Selected Topics in Network Optimization: Aligning BDDs for a Facility Location Problem and a Search Tree Method for DSPI.

Dissertation defense

Alexey Bochkarev

2021-11-02 Tue, 08-30, Zoom / Freeman Hall 129

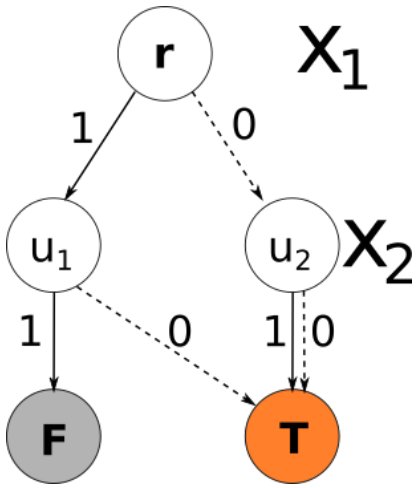
Thanks for joining in the morning!

Presentation outline

What is an (O)BDD?

In a nutshell:

- A (maybe weighted) layered (D)AG
- Two outgoing arcs from each node
- One root, two terminals



Applications:

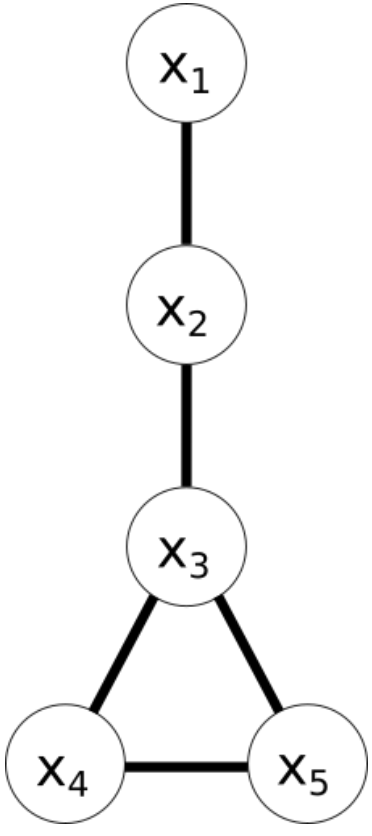
- Can encode a Boolean function ...
- ... or a combinatorial opt problem.

How it works:

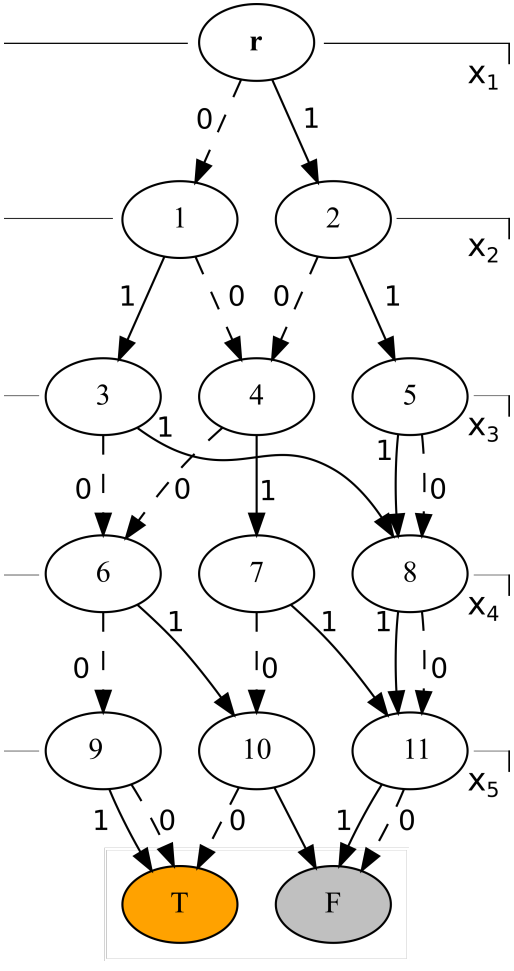
- A layer corresponds to a "decision" (binary variable),
- one-arc (or yes-arc) – $x_i = 1$,
- zero-arc (or no-arc) – $x_i = 0$.
- A path from root to terminal corresponds to an assignment of all variables.

