A Heuristic Branch and Bound for the EURO/ROADEF 2018 challenge

Luc Libralesso, Florian Fontan

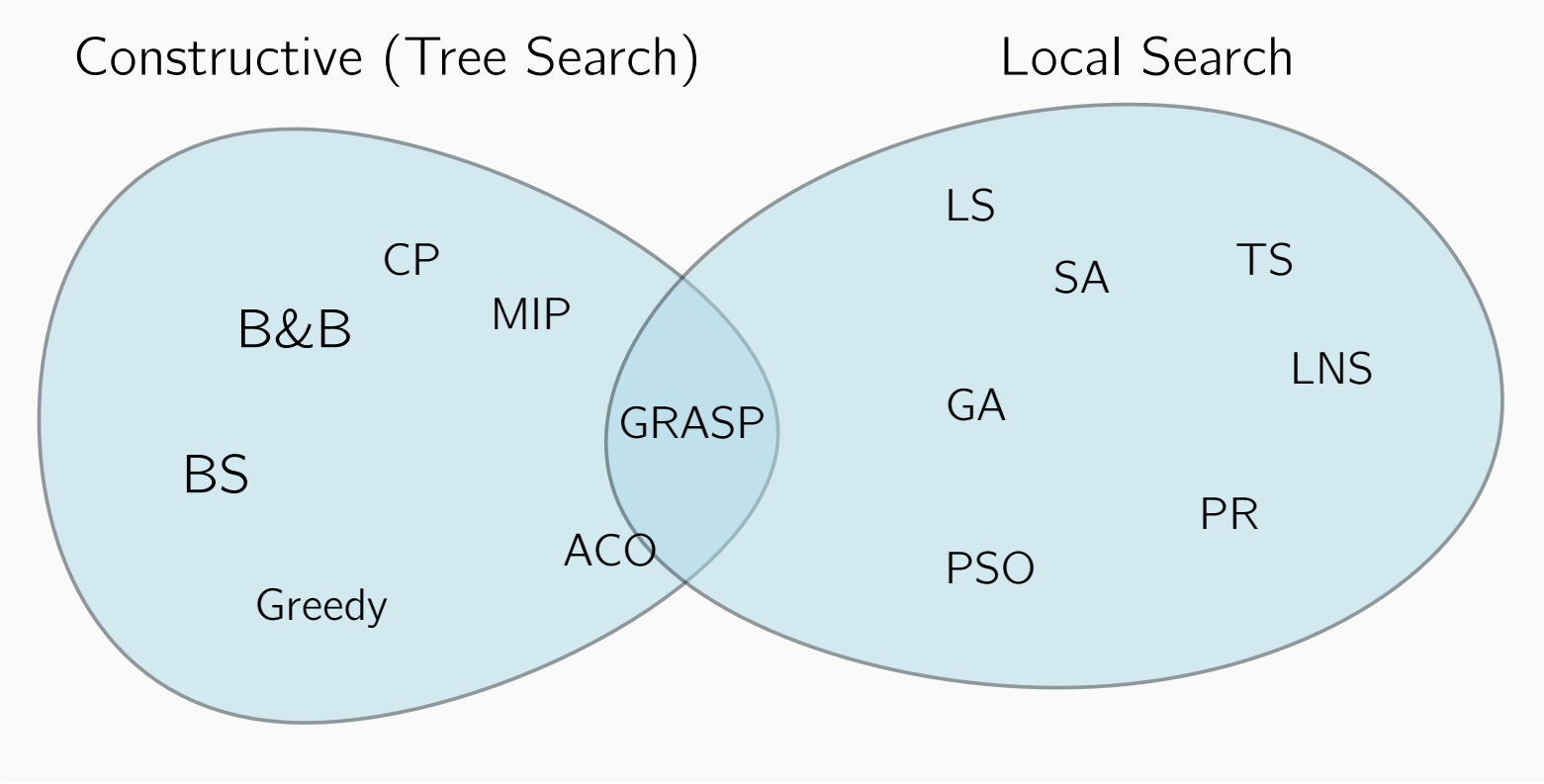
June 24, 2019

G-SCOP

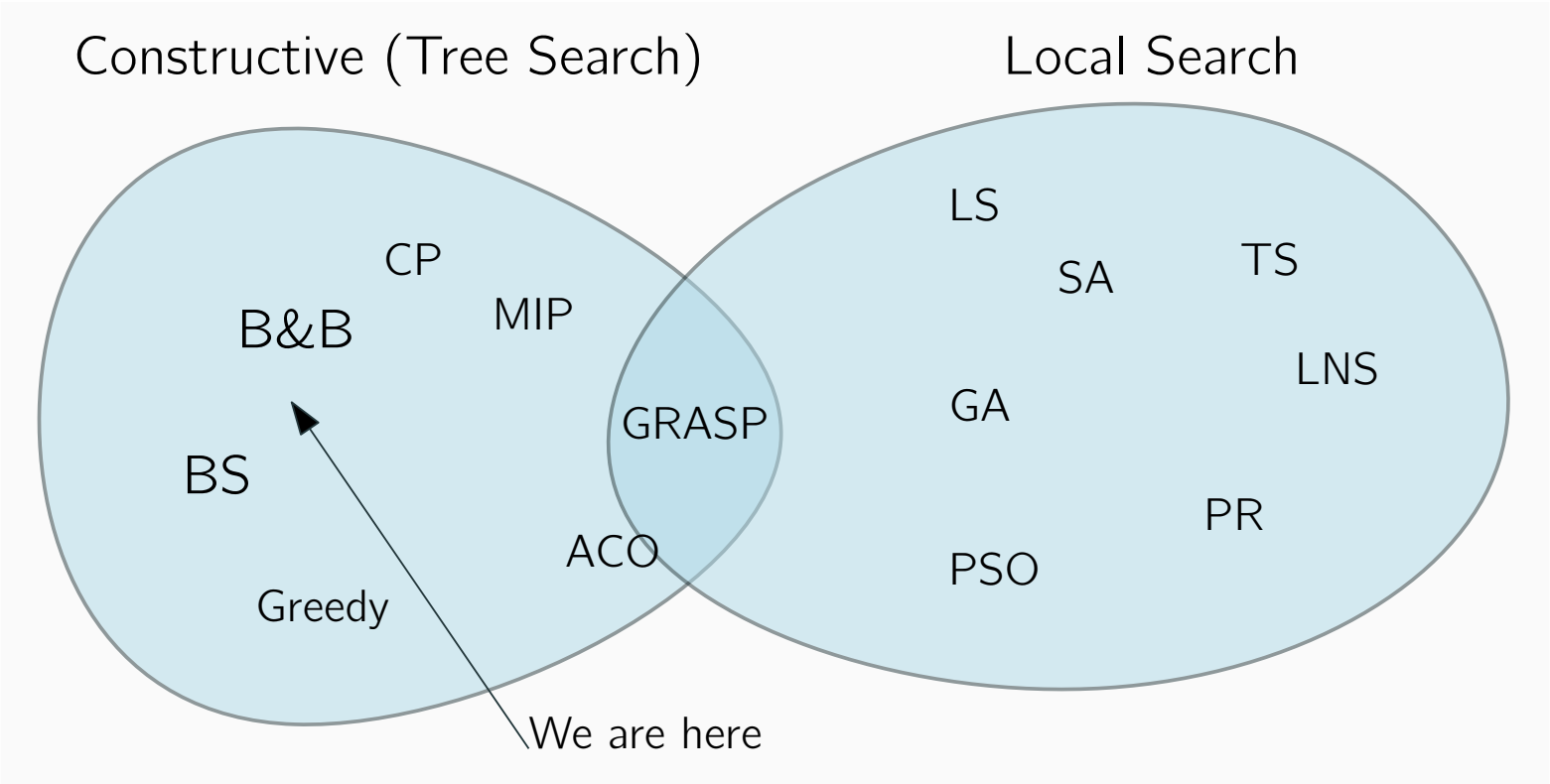
Table of contents

Methodology

Constructive vs Local Search



Constructive vs Local Search



Our method integrates parts of *Branch and bounds* and *Beam Search*

Anatomy of a Tree Search

Tree Searches are made of two parts:

Anatomy of a Tree Search

Tree Searches are made of two parts:

- the Branching Scheme (*i.e. problem specific*):
  - root definition
  - children generation
  - bounds
  - *etc.*

Anatomy of a Tree Search

Tree Searches are made of two parts:

- the Branching Scheme (*i.e. problem specific*):
  - root definition
  - children generation
  - bounds
  - *etc.*
- the Search strategy (*i.e. generic*):
  - DFS, A*, Best First, Beam Search, LDS, *etc.*

Anatomy of a Tree Search

Tree Searches are made of two parts:

- the Branching Scheme (*i.e. problem specific*):
  - root definition
  - children generation
  - bounds
  - *etc.*
- the Search strategy (*i.e. generic*):
  - DFS, A*, Best First, Beam Search, LDS, *etc.*
  - others known in AI/planning: SMA*, BULB, wA* *etc.*

Anatomy of a Tree Search

Tree Searches are made of two parts:

- the Branching Scheme (*i.e. problem specific*):
  - root definition
  - children generation
  - bounds
  - *etc.*
- the Search strategy (*i.e. generic*):
  - DFS, A*, Best First, Beam Search, LDS, *etc.*
  - others known in AI/planning: SMA*, BULB, wA* *etc.*

We developed our algorithm using this principle.

Our algorithm

- **the Branching Scheme** (*i.e. problem specific*)
- the Search strategy (*i.e. generic*)

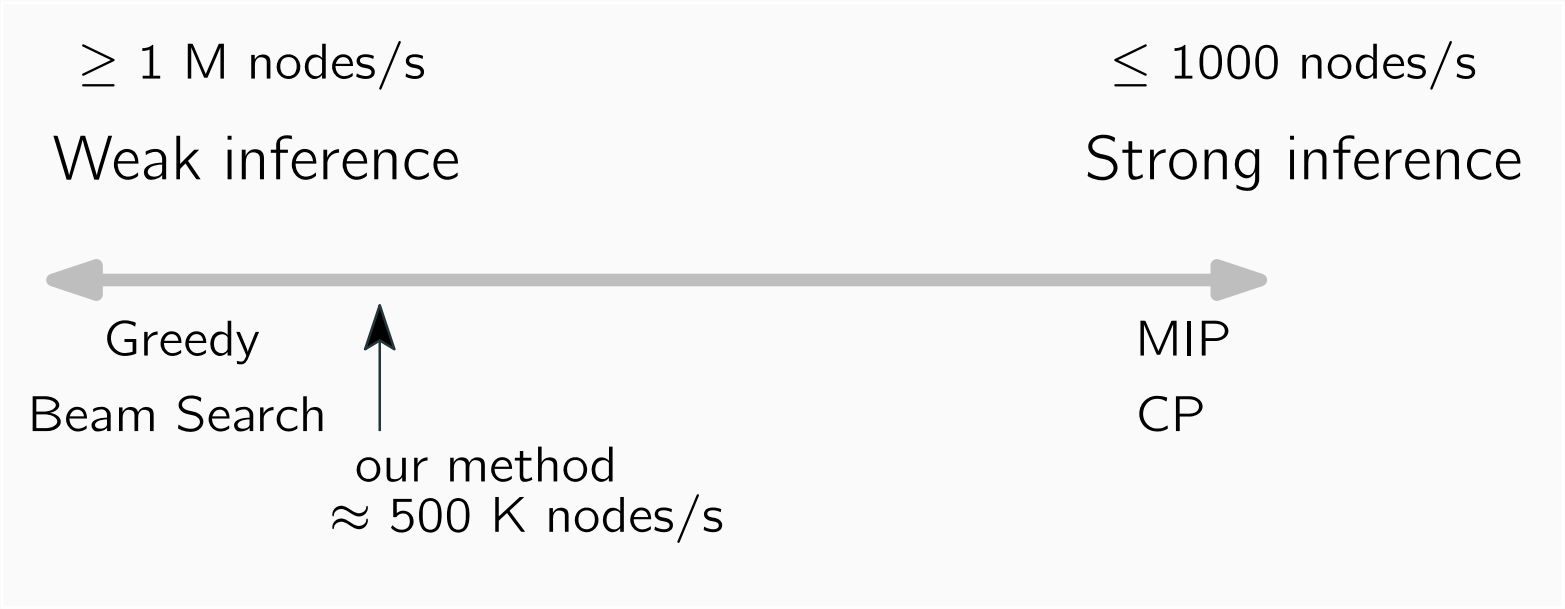
Branching Scheme

Node inference trade-off

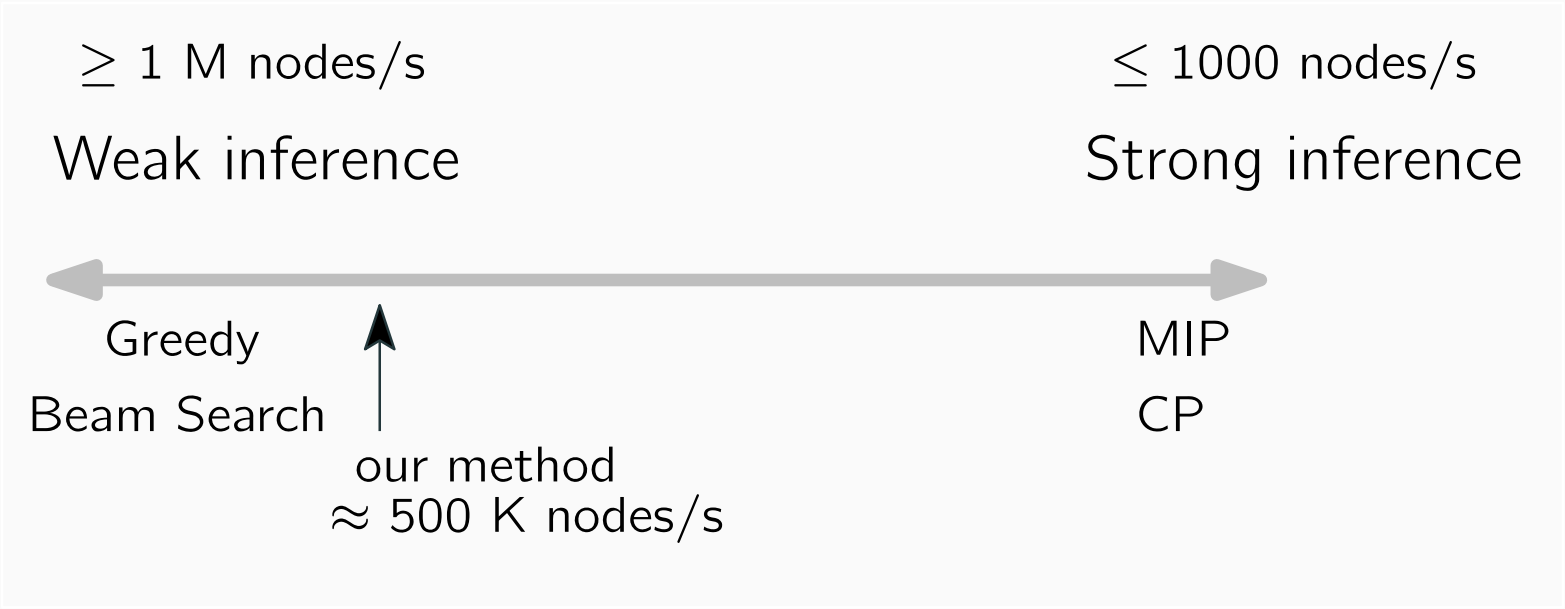
$\geq$ 1 M nodes/s	$\leq$ 1000 nodes/s
Weak inference	Strong inference

