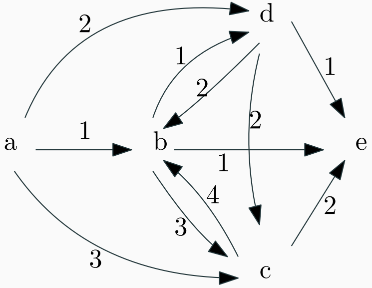


Example: Mixed Integer Programming



Example: Dedicated Branch and Bound

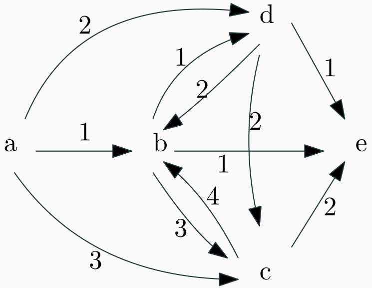
A Sequential Ordering Problem Branch and Bound (more later)

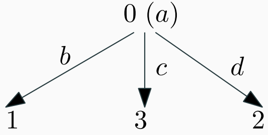


$0\ (a)$

Example: Dedicated Branch and Bound

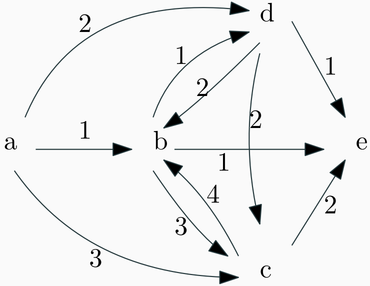
A **Sequential Ordering Problem** Branch and Bound (more later)

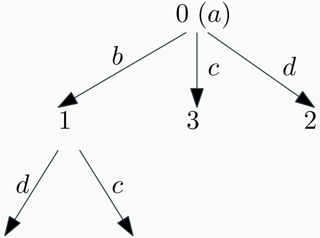




Example: Dedicated Branch and Bound

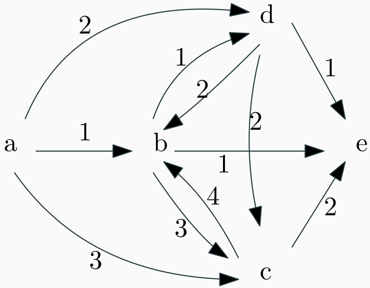
A **Sequential Ordering Problem** Branch and Bound (more later)

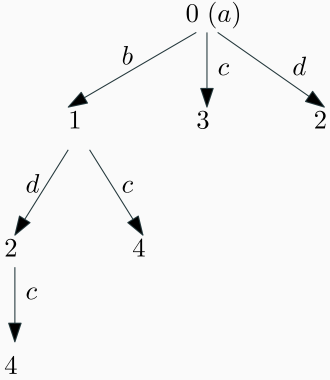




Example: Dedicated Branch and Bound

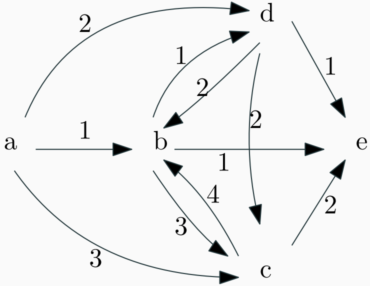
A **Sequential Ordering Problem** Branch and Bound (more later)

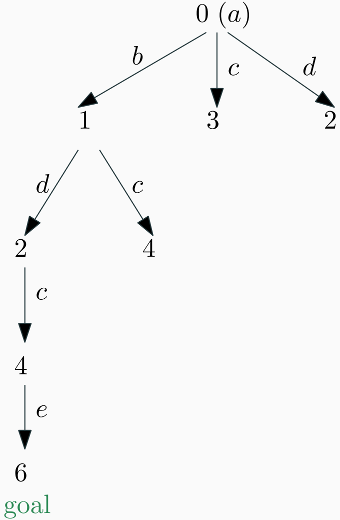




Example: Dedicated Branch and Bound

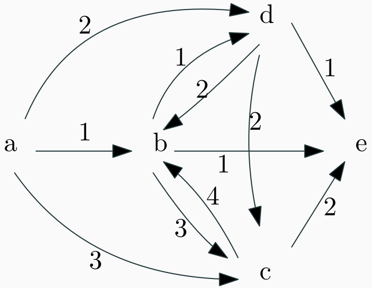
A **Sequential Ordering Problem** Branch and Bound (more later)