Order of variables matters: a MIS example

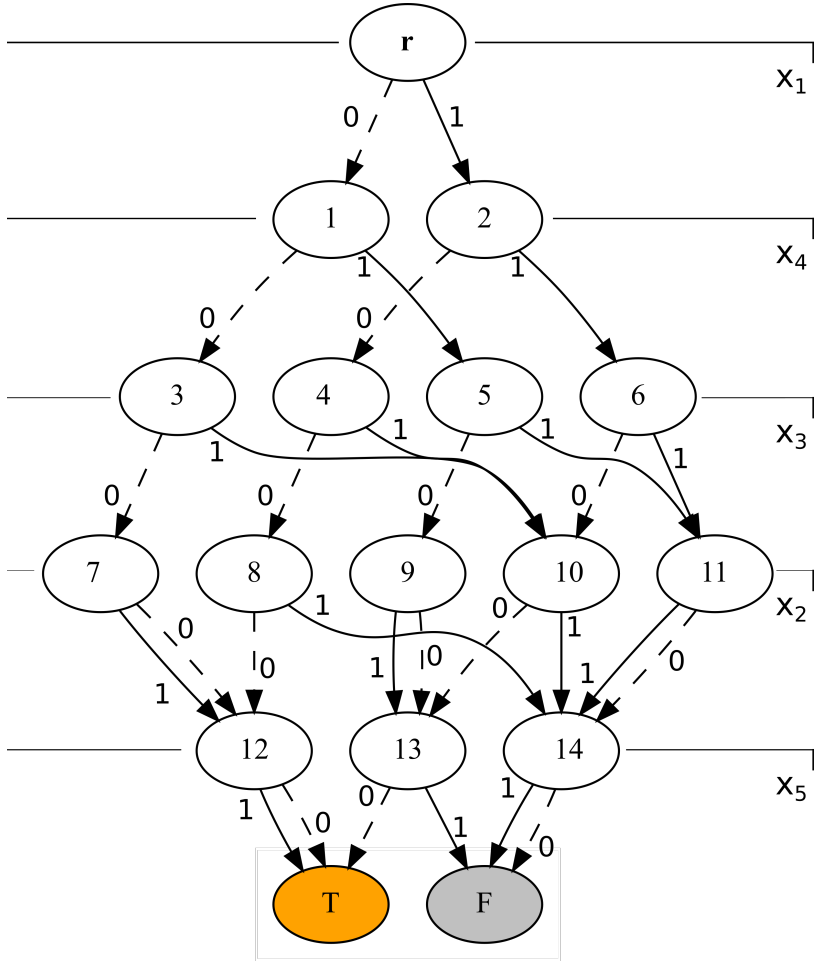
Original graph:



14 nodes:



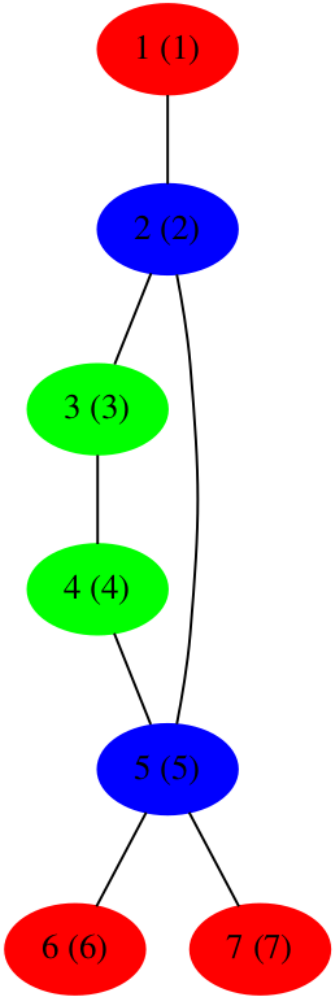
17 nodes:



The same order of variables also matters

- Many problems can be reformulated as a set of (connected) instances over a collection of BDDs.
- If we have the same order of variables:
  [Lozano et al.(2020)Lozano, Bergman, and Smith] proposed an algorithm how to tackle such instances.
- So, finding a good shared order of variables might be desirable.