Node inference trade-off

Node inference trade-off

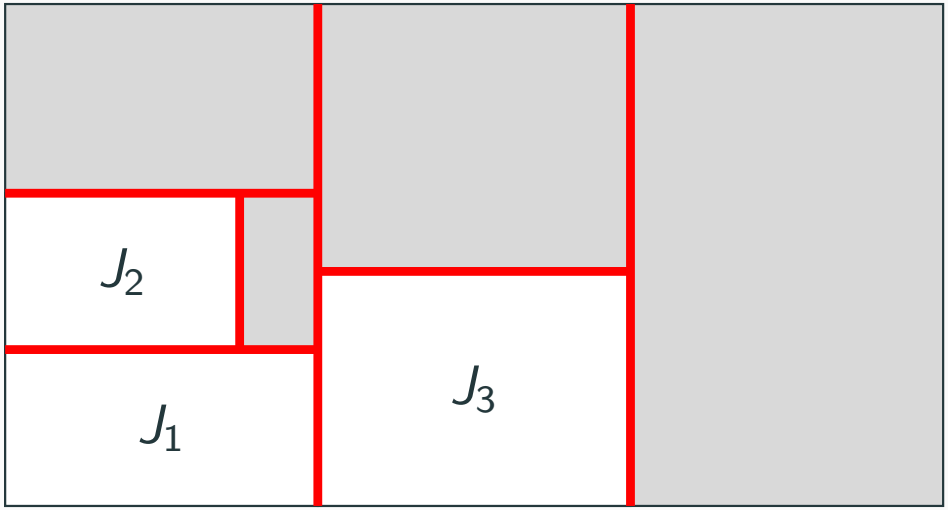


Node inference trade-off

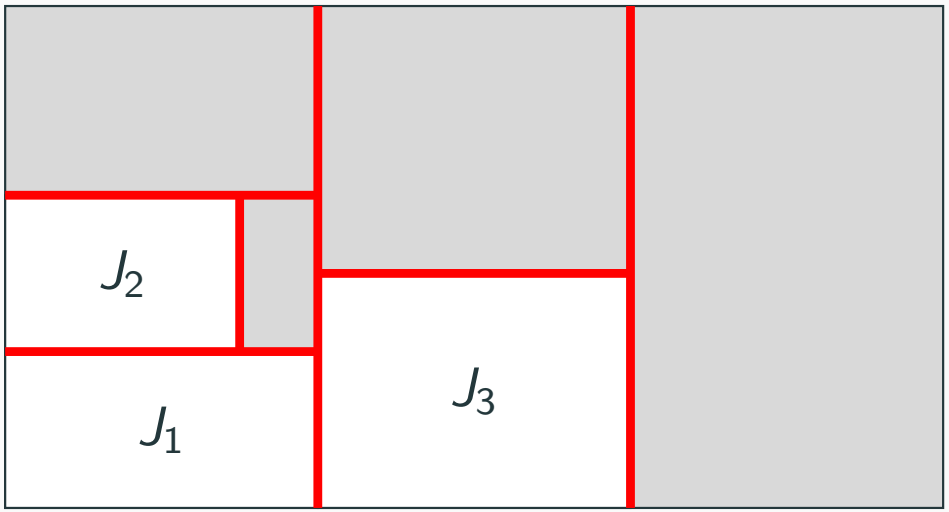


- We integrate quick bounds, symmetry breaking, dominance checking
- The idea of integrating Branch and Bound parts into Beam Searches can be found in [STDC18]

Packing in the bottom left corner



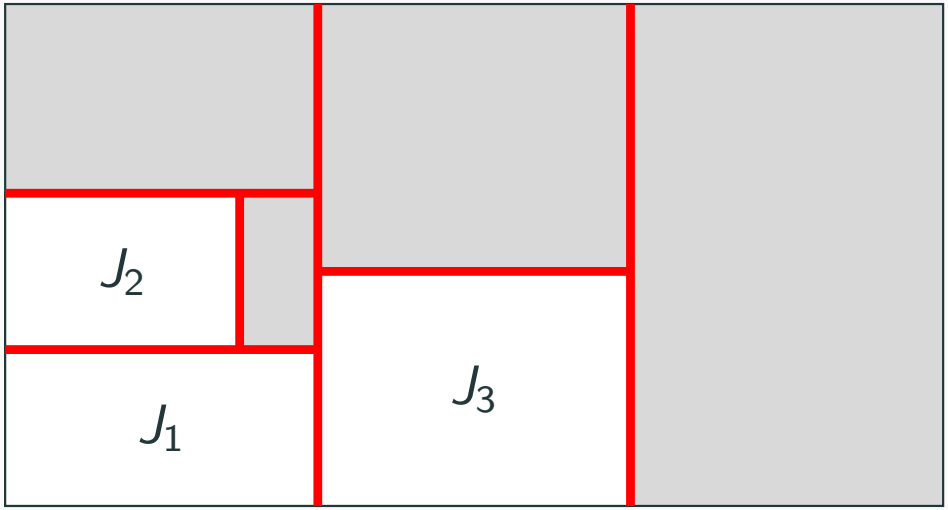
Packing in the bottom left corner



We prove that it is optimal if:

- guillotine and defects and precedence only
- guillotine and min waste only

Packing in the bottom left corner



We prove that it is optimal if:

- guillotine and defects and precedence only
- guillotine and min waste only

We prove it is not if:

- guillotine and min waste and precedences
- guillotine and min waste and defects

Not dominant in the challenge

Since guillotine and min waste and precedences and defects constraints.

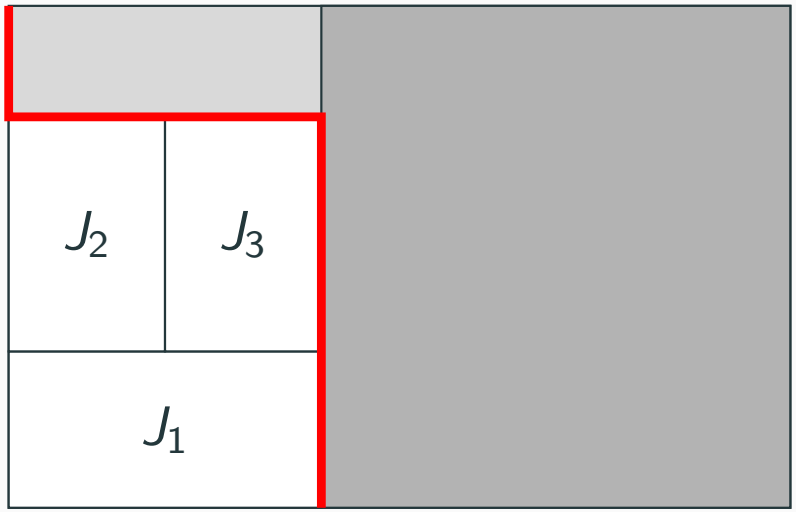
Good news - It still works very well !

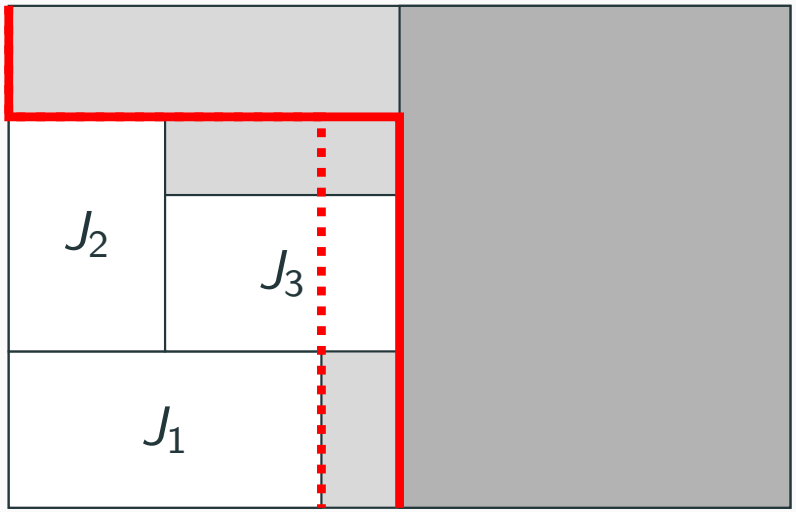
We only need good solutions, so we make a heuristic Branch and Bound.

How to construct children

- Root node: empty solution
- Children: all possible items in all possible positions (*i.e.* new plate, new 1-cut, new 2-cut, new 3-cut or new 4-cut, rotations, defect avoidance)

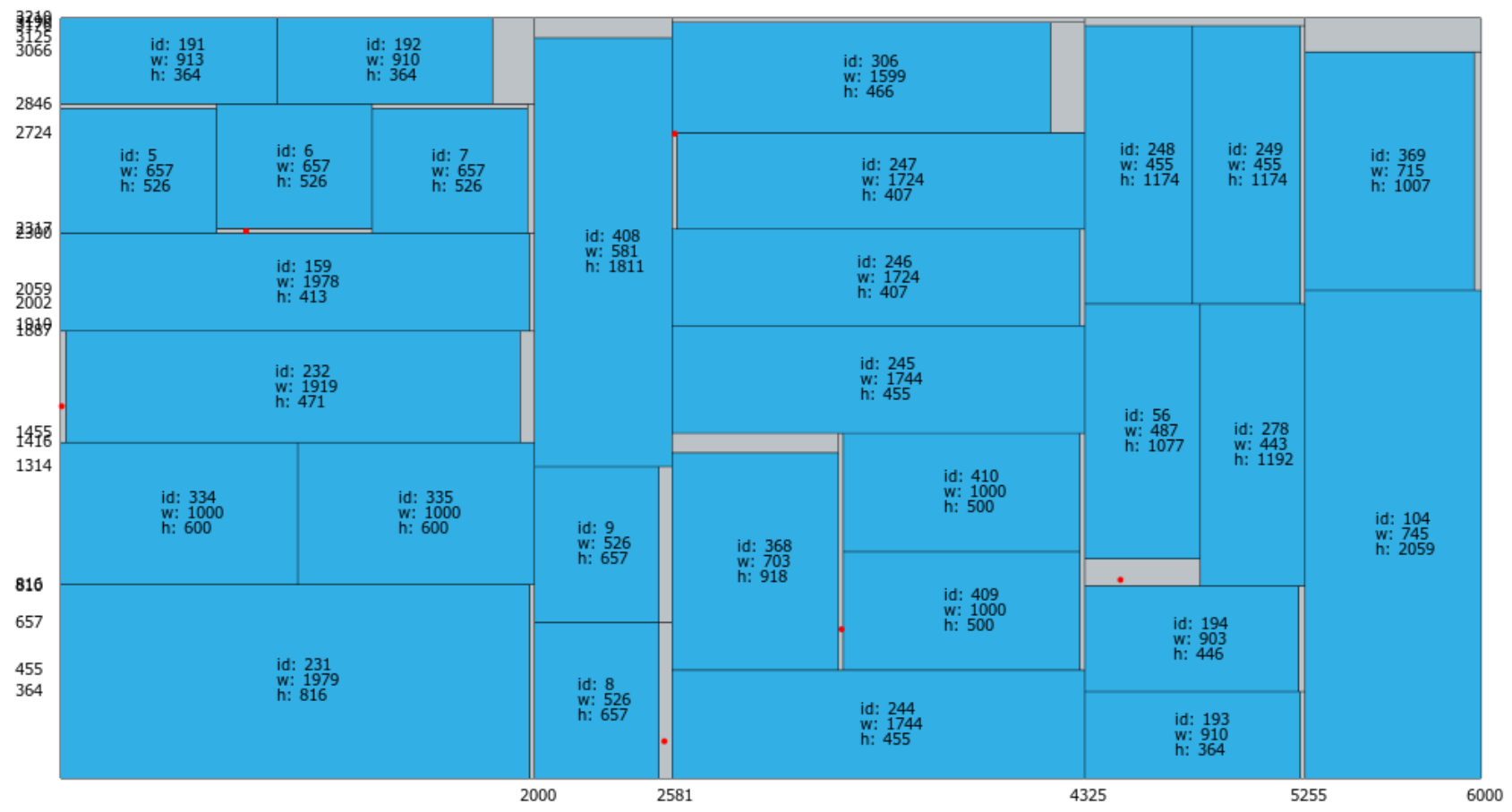
Pseudo dominance





Symmetry breaking

- Symmetry breaking strategy: for two consecutive blocks, the one with the smallest minimum item id comes before.

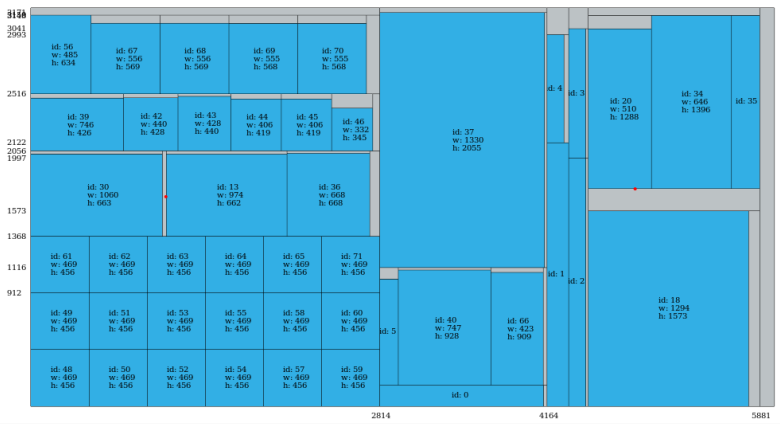


Node Goodness measure

Waste accumulated so far

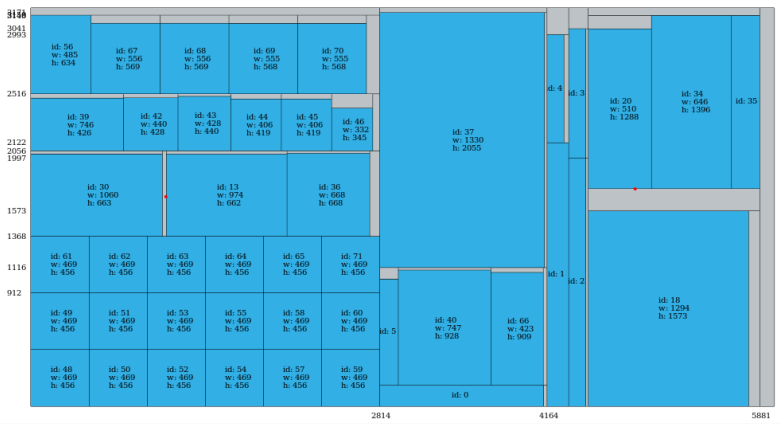
Node Goodness measure

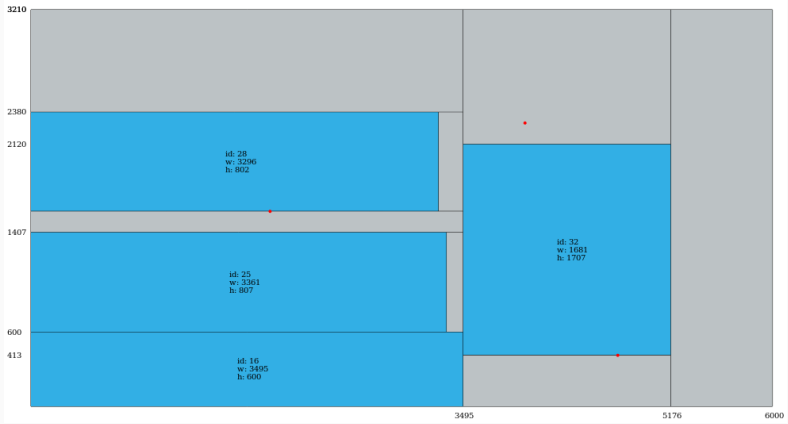
Waste accumulated so far



Node Goodness measure

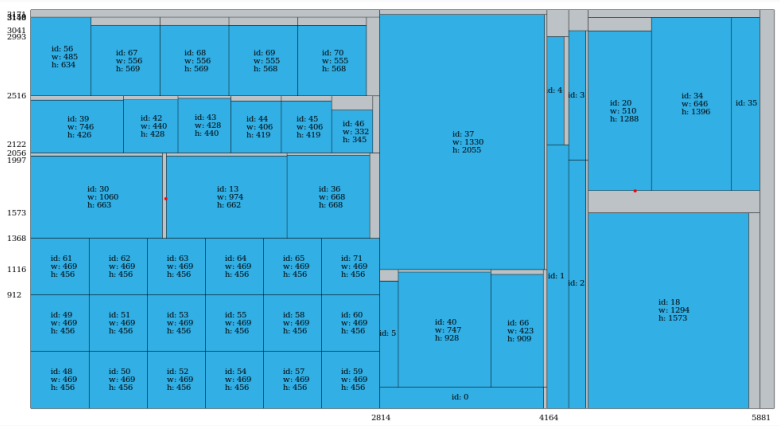
Waste accumulated so far

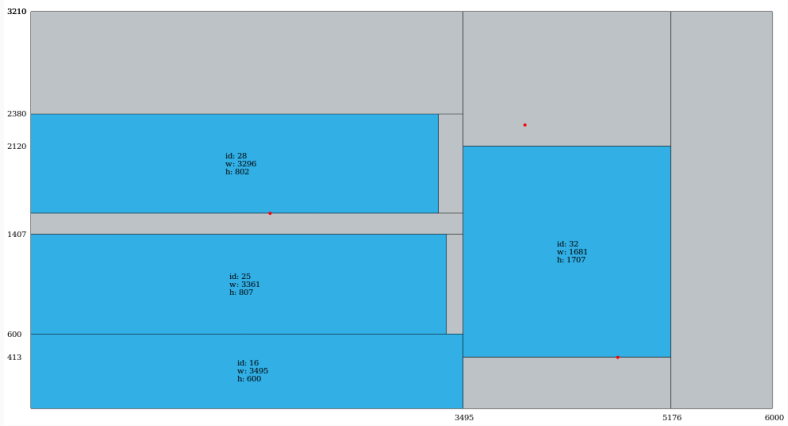




Node Goodness measure

Waste accumulated so far





Problem with waste:

- Small items at the beginning and big items at the end

A better node goodness measure

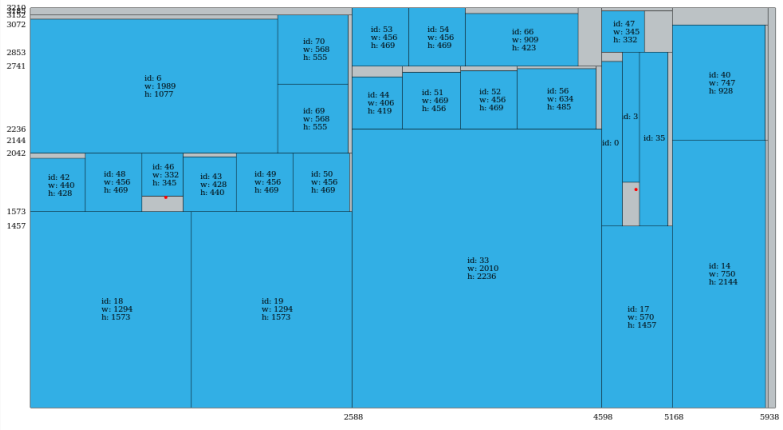
waste percentage

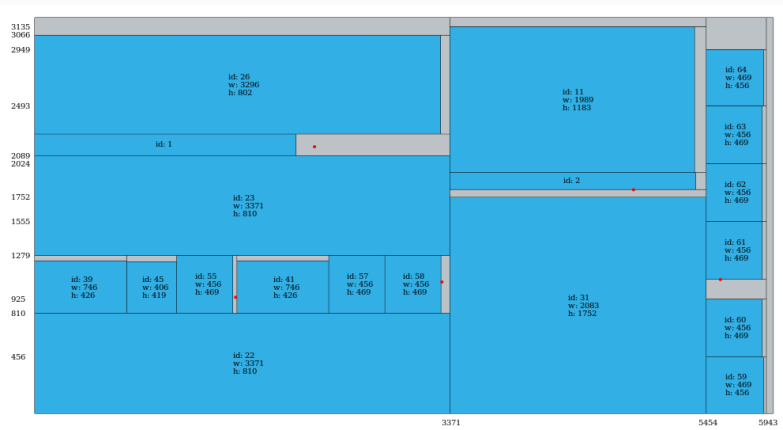
An even better node goodness measure

$$\frac{\text{waste}}{\text{total area} \cdot \text{mean area}}$$

An even better node goodness measure

$$\frac{\text{waste}}{\text{total area} \cdot \text{mean area}}$$





Search Strategy

Our algorithm

- the Branching Scheme (*i.e. problem specific*)
- **the Search strategy** (*i.e. generic , DFS, Best First, Beam Search, ...*)

MBA*

Inspired from *Beam Search* and *SMA**

MBA*

Inspired from *Beam Search* and *SMA**

- Best First strategy
- Delete some bad nodes if too many at the same time
- If finished, Restart with a bigger node limit D ($D_{n+1} \leftarrow D_n \times 2$)

MBA*

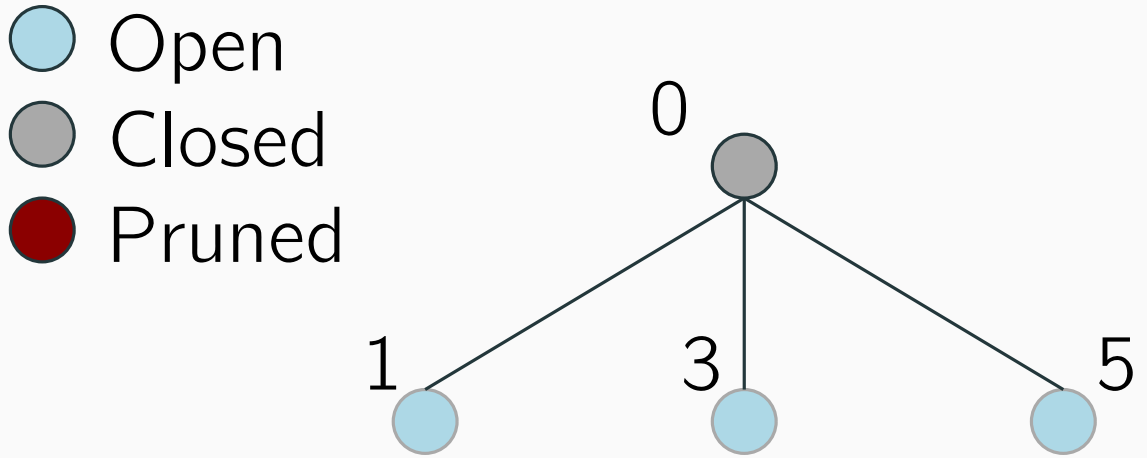
Inspired from *Beam Search* and *SMA**

- Best First strategy
- Delete some bad nodes if too many at the same time
- If finished, Restart with a bigger node limit D ($D_{n+1} \leftarrow D_n \times 2$)

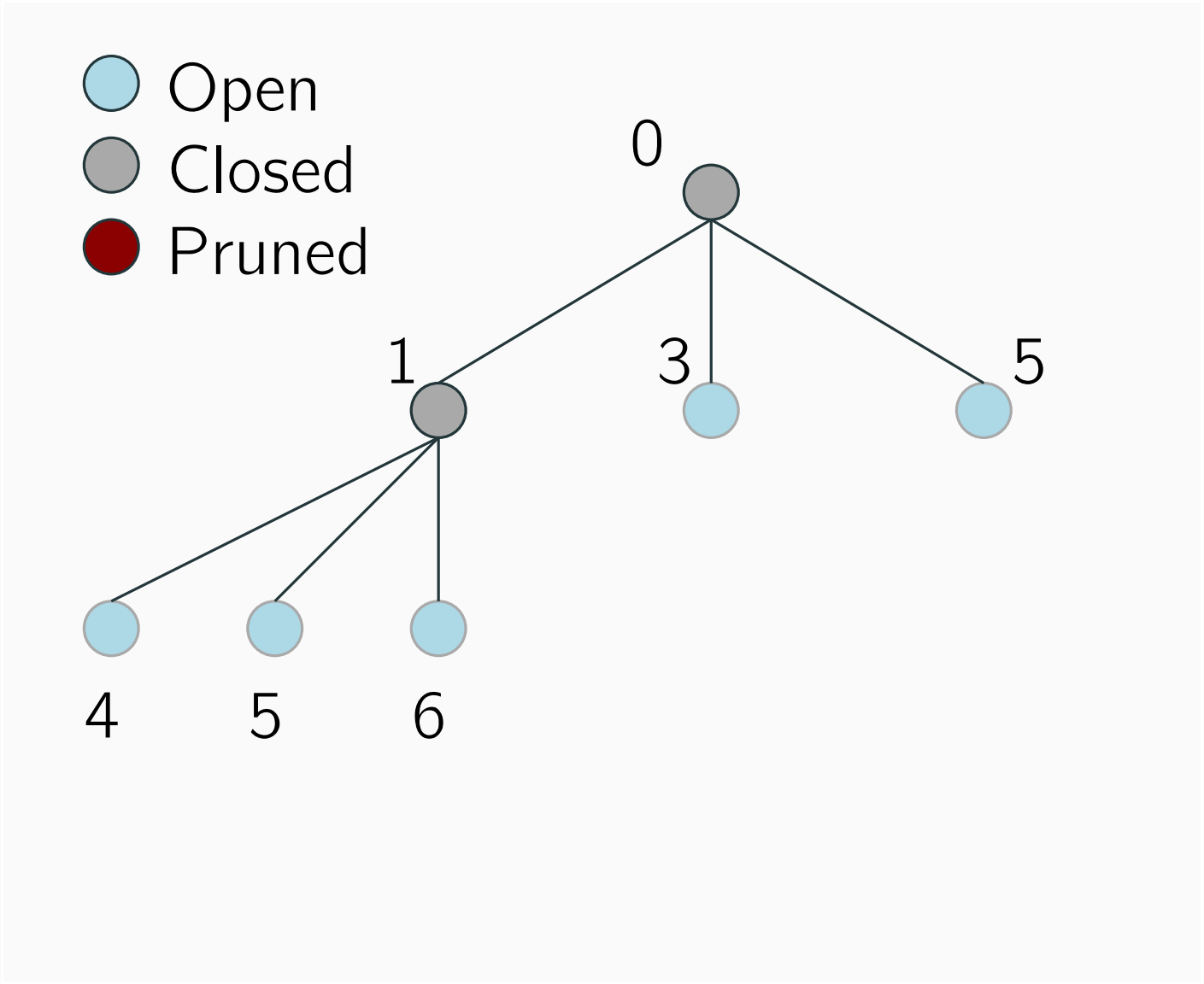
- at the beginning ($D = 1$), it behaves like a greedy algorithm
- at the end ($D \approx \infty$), it behaves like a Best First Search