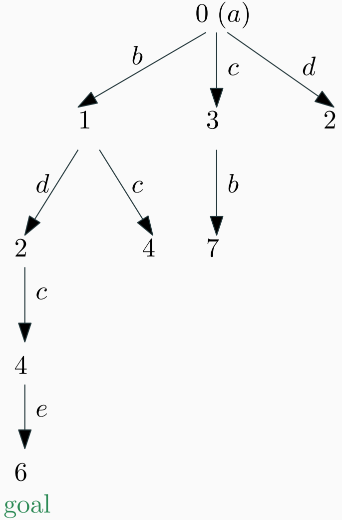




Example: Dedicated Branch and Bound

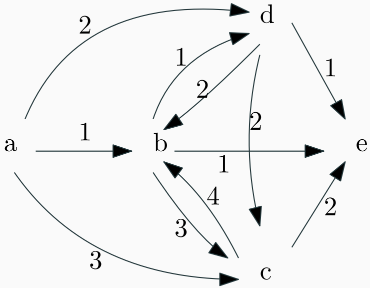
A **Sequential Ordering Problem** Branch and Bound (more later)

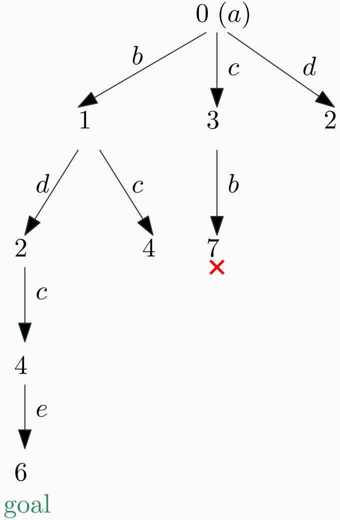




Example: Dedicated Branch and Bound

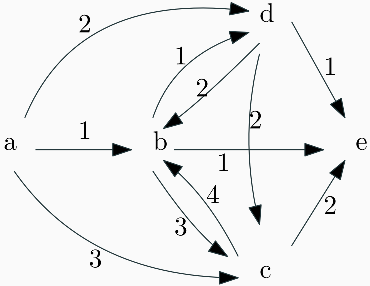
A **Sequential Ordering Problem** Branch and Bound (more later)

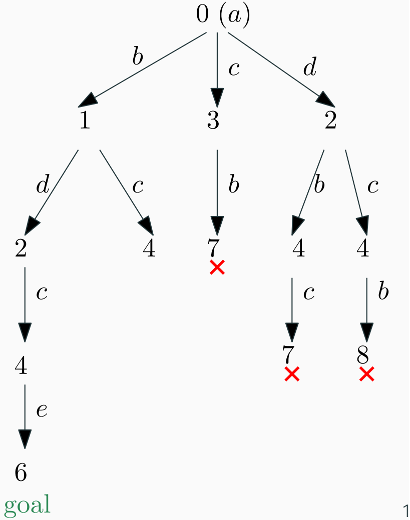




Example: Dedicated Branch and Bound

A **Sequential Ordering Problem** Branch and Bound (more later)





How to improve a branch and bound

- Better bounds: MST, assignment problem, LP, *etc.* The stronger, the more prunings, but also more expensive to compute.

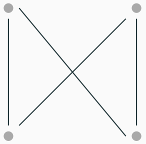
How to improve a branch and bound

- Better bounds: MST, assignment problem, LP, *etc.* The stronger, the more prunings, but also more expensive to compute.
- Better search strategy (will be discussed in this talk)

Local Search

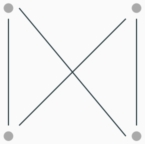
- usually improves an existing solution
- by exploring similar solutions

Local Search Example - 2-opt for the TSP

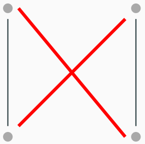


Initial solution (tour)

Local Search Example - 2-opt for the TSP

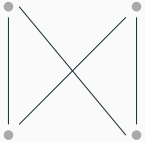


Initial solution (tour)

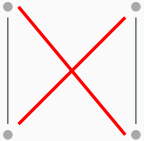


Identify edges to remove (in red)

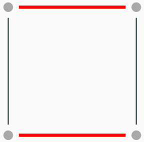
Local Search Example - 2-opt for the TSP



Initial solution (tour)



Identify edges to remove (in red)



Replace them

Population Based Search

- Consider a set of solutions (population)
- Combines promising solutions together (crossover)
- Possibly alter solutions (mutations)

Recap

	Operators	Examples
Tree Search	children, bounds	MIP, CP, (more later ...)
Local Search	neighbourhood	Tabu Search, SA ...
Population Based	crossover, mutation distance from a solution	Genetic/Evolutionary

Tree Search

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 search algorithms**

Paradigm

Made of two parts:

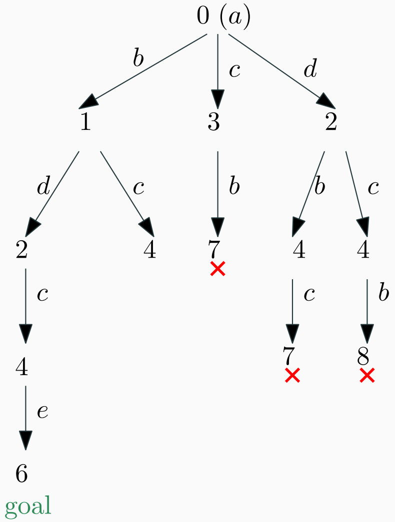
- The Implicit Tree definition:
  - root
  - children
  - bounds (optimistic estimate)
  - isGoal
  - possibly other information (*i.e.* guides, dominance prunings)

Paradigm

Made of two parts:

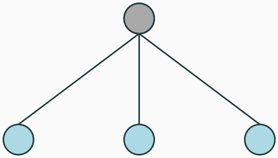
- The Implicit Tree definition:
    - root
    - children
    - bounds (optimistic estimate)
    - isGoal
    - possibly other information (*i.e.* guides, dominance prunings)
- The Search Procedure (generic):
    - Depth First Search (DFS)
    - Best First Search
    - A*
    - Others (discussed in a few slides)