A motivating example: t-UFLP



Assume:

- A graph; can locate a facility at any point,
- A located facility covers all neighboring points.

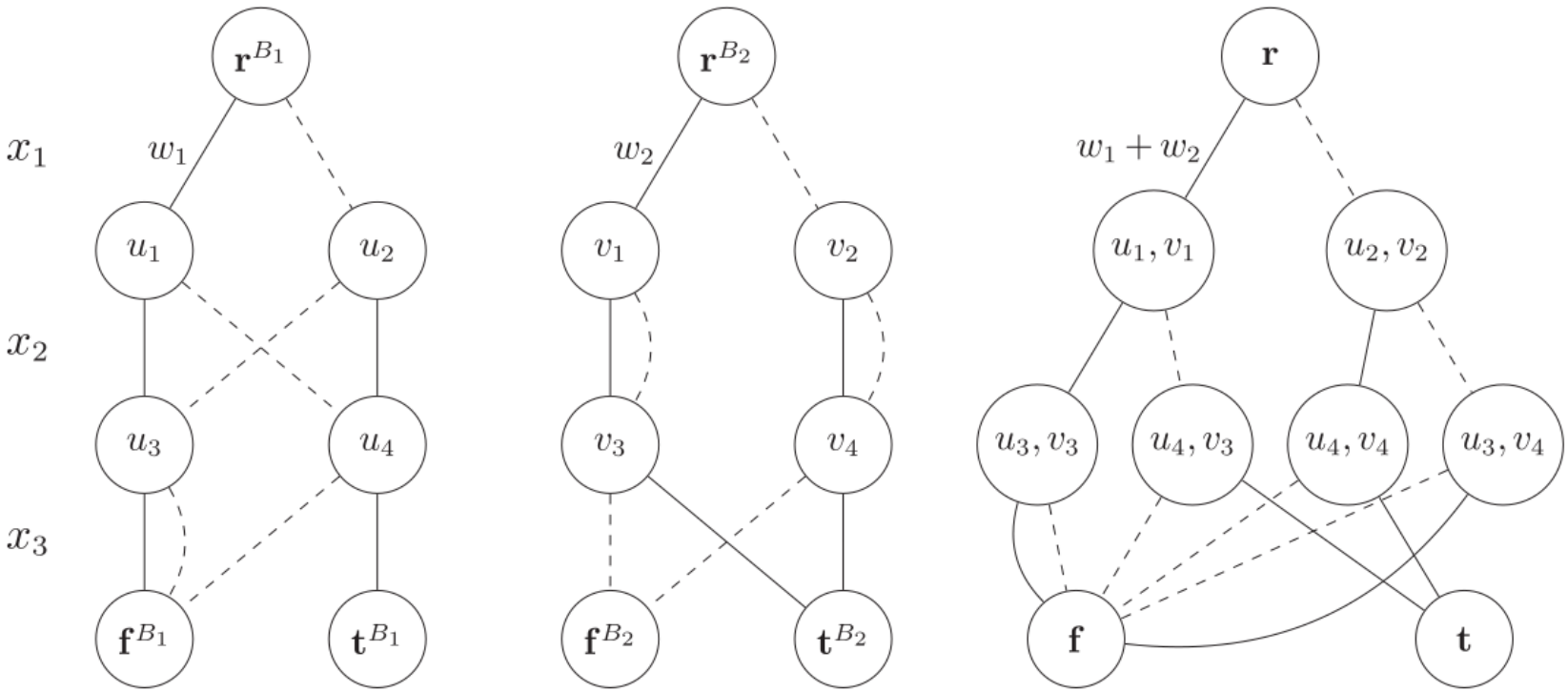
Objective: cover all points at min cost.