MBA* - an example

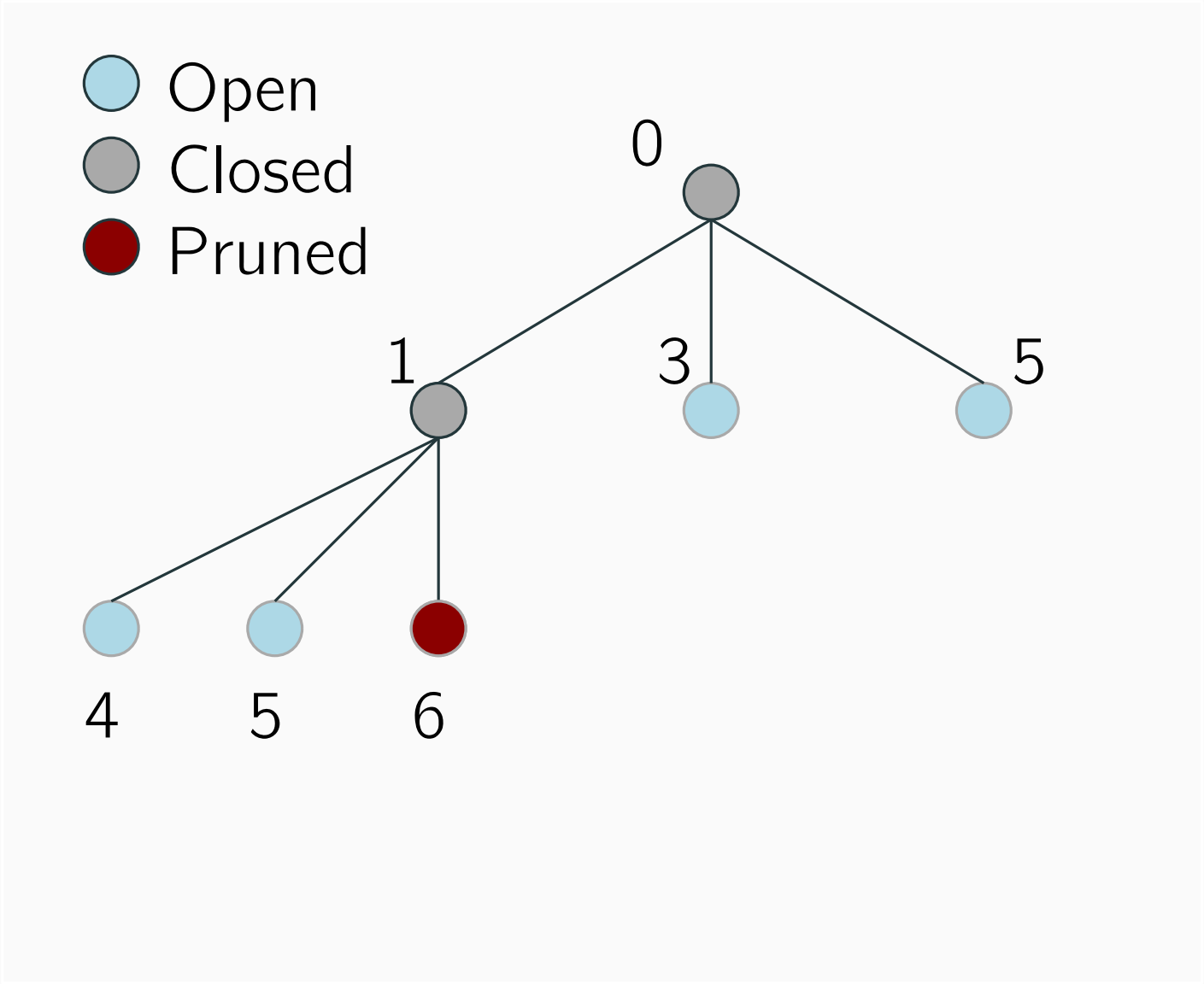
MBA* - an example



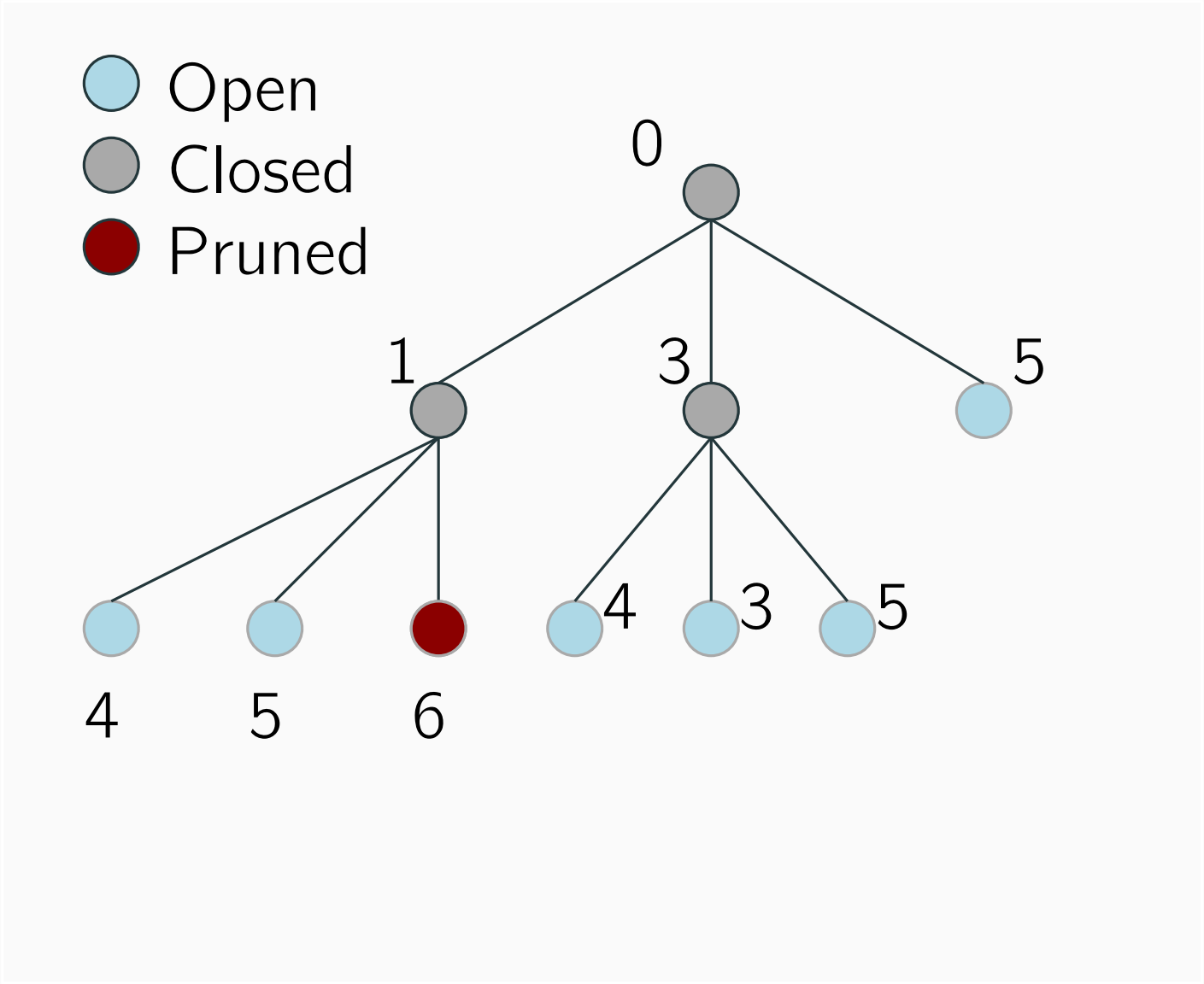
MBA* - an example



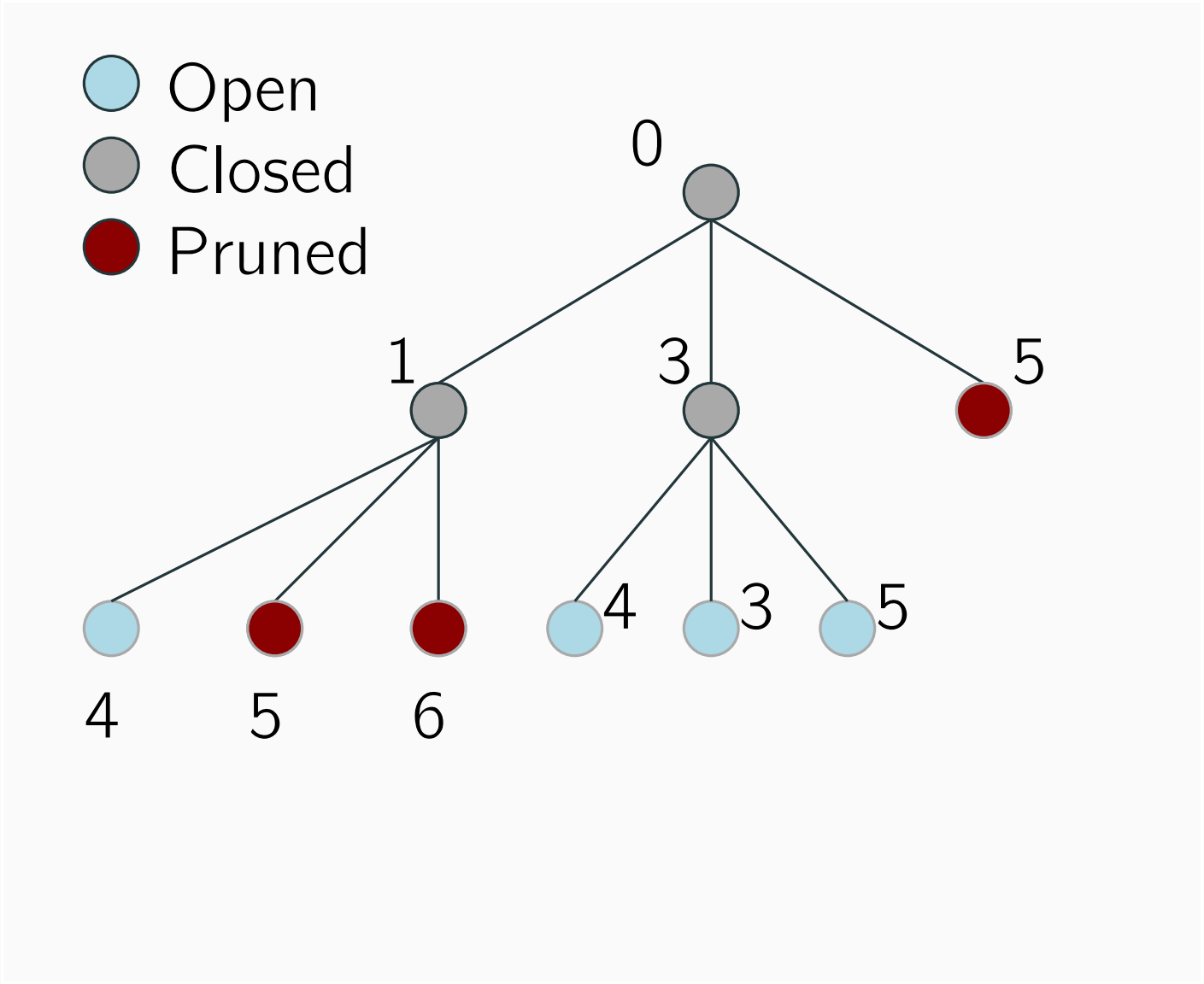
MBA* - an example



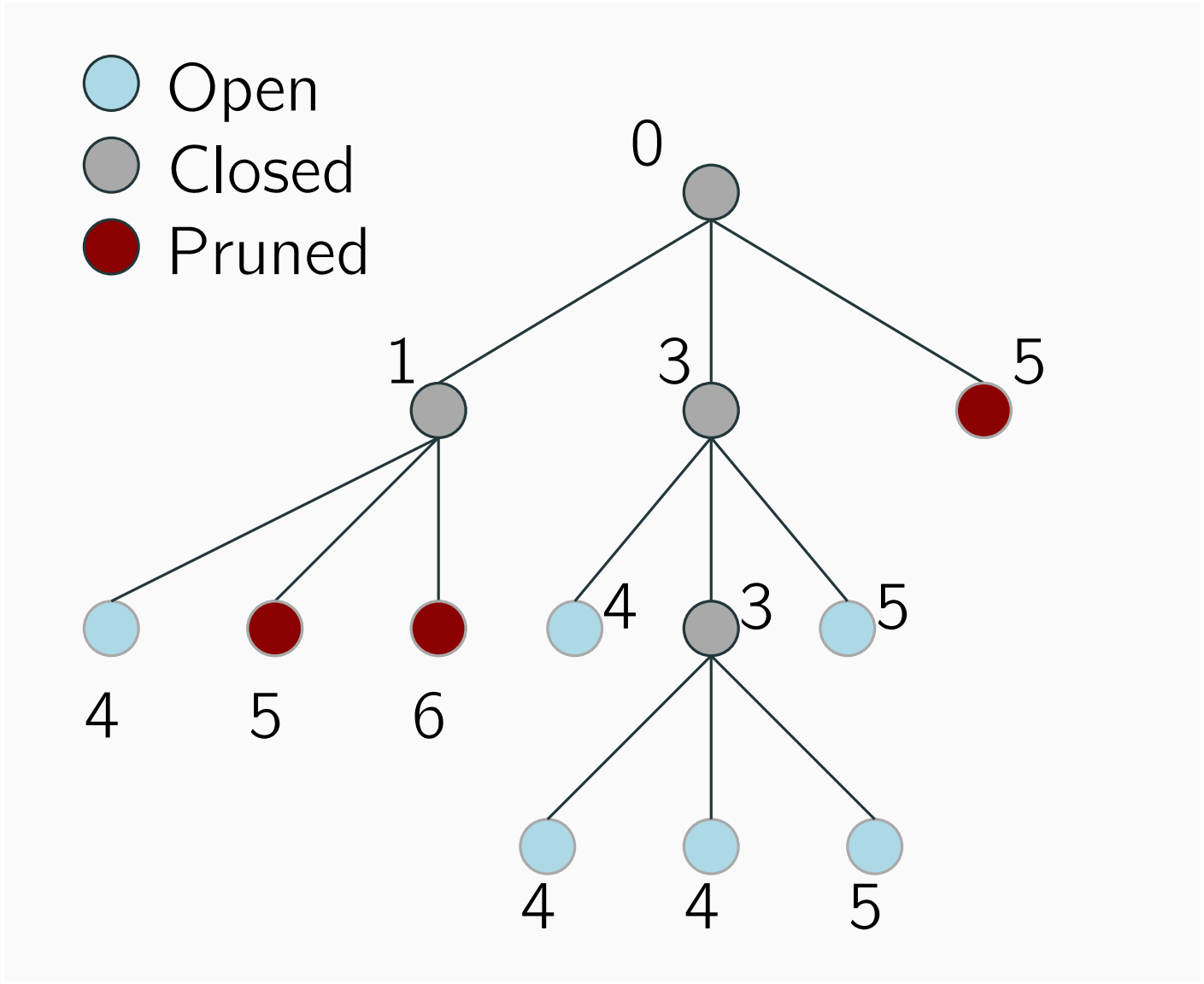
MBA* - an example



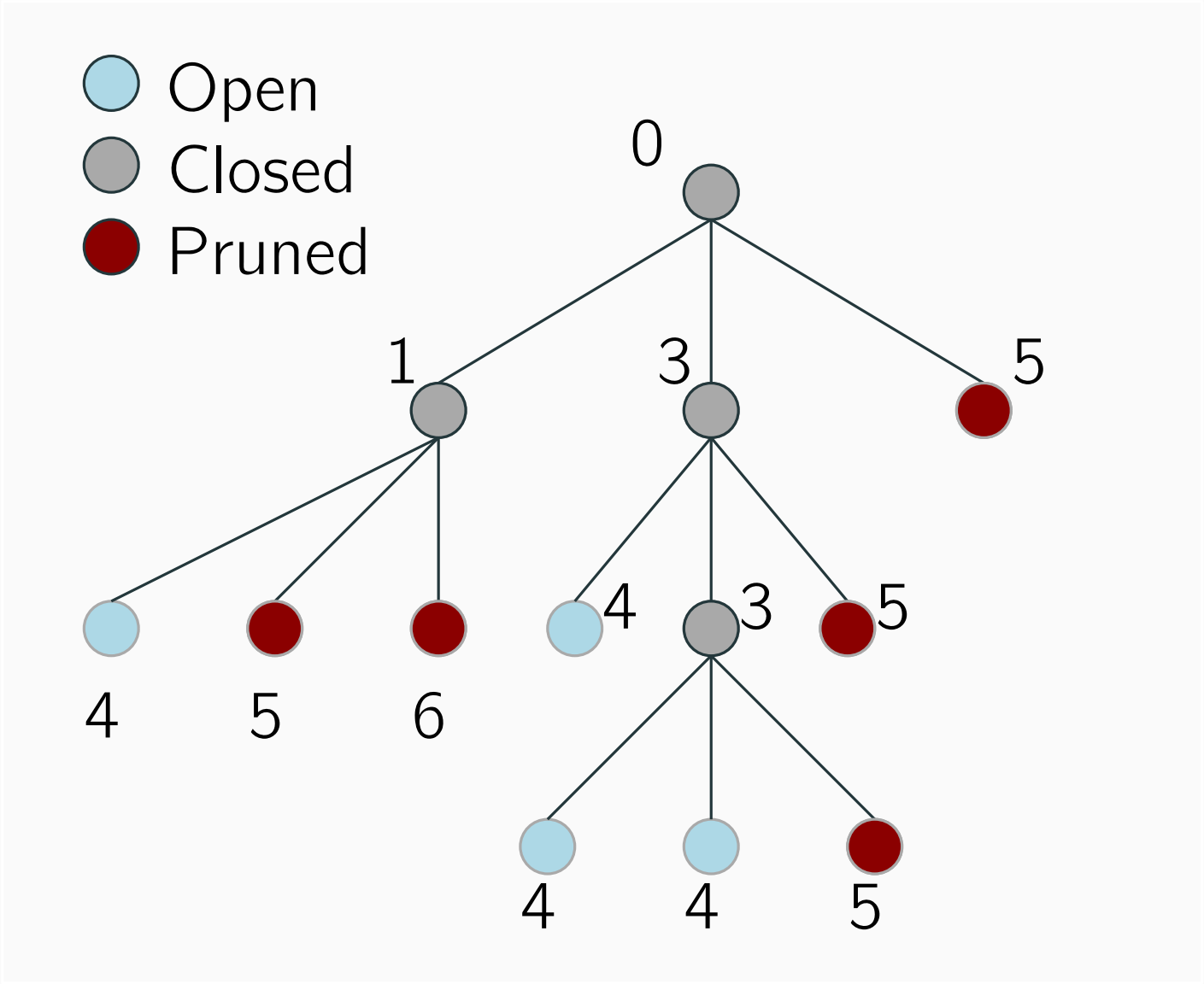
MBA* - an example



MBA* - an example

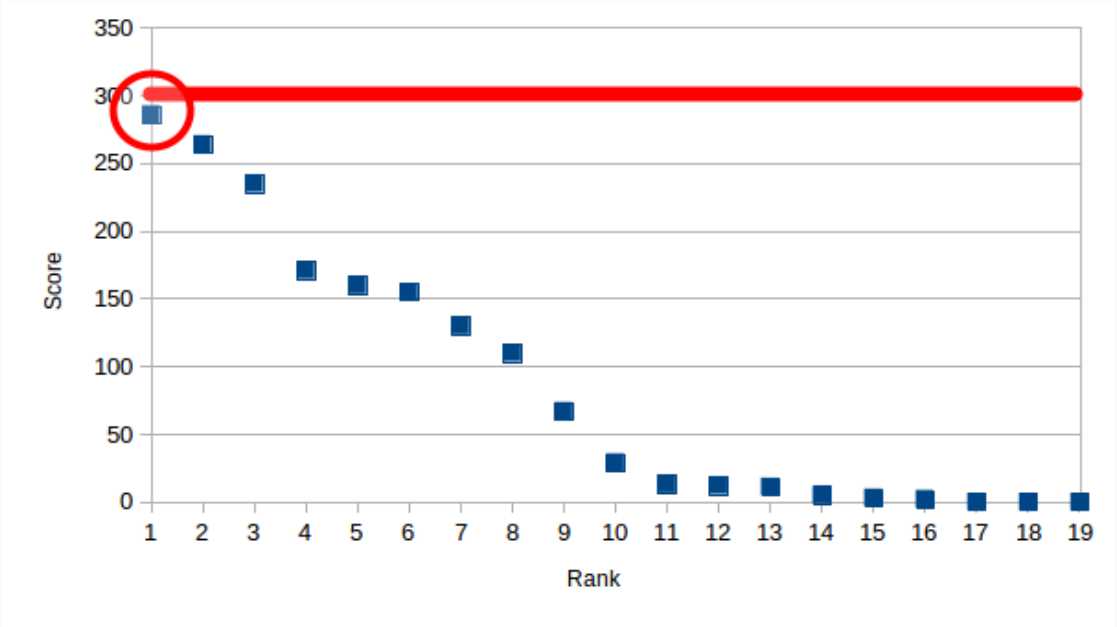


MBA* - an example

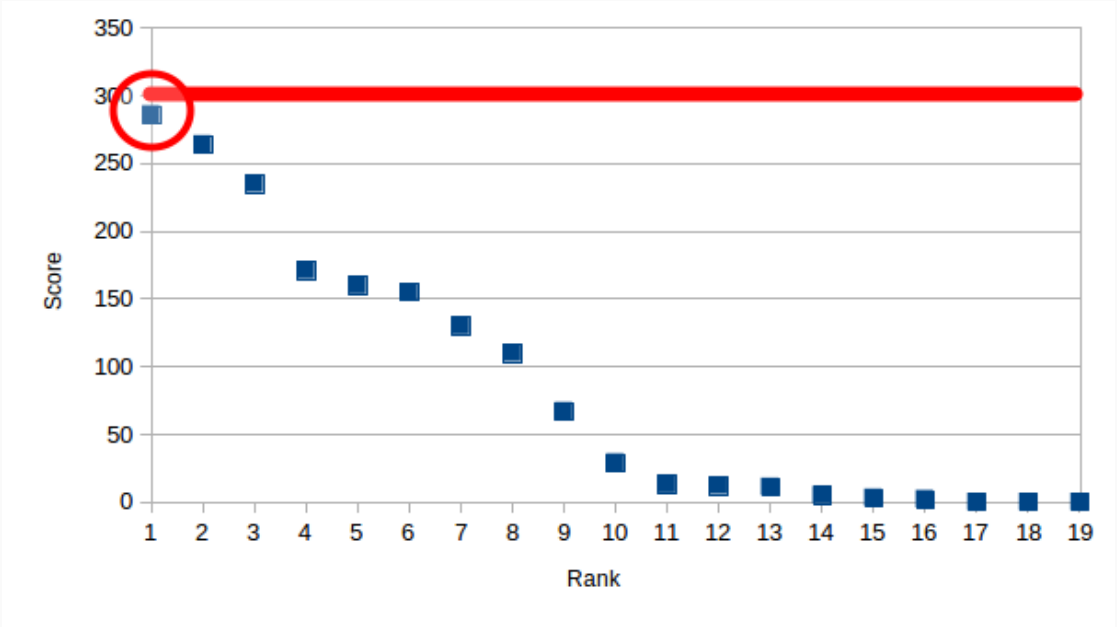


Results

performance

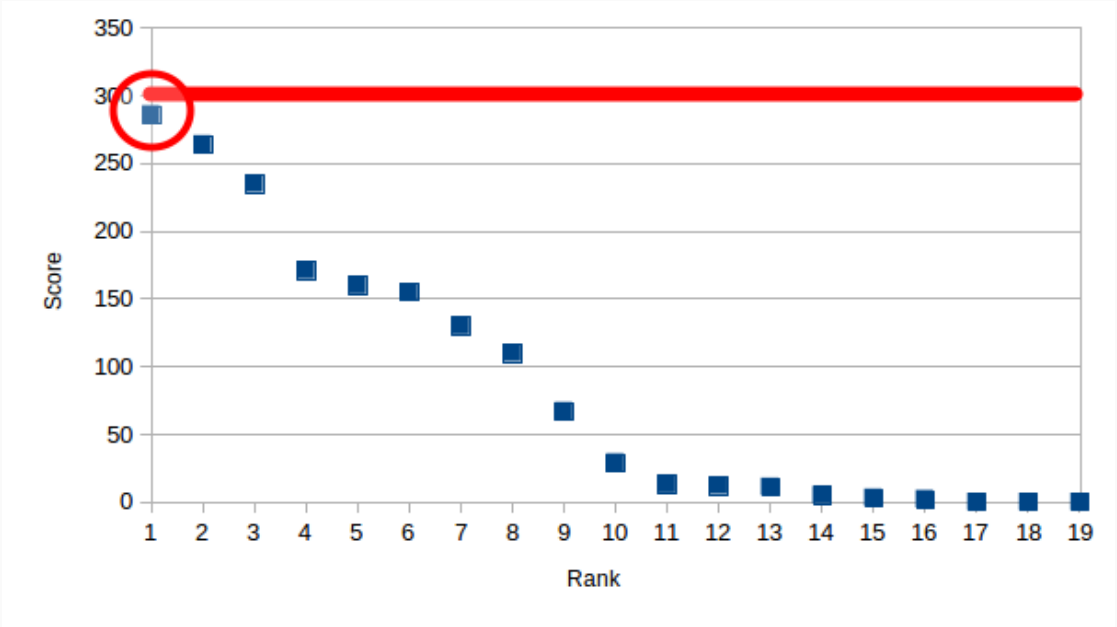


performance



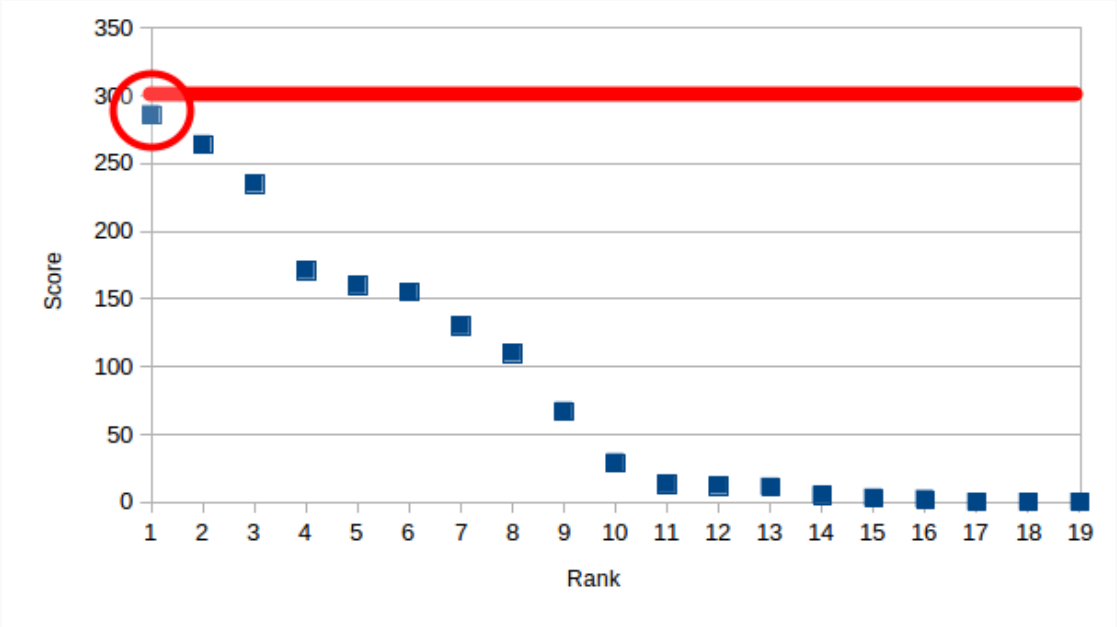
- Best solutions found on 20 over 30 instances.