About the Tree Search Formalism: An example

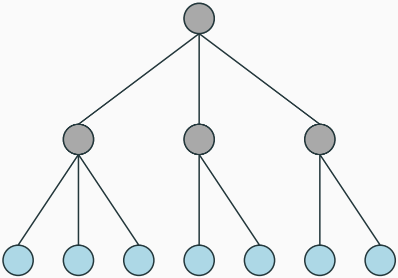


Breadth First Search

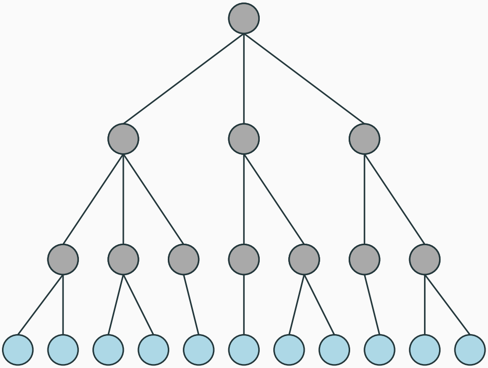
Breadth First Search



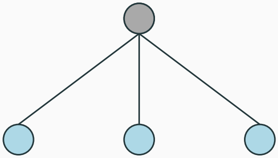
Breadth First Search



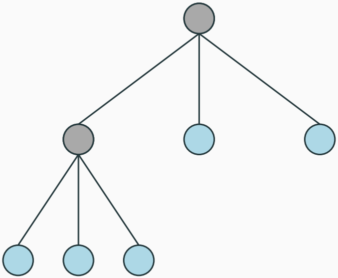
Breadth First Search



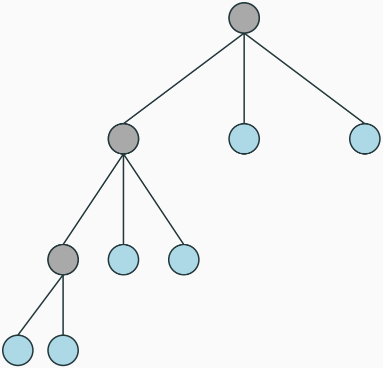
Depth First Search



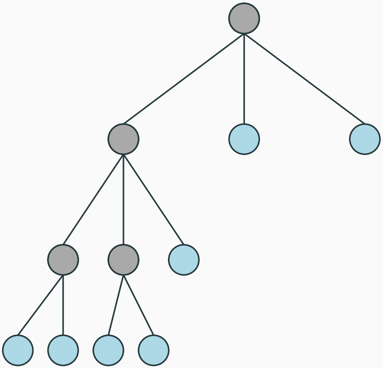
Depth First Search



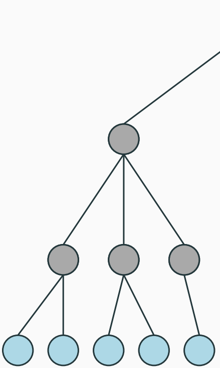
Depth First Search



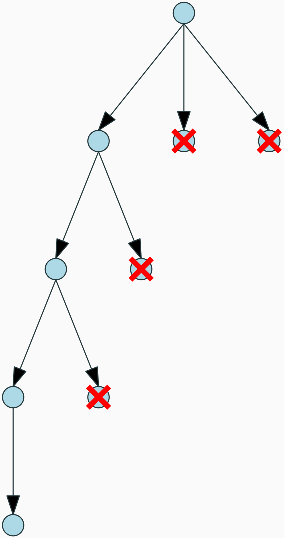
Depth First Search



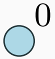
Depth First Search



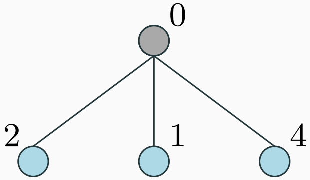
Greedy



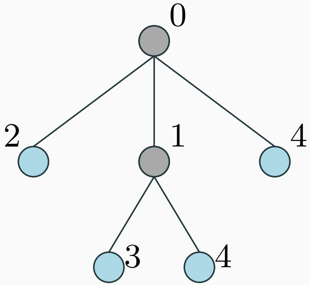
Best (bound) First / A*



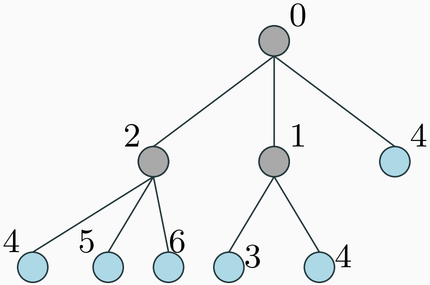
Best (bound) First / A*



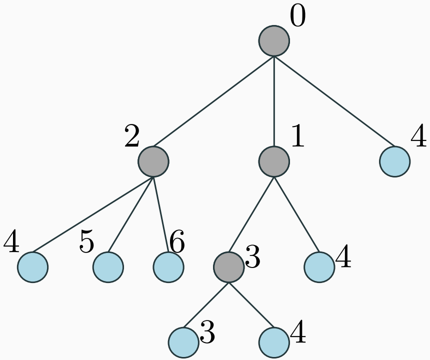
Best (bound) First / A*



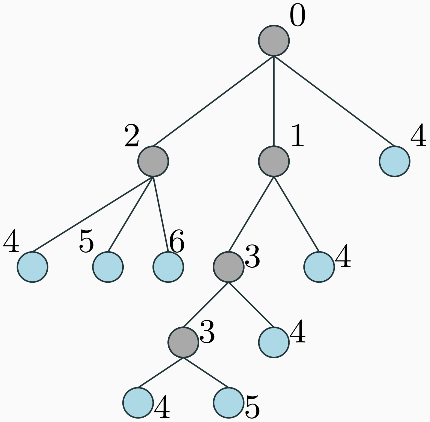
Best (bound) First / A*



Best (bound) First / A*



Best (bound) First / A*



Exercise 1 : Advantages and Drawbacks

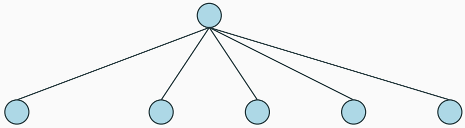
	Depth First Search	A*/Best First

Exercise 1 : Advantages and Drawbacks

	Depth First Search	A*/Best First
Pros	1. Anytime 2. Memory Bounded	1. less nodes to close the instance 2. no need of good solutions
Cons	1. requires good solutions 2. suffers from early bad decisions	1. not anytime 2. Can use too much memory

Limited Discrepancy Search (LDS) - key idea

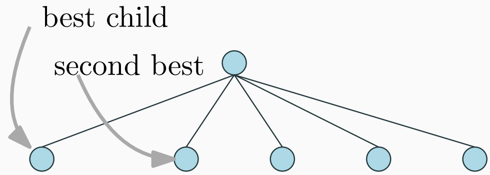
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

Limited Discrepancy Search (LDS) - key idea

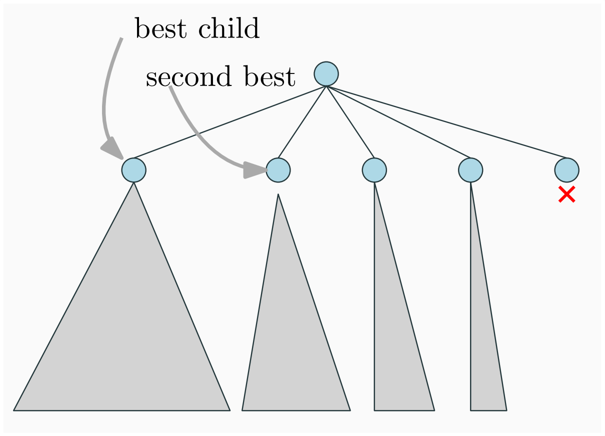
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