Constraints:

- Each point implies a facility "type",
- There is a limit on number of facilities by type.

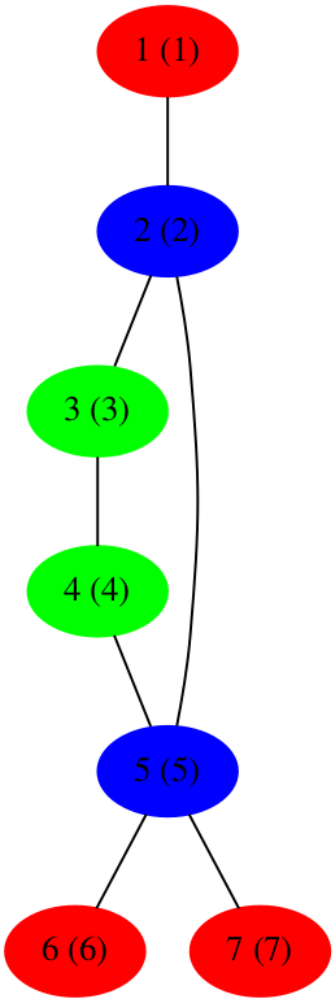
How to merge two diagrams?

Figure 4. Building an Intersection BDD



Source: [Lozano et al.(2020)Lozano, Bergman, and Smith]

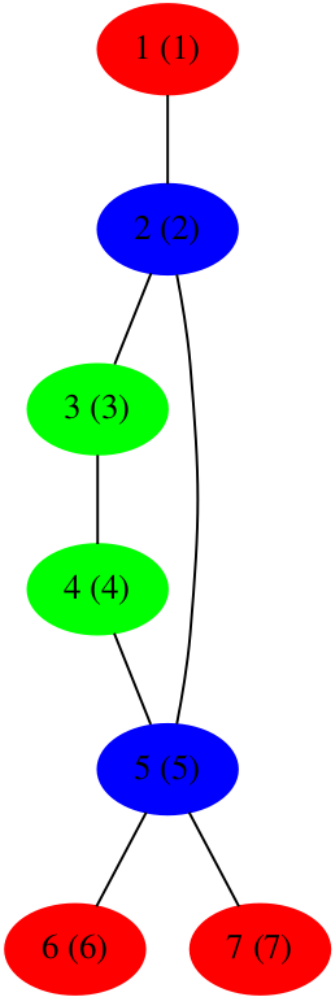
A motivating example: t-UFLP



Plan of attack:

- Formulate the problem with two BDDs,
- Enforce the shared order of variables,
- Merge ("intersect") the two diagrams into one*
- Solve a shortest-path in the intersection DD.

A motivating example: t-UFLP



Plan of attack:

- Formulate the problem with two BDDs,
- Enforce the shared order of variables, ← this part
- Merge ("intersect") the two diagrams into one*
- Solve a shortest-path in the intersection DD.

In this section:

- BDD alignment problem,
- A simplified model / heuristic,
- Numerical experiments on aligning BDDs.

BDD alignment problem

Let us denote $T^*[D, \mathbf{v}]$ – a "transformed" version of BDD D to variable order $\mathbf{v} = (v_1, \ldots, v_N)$. Then:

BDD Alignment Problem / "original" problem

$$s^* = \min_{\mathbf{v}} \Big\{ |T^*[A, \mathbf{v}]| + |T^*[B, \mathbf{v}]| \Big\},$$

where $|D|$ is the number of nodes in diagram D. The problem is NP-hard (because the min single BDD is NP-hard).

Research context

- Vast literature on minimizing single BDD.
- One of the central ideas: "Dynamic variable reordering" / Sifting algorithm (we will use it as a baseline).
- Limited consideration of the multi-BDD version, and all working with BDDs (obviously).

The purpose of this work

Try to avoid manipulations with BDD as much as possible, by introducing a "simpler" auxiliary problem.

How the BDDs are transformed?

The simplified problem

Plan of attack with the simplified problem

How to solve it now? Key properties