performance



- Best solutions found on 20 over 30 instances.
- Total waste 2nd team: $506M$
- Total waste: $493M$ ($13M$ less)

performance



- Best solutions found on 20 over 30 instances.
- Total waste 2nd team: $506M$
- Total waste: $493M$ ($13M$ less)
- Total waste new version: $469M$ ($24M$ less than our submission)

Conclusions

- Simple and effective algorithm
- Tree searches can be competitive with other methods
- Decomposing the algorithm helps to identify good (and bad) parts

Conclusions

- Simple and effective algorithm
- Tree searches can be competitive with other methods
- Decomposing the algorithm helps to identify good (and bad) parts

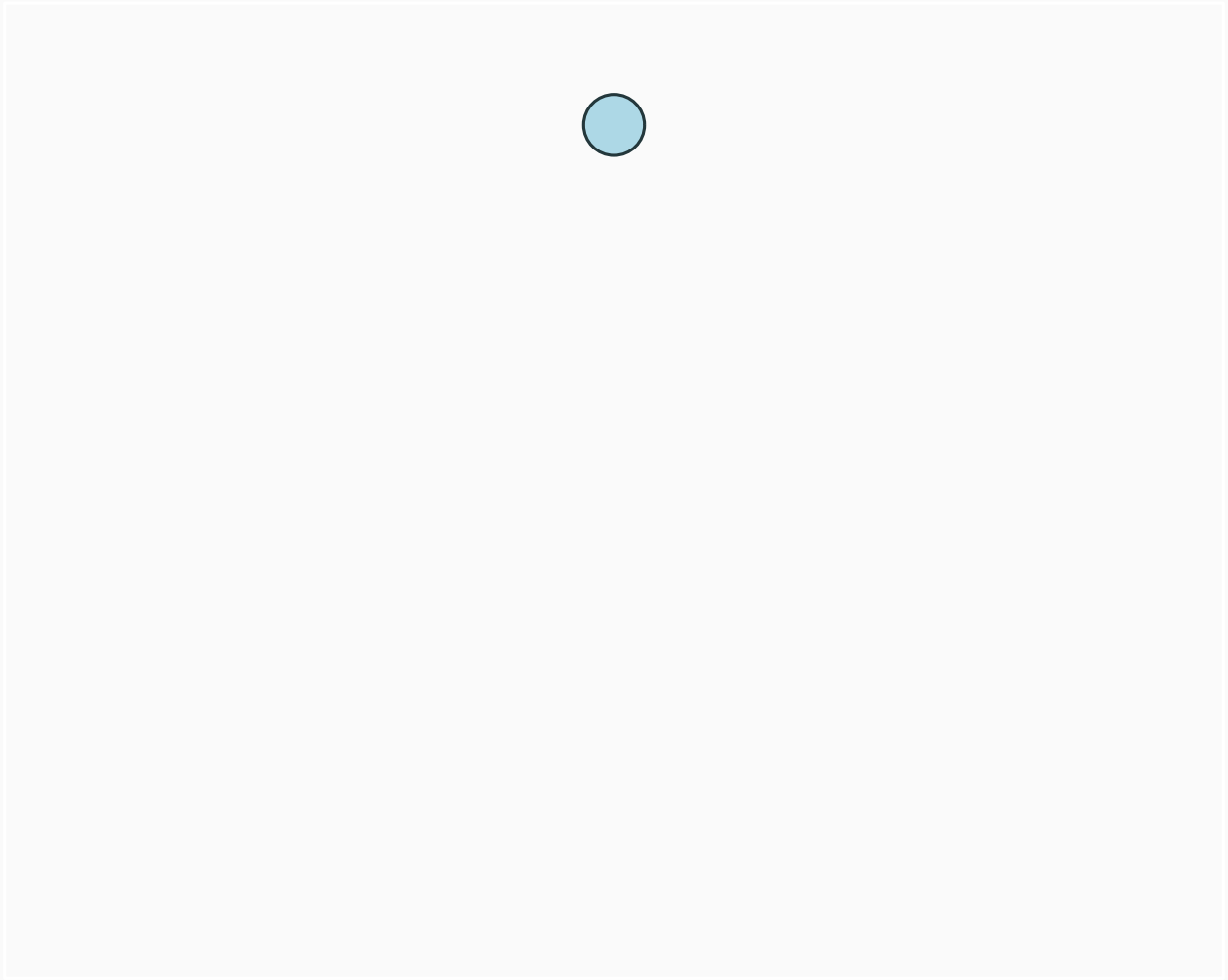
- We tried
  - several search strategies (DFS, Beam Search, LDS, and MBA*)
  - several guides
- Chose best combination

Questions or remarks ?

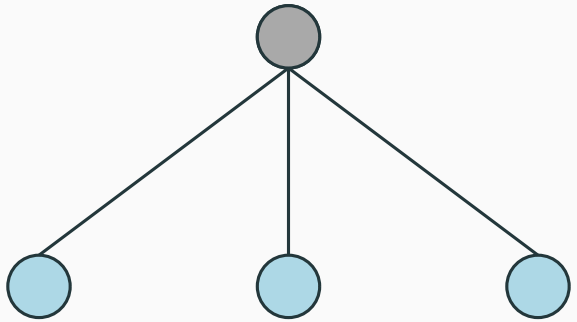
Bibliography


Lei Shang, Vincent T'Kindt, and Federico Della Croce.
The memorization paradigm: Branch & memorize algorithms for the efficient solution of sequencing problems.
2018.