Limited Discrepancy Search (LDS) - key idea

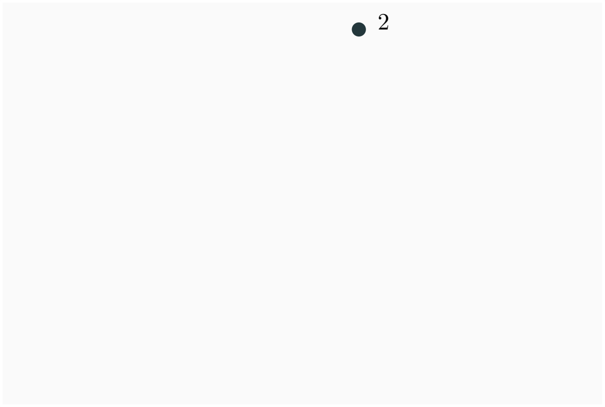
Correct DFS drawback: early bad decisions



Explore more the most promising but still keep exploring others

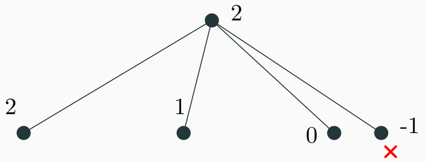
Limited Discrepancy Search (LDS) - An example: D=2

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



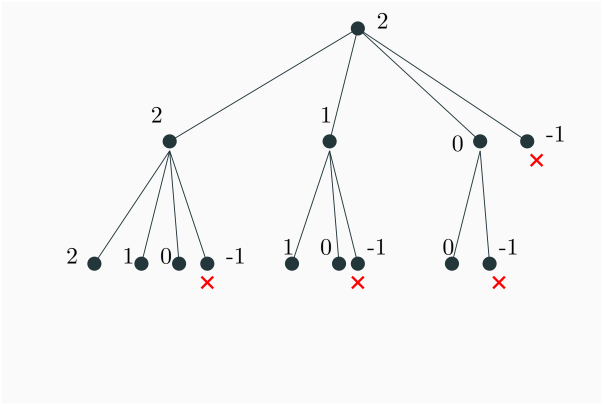
Limited Discrepancy Search (LDS) - An example: D=2

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



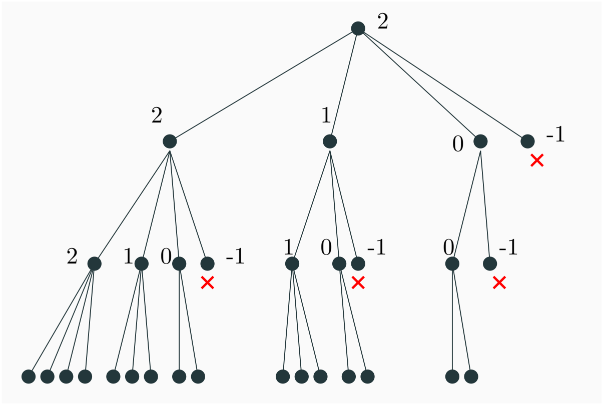
Limited Discrepancy Search (LDS) - An example: D=2

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

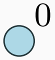


Limited Discrepancy Search (LDS) - An example: D=2

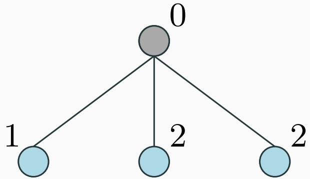
Count the number of deviations from the best child (discrepancies)



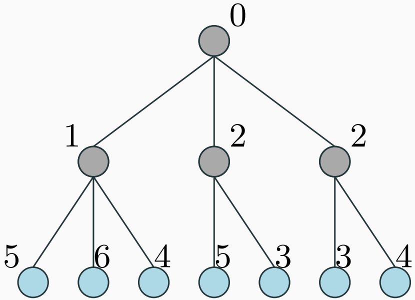
Beam Search (D=3)



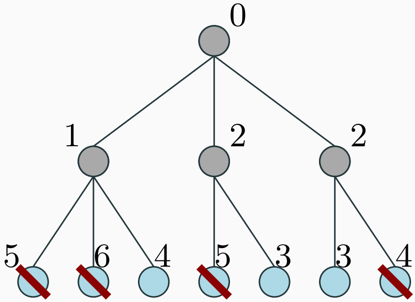
Beam Search (D=3)



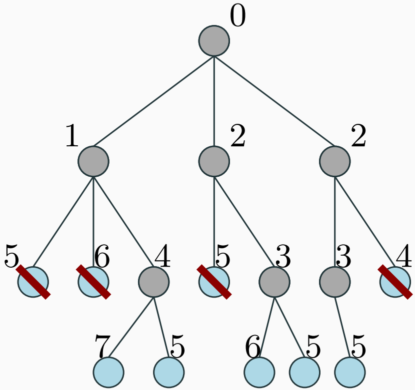
Beam Search (D=3)



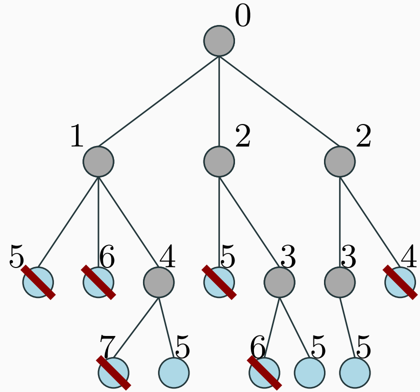
Beam Search (D=3)



Beam Search (D=3)



Beam Search (D=3)



Iterative Beam Search

- Start a beam search of size $D = 1$ (greedy)
- Then a beam search of size $D = 2$
- Then 4, 8, *etc.*

Iterative Beam Search

- Start a beam search of size $D = 1$ (greedy)
- Then a beam search of size $D = 2$
- Then 4, 8, *etc.*

A few properties:

- A **complete/exact** algorithm. When the beam is wide enough, no "heuristic" fathoming.

Iterative Beam Search

- Start a beam search of size $D = 1$ (greedy)
- Then a beam search of size $D = 2$
- Then 4, 8, *etc.*

A few properties:

- A **complete/exact** algorithm. When the beam is wide enough, no "heuristic" fathoming.
- The algorithm may open a node multiple times

Iterative Beam Search

- Start a beam search of size $D = 1$ (greedy)
- Then a beam search of size $D = 2$
- Then 4, 8, *etc.*

A few properties:

- A **complete/exact** algorithm. When the beam is wide enough, no "heuristic" fathoming.
- The algorithm may open a node multiple times

Theorem: Because of the exponential growth, no more than 2 times in average.

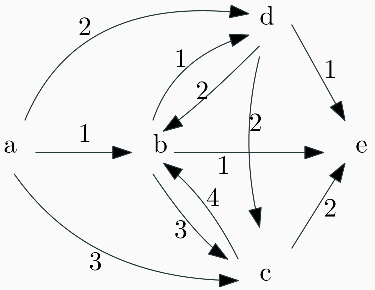
Sequential Ordering Problem

Sequential Ordering Problem (SOP)

Collaboration with
Abdel-Malik Bouhassoun, Hadrien Cambazard and Vincent Jost

SOP - problem definition

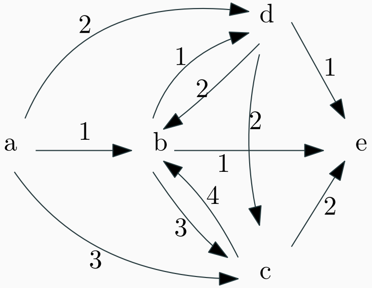
Asymmetric Traveling Salesman Problem with precedence constraints



- a before b, c, d, e
- a, b, c, d before e
- d before c

SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints

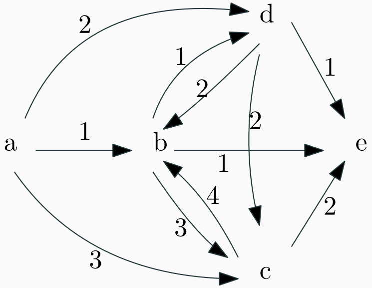


- a before b, c, d, e
- a, b, c, d before e
- d before c

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

SOP - problem definition

Asymmetric Traveling Salesman Problem with precedence constraints

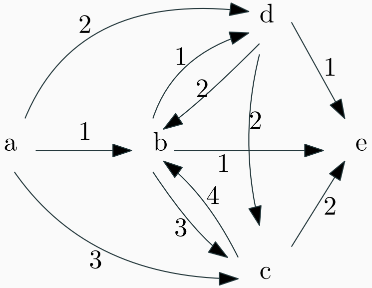


- a before b, c, d, e
- a, b, c, d before e
- d before c

- **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 before b, c, d, e
- a, b, c, d before e
- d before c

- **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 search (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 search (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

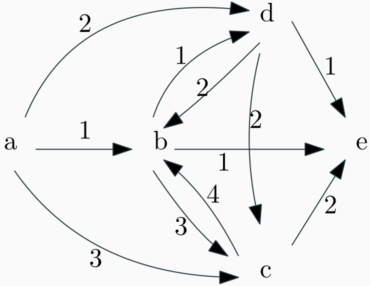
Two parts:

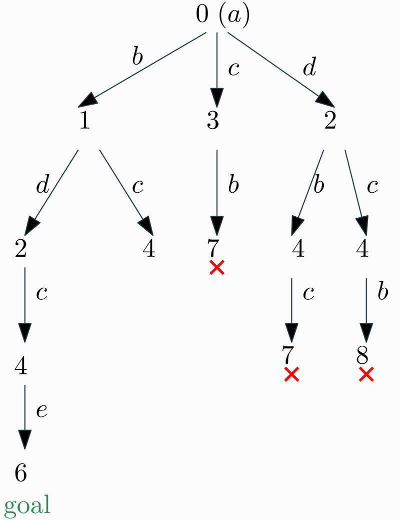
Implicit tree: how to branch, bounds ...

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

Implicit tree - Branching

Forward branching





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.

Prefix equivalence fathomings

Inspired from dynamic programming

Example, two partial equivalent[1] solutions:

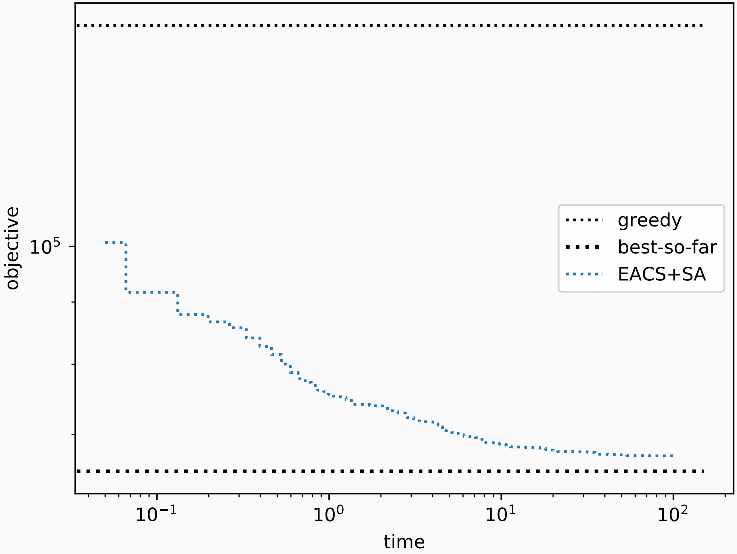
1. **a,b,c,d** cost 10
2. **a,c,b,d** cost 12

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

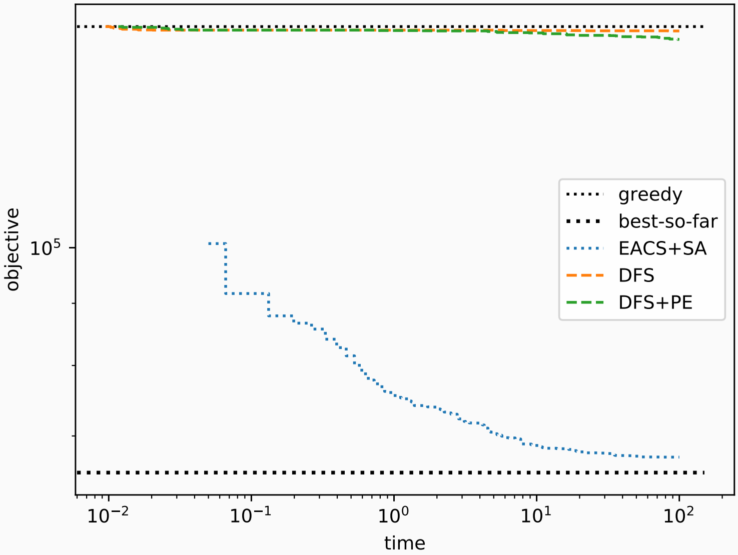
Results - Performance profiles on R.700.1000.15

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



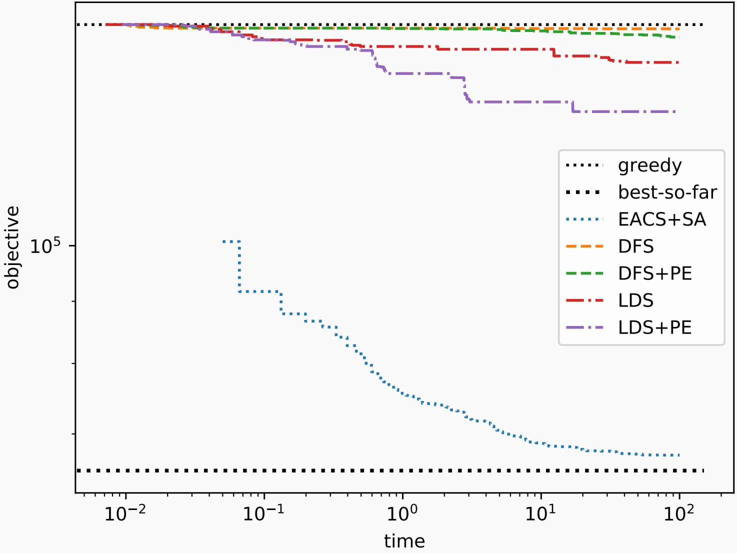
Results - Performance profiles on R.700.1000.15

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



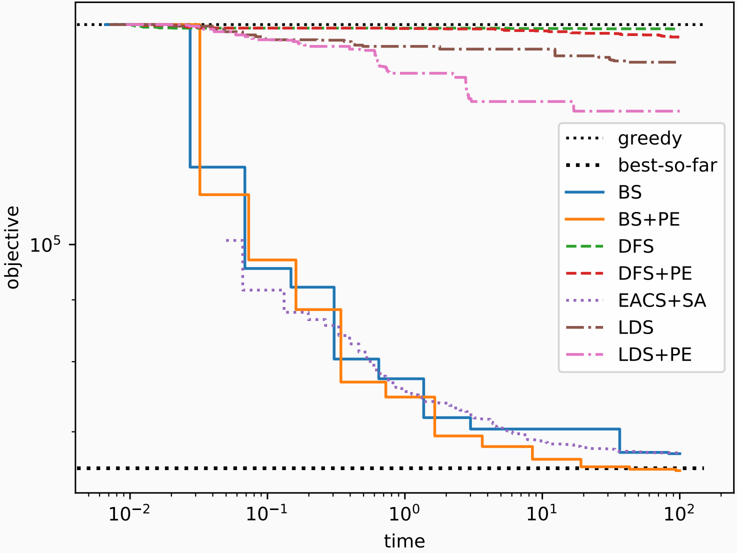
Results - Performance profiles on R.700.1000.15

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 search
- 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 search
- but (relatively) easy for constructive algorithms
- thus the need to consider anytime tree search

Other works

EURO/ROADEF challenge (glass cutting)

Collaboration with Florian Fontan

- Every two years, the french OR society organizes an optimization challenge in collaboration with a company.

EURO/ROADEF challenge (glass cutting)

Collaboration with Florian Fontan

- Every two years, the french OR society organizes an optimization challenge in collaboration with a company.
- last year's challenge was about optimizing *Saint Gobain*'s glass cutting process

EURO/ROADEF challenge (glass cutting)

Collaboration with Florian Fontan

- Every two years, the french OR society organizes an optimization challenge in collaboration with a company.
- last year's challenge was about optimizing *Saint Gobain*'s glass cutting process
- We won the 3 final phase's prizes with a tree search algorithm

EURO/ROADEF challenge (glass cutting)

Collaboration with Florian Fontan

- Every two years, the french OR society organizes an optimization challenge in collaboration with a company.
- last year's challenge was about optimizing *Saint Gobain*'s glass cutting process
- We won the 3 final phase's prizes with a tree search algorithm
- We later generalized our algorithm on multiple variants and obtained excellent results.

Longest Common Sequence

Collaboration with Vincent Jost and Aurélien Secardin

- large application range. For instance in molecular biology, gene recognition, pattern matching, *etc.*

Longest Common Sequence

Collaboration with Vincent Jost and Aurélien Secardin

- large application range. For instance in molecular biology, gene recognition, pattern matching, *etc.*
- many tree search algorithms in the literature to solve this problem

Longest Common Sequence

Collaboration with Vincent Jost and Aurélien Secardin

- large application range. For instance in molecular biology, gene recognition, pattern matching, *etc.*
- many tree search algorithms in the literature to solve this problem
- We show that the (simple) Iterative Beam Search obtains state-of-the-art performance compared to more intricate methods

Take aways

Tree search formalism

- Model the implicit search tree
- Then, perform some tree search algorithm

Are tree suited on large instances?

YES!

- If the tree size is (relatively) small

Are tree suited on large instances?

YES!

- If the tree size is (relatively) small
- Some constraints favor tree search (for instance precedences)

Are tree suited on large instances?

YES!

- If the tree size is (relatively) small
- Some constraints favor tree search (for instance precedences)
- Good greedy guides

Tree search strategy

The choice is crucial

- **Iterative beam search** seems to work well in most scenarios

References

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

Anytime tree search for discrete optimization

A tutorial

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

April, 28, 2020 - ROSP Seminar

G-SCOP, Grenoble, France
email: luc.libralesso@grenoble-inp.fr