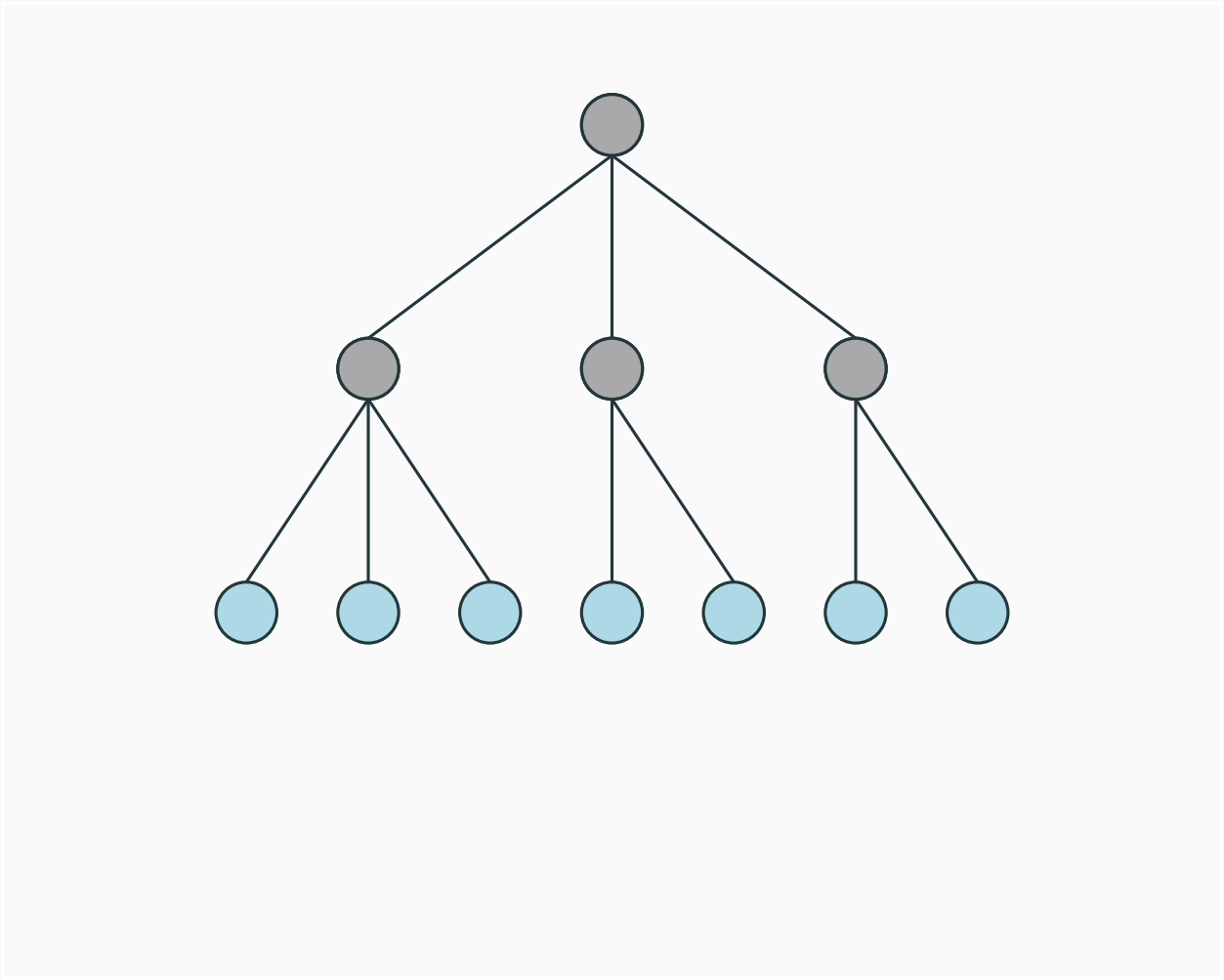
Breadth First Search



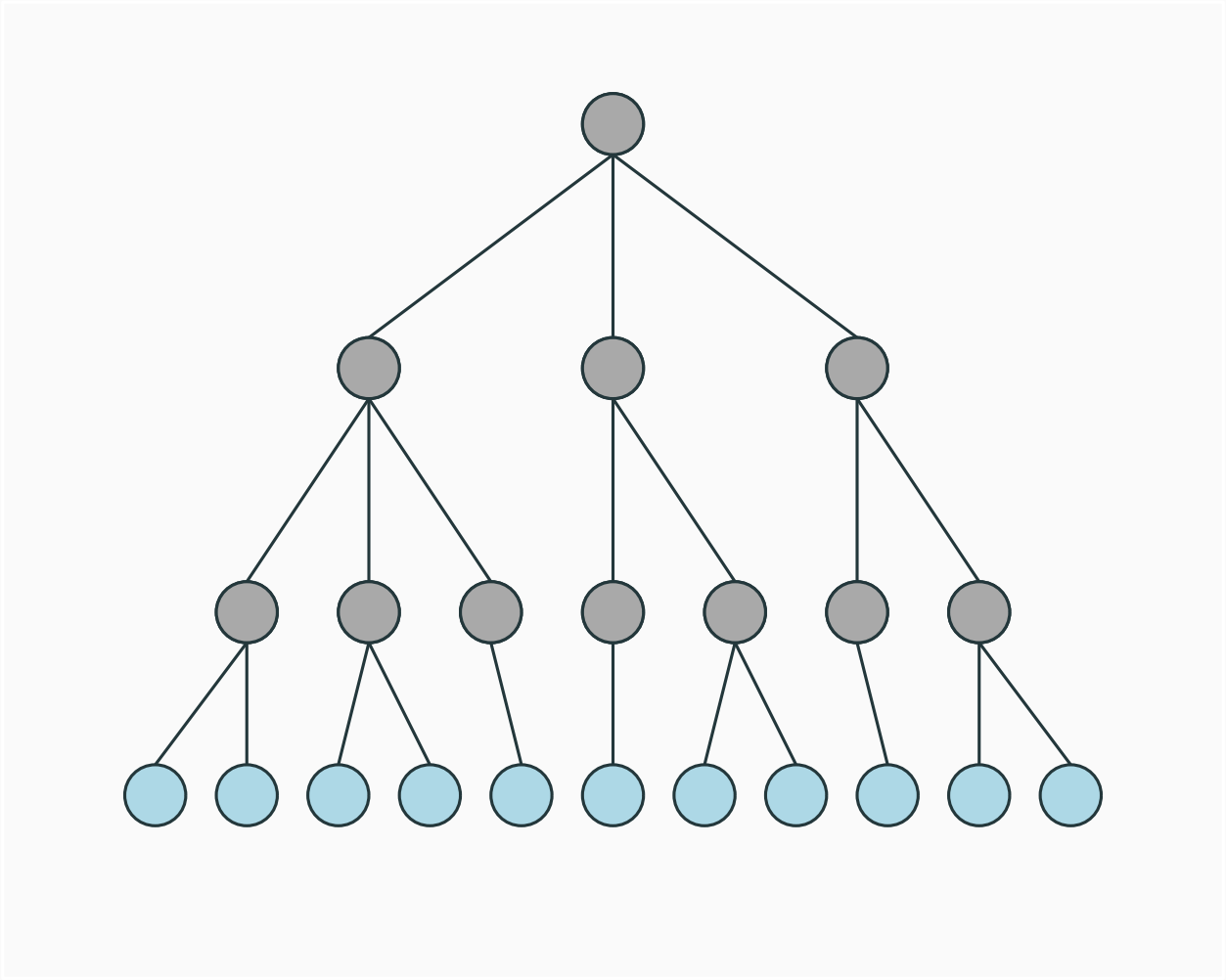
Breadth First Search



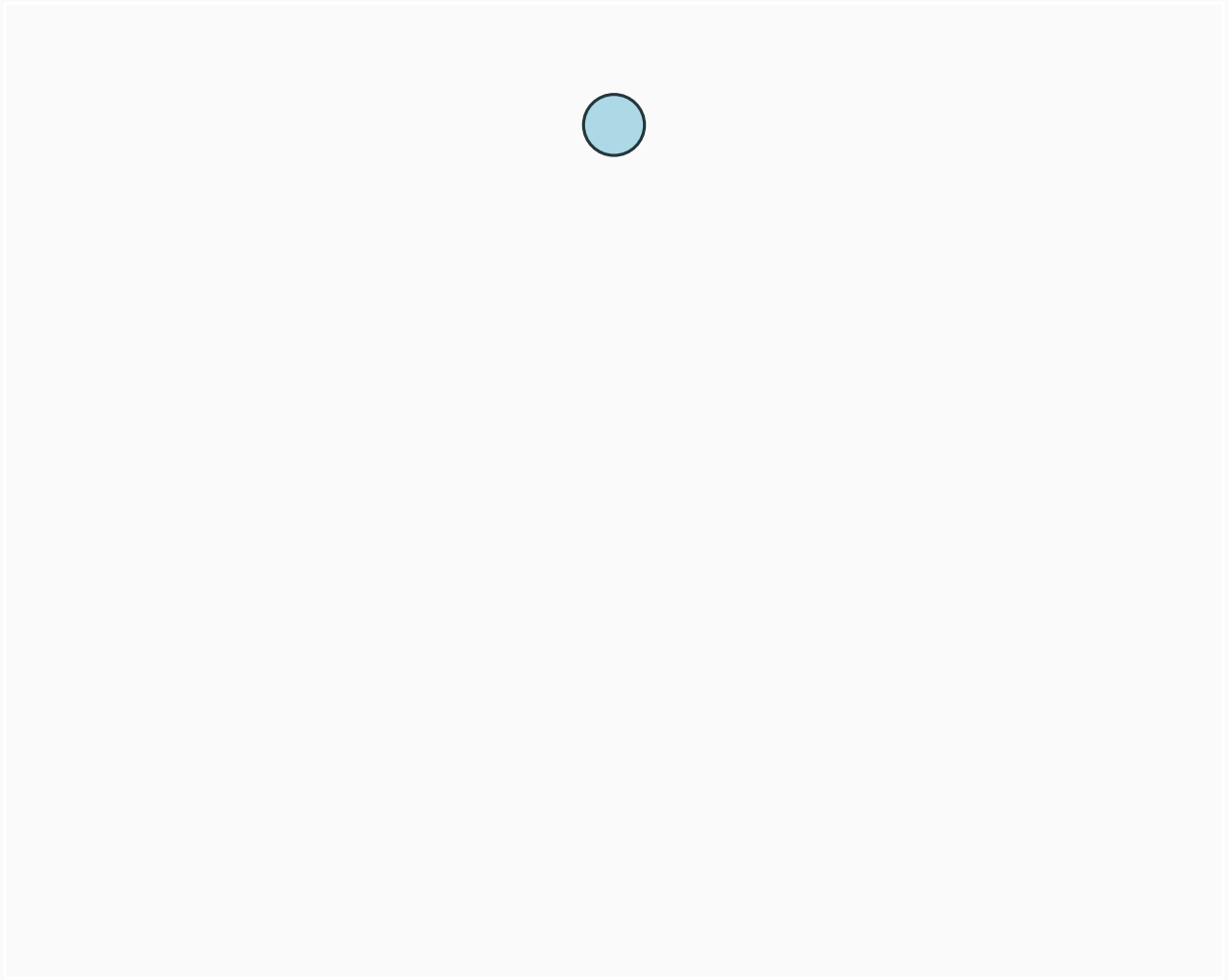
Breadth First Search



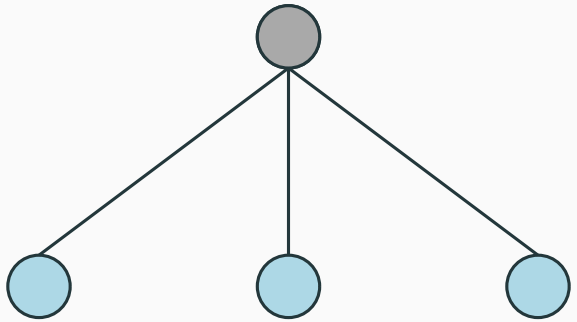
Breadth First Search



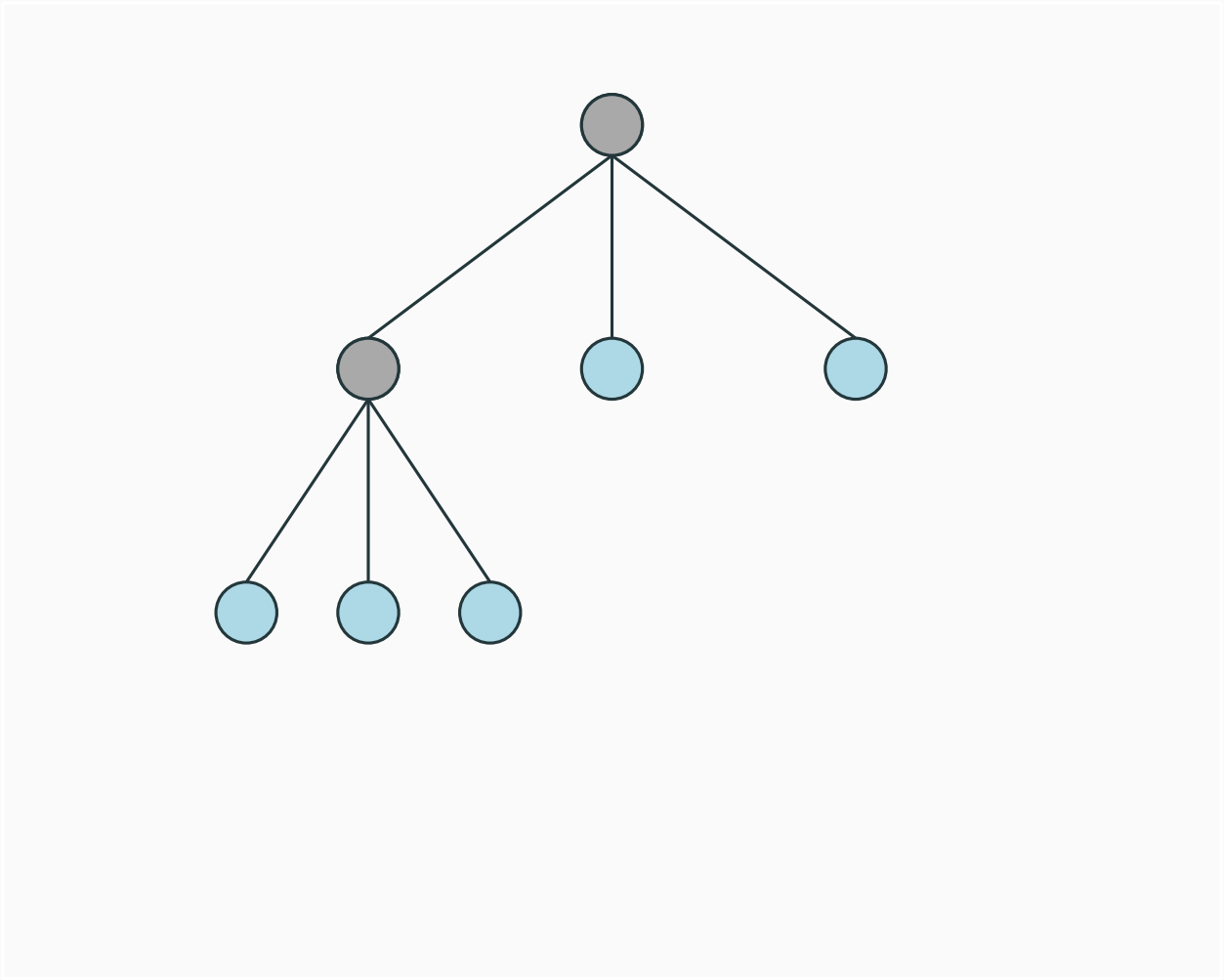
Depth First Search



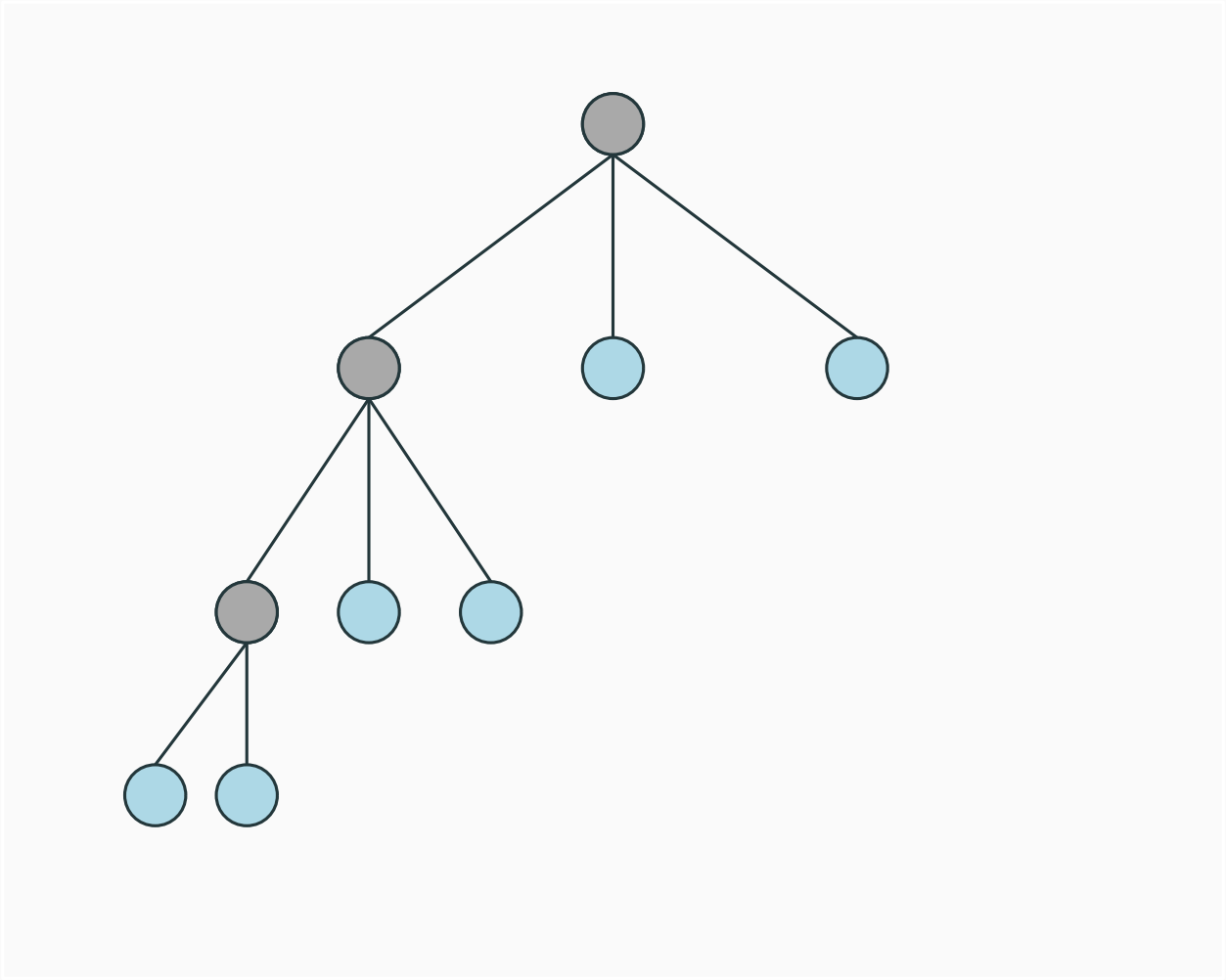
Depth First Search



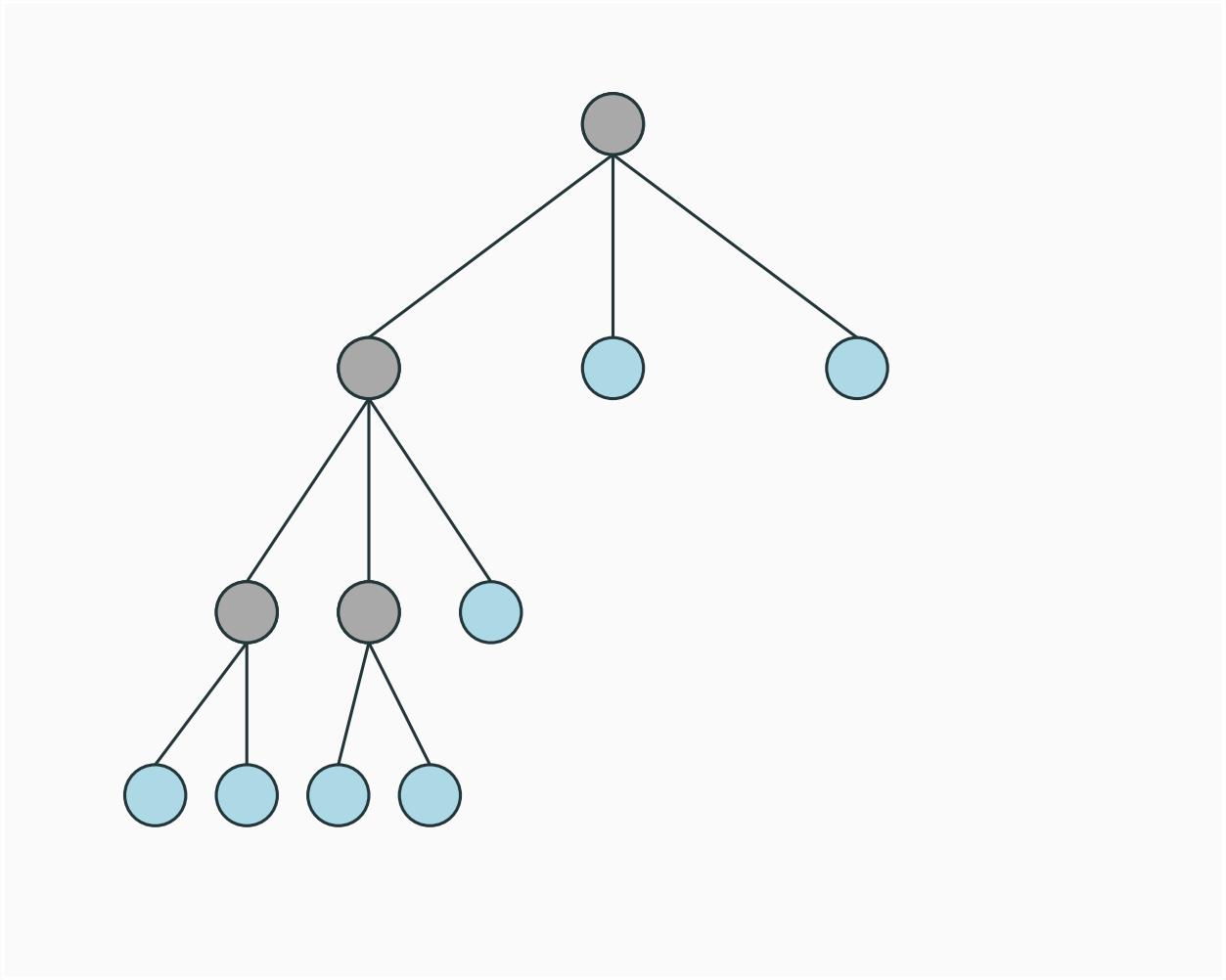
Depth First Search



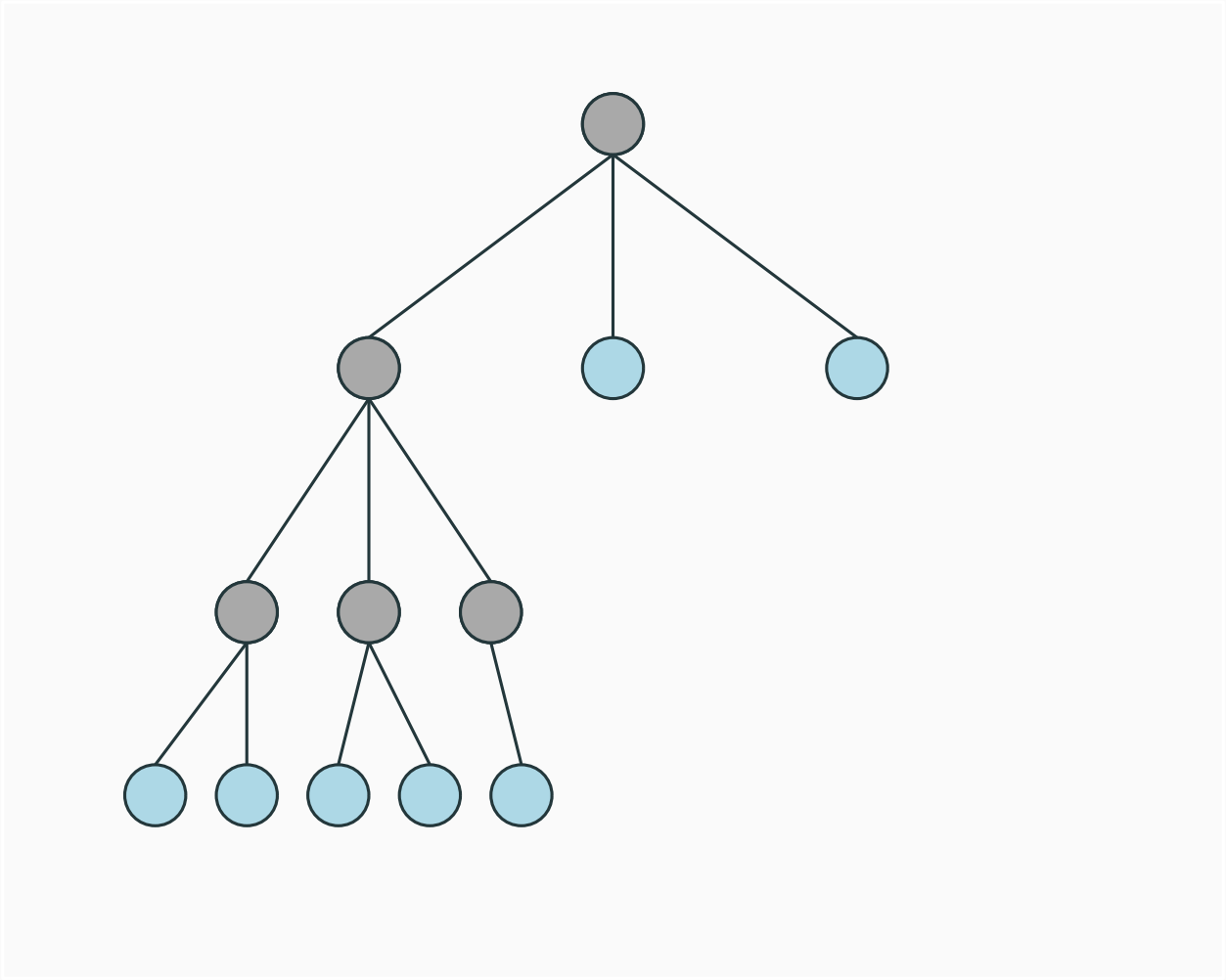
Depth First Search



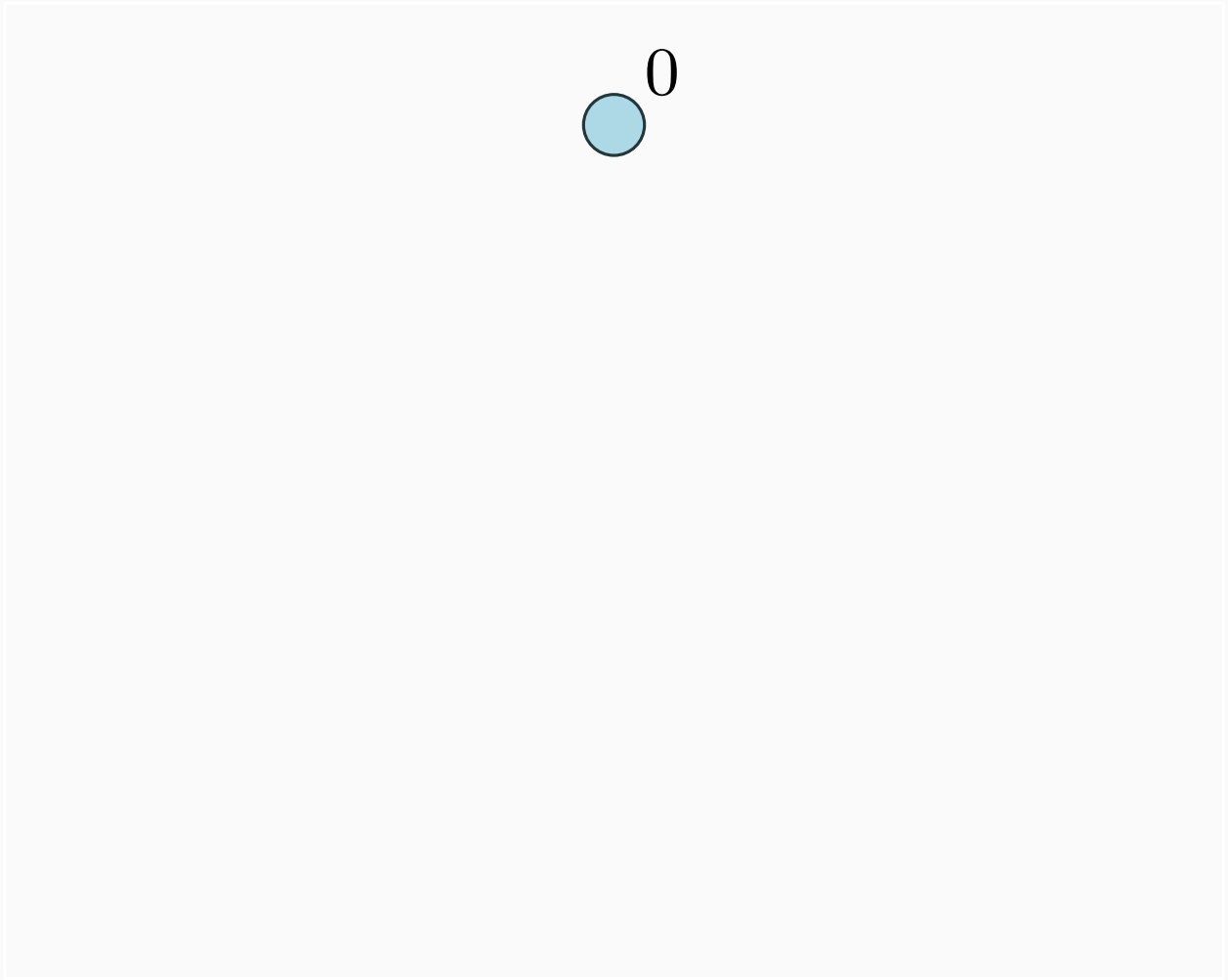
Depth First Search



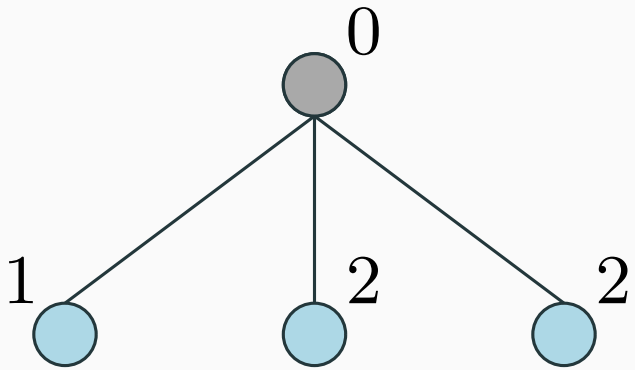
Depth First Search



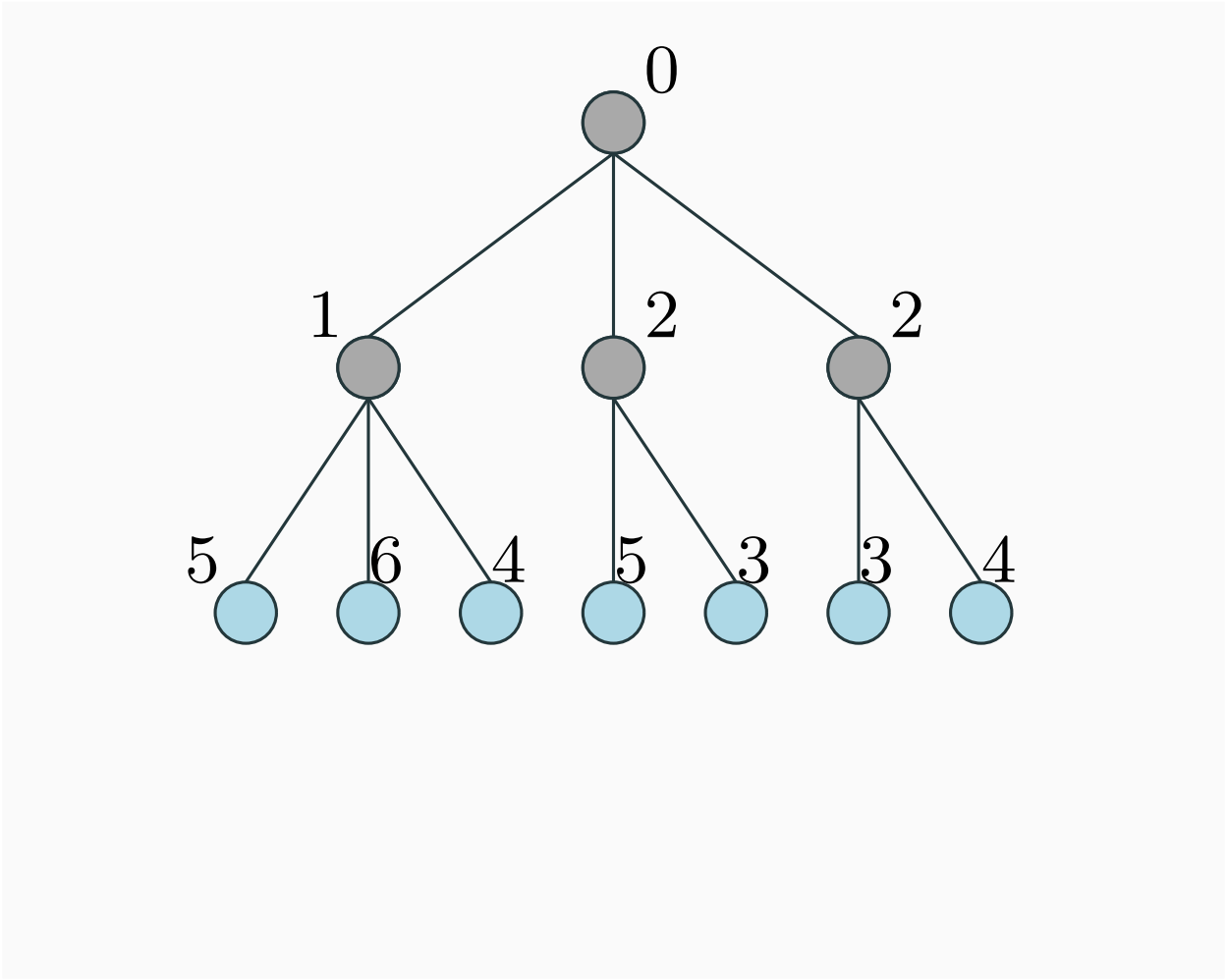
Beam Search ($D = 3$)



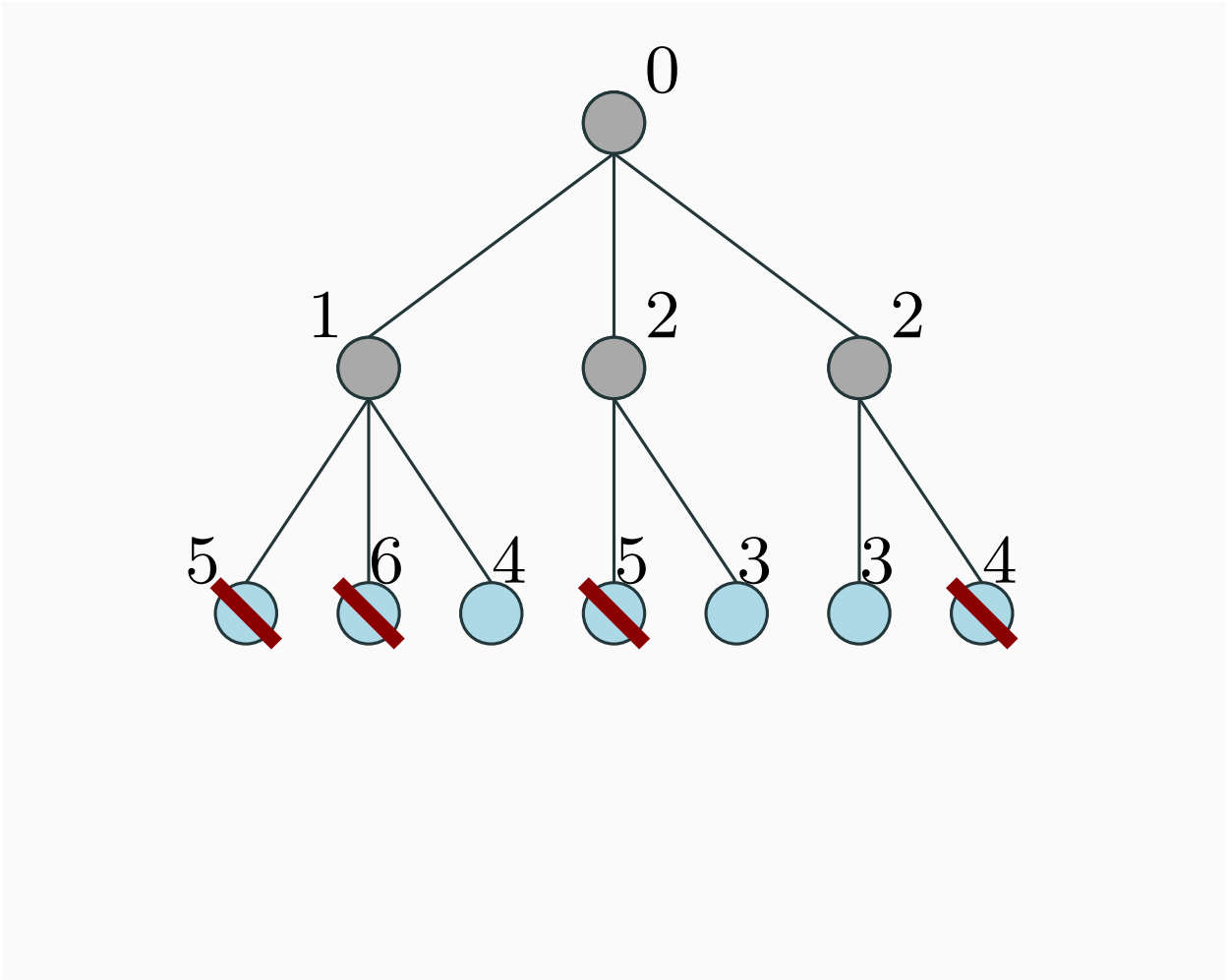
Beam Search ($D = 3$)



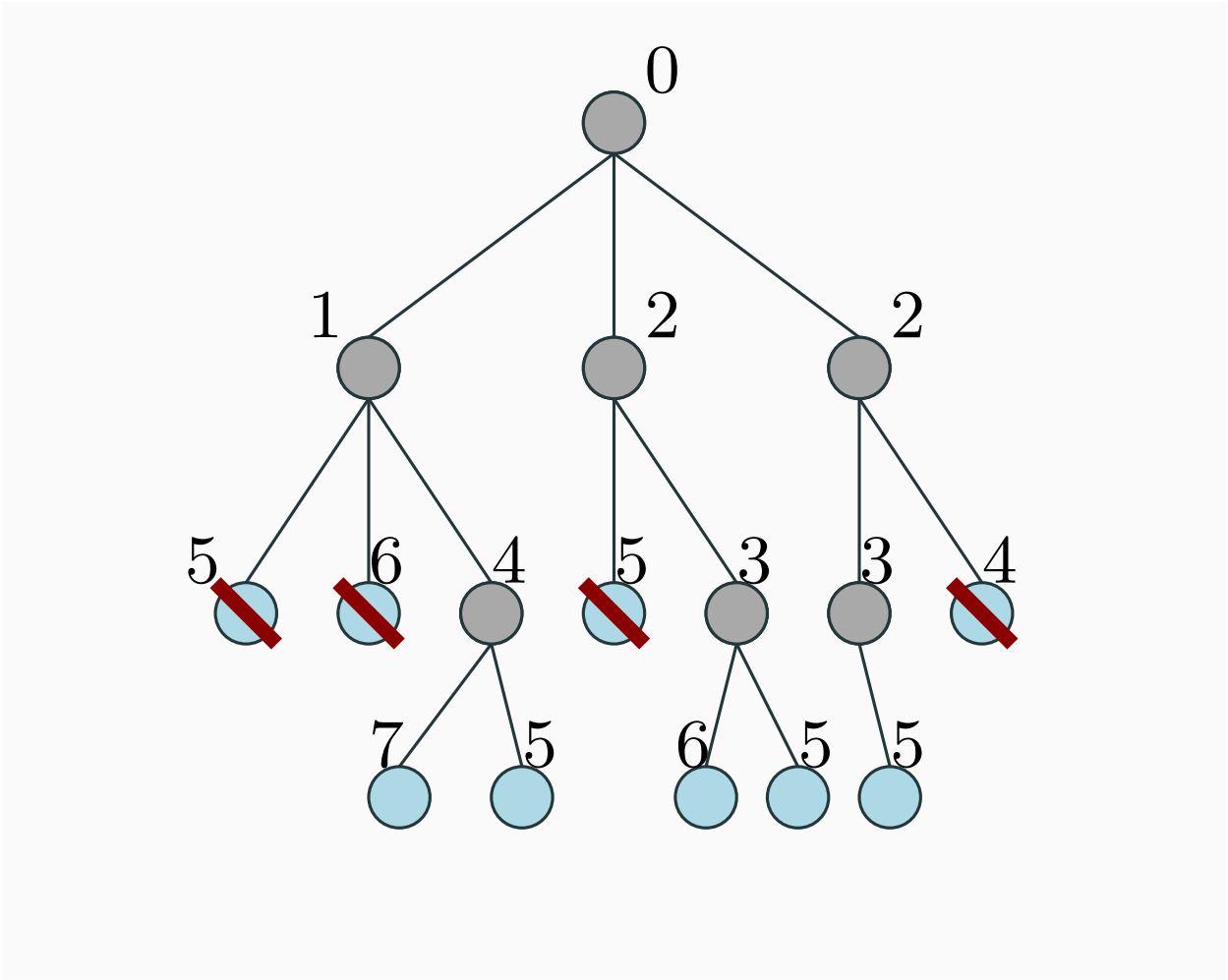
Beam Search ($D = 3$)



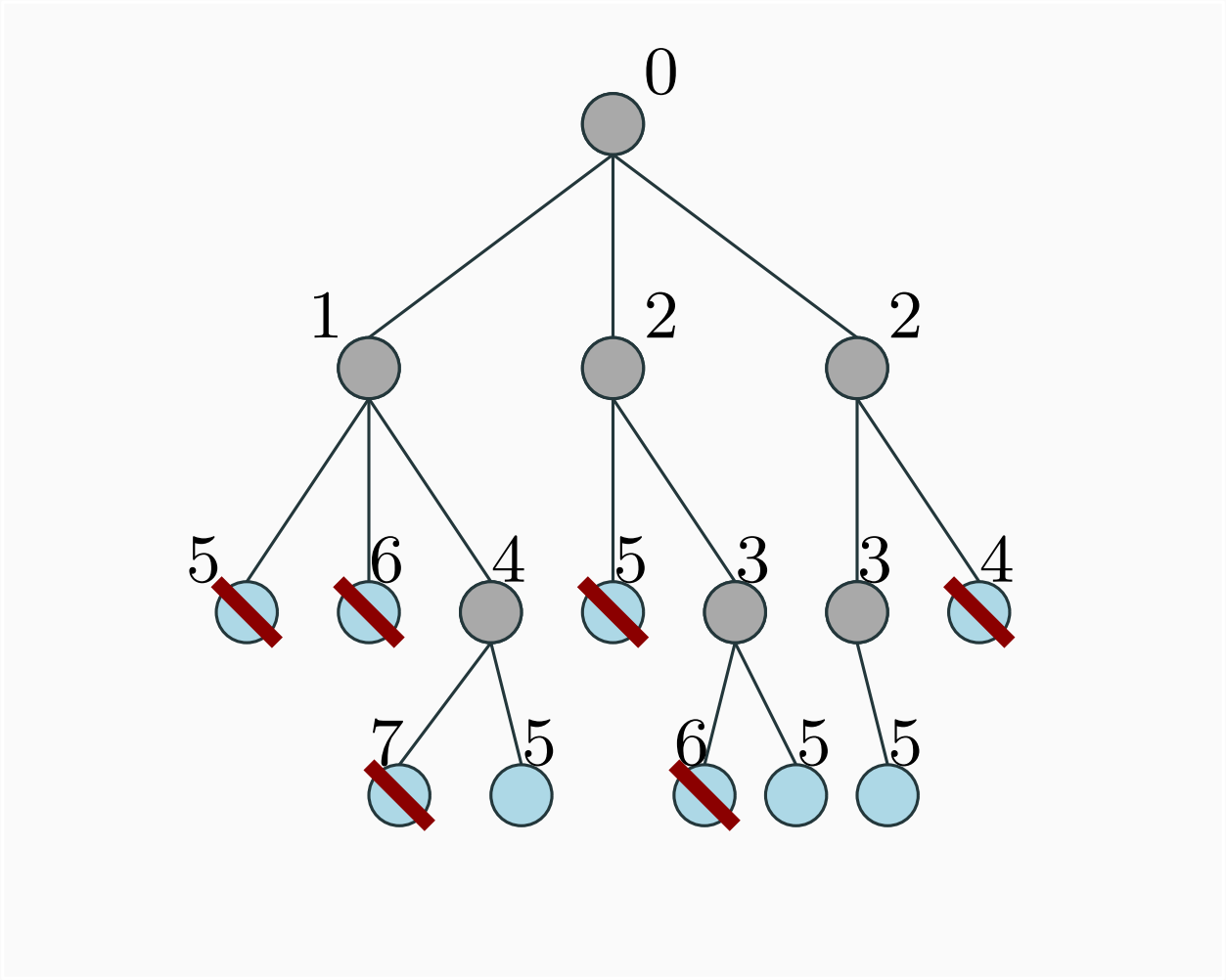
Beam Search ($D = 3$)



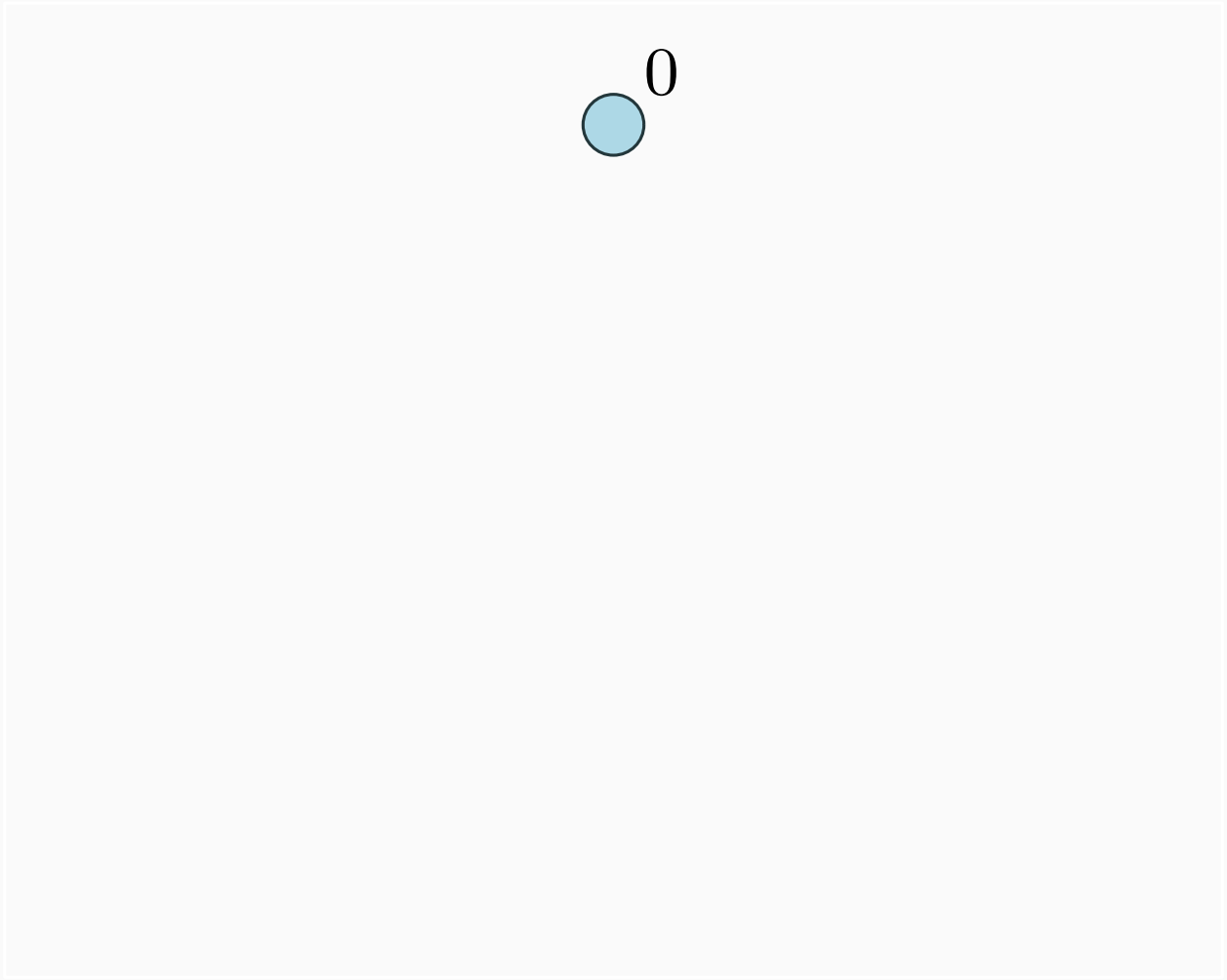
Beam Search ($D = 3$)



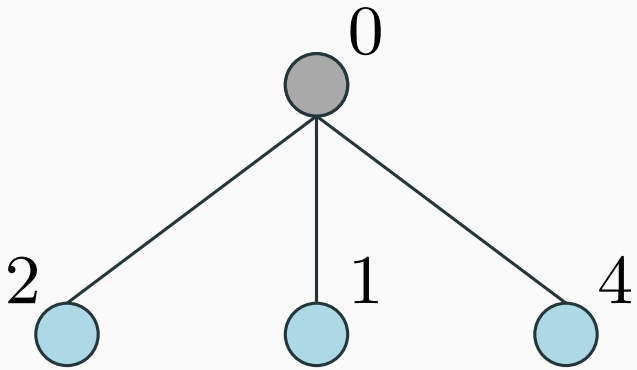
Beam Search ($D = 3$)



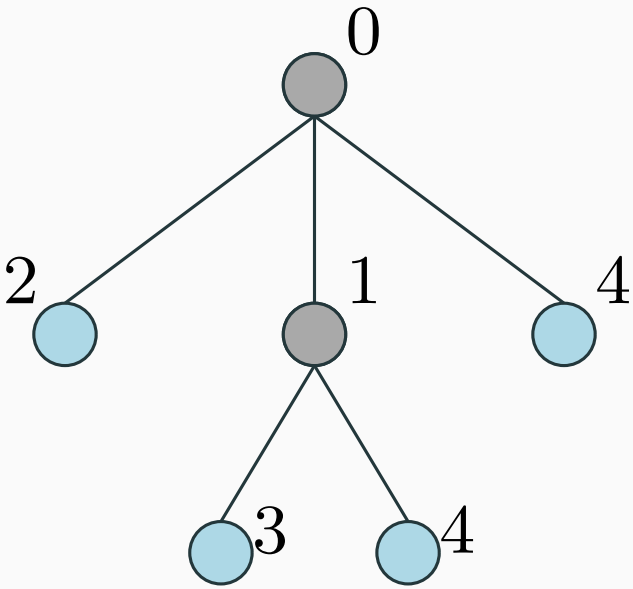
Best First



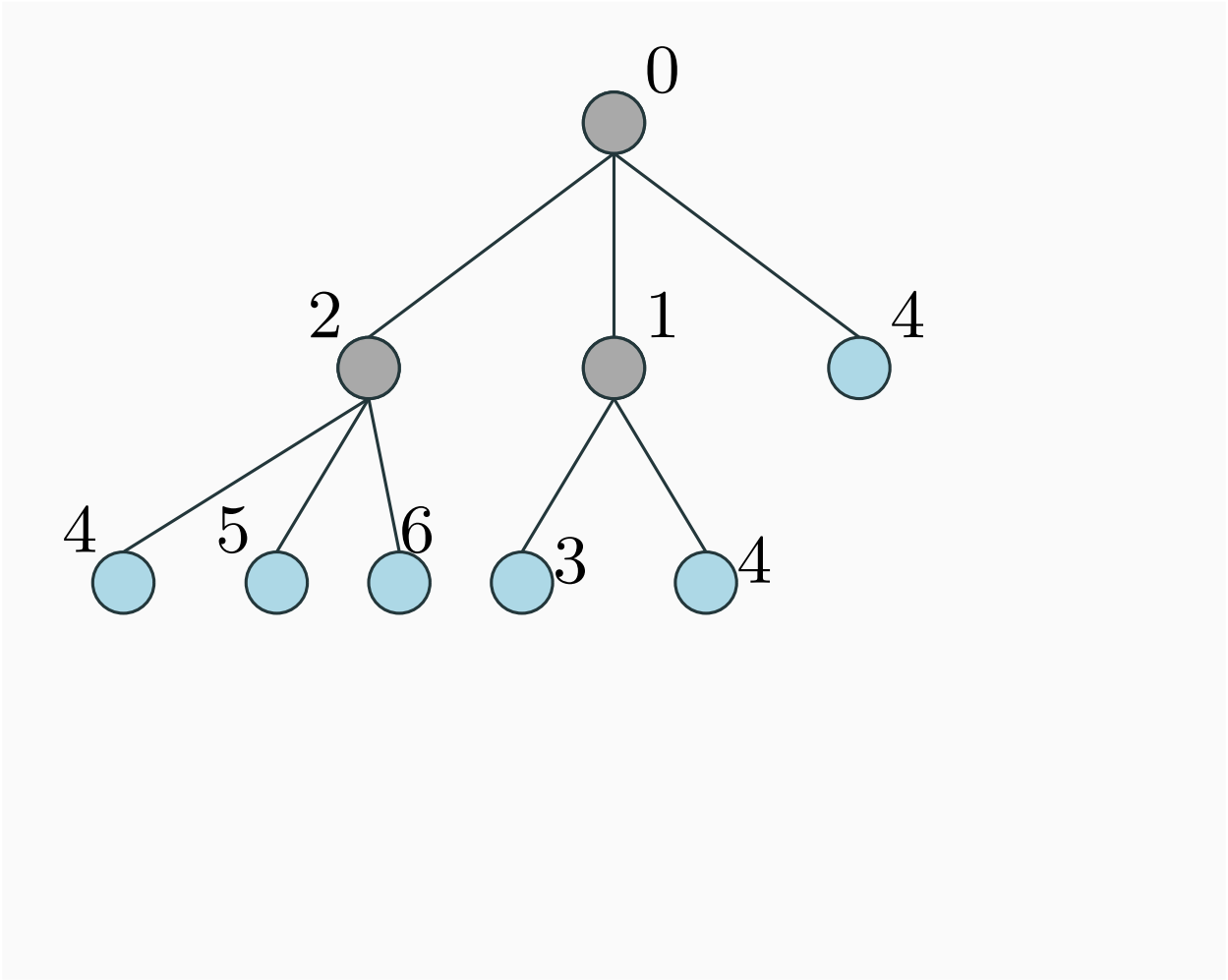
Best First



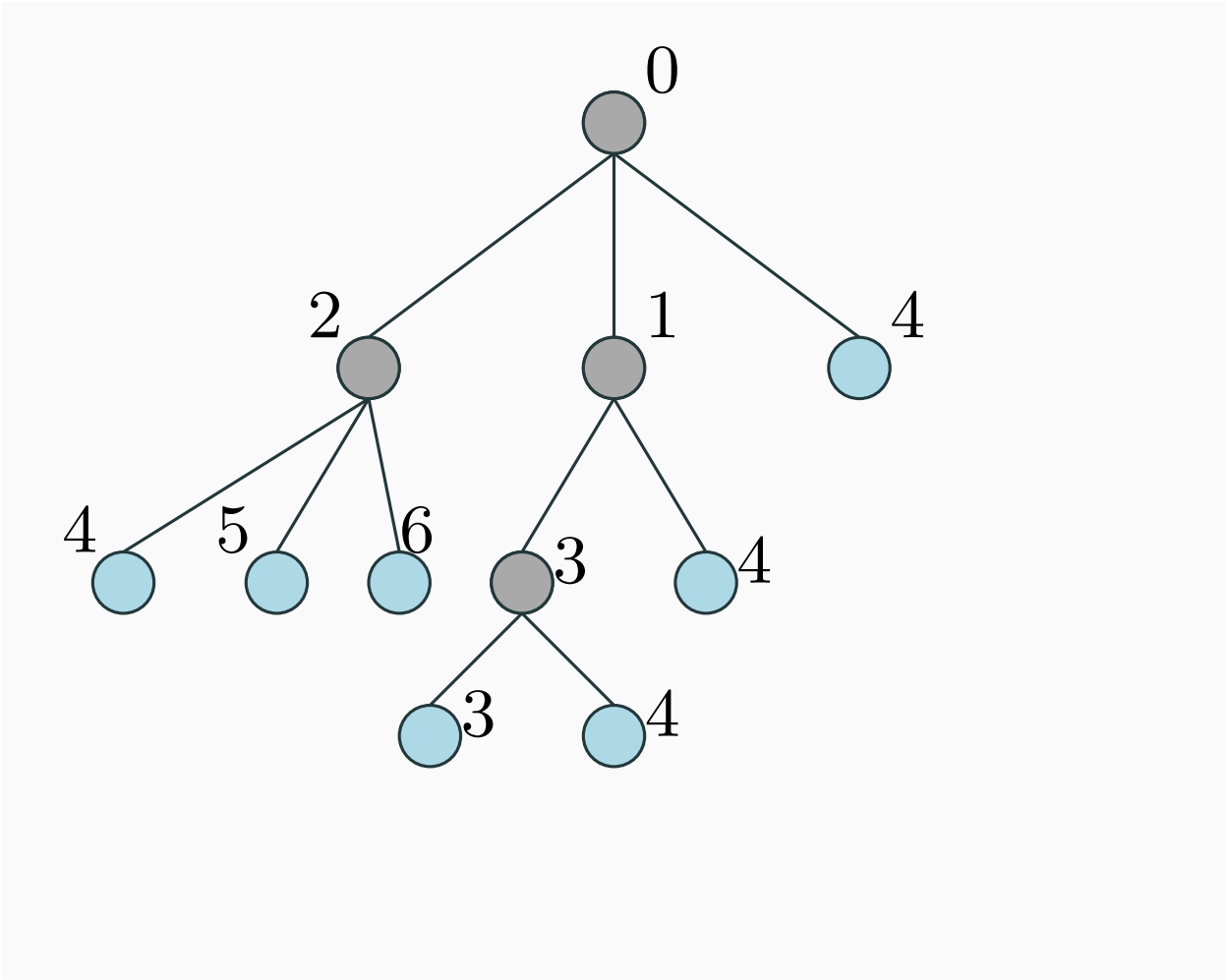
Best First



Best First



Best First



Best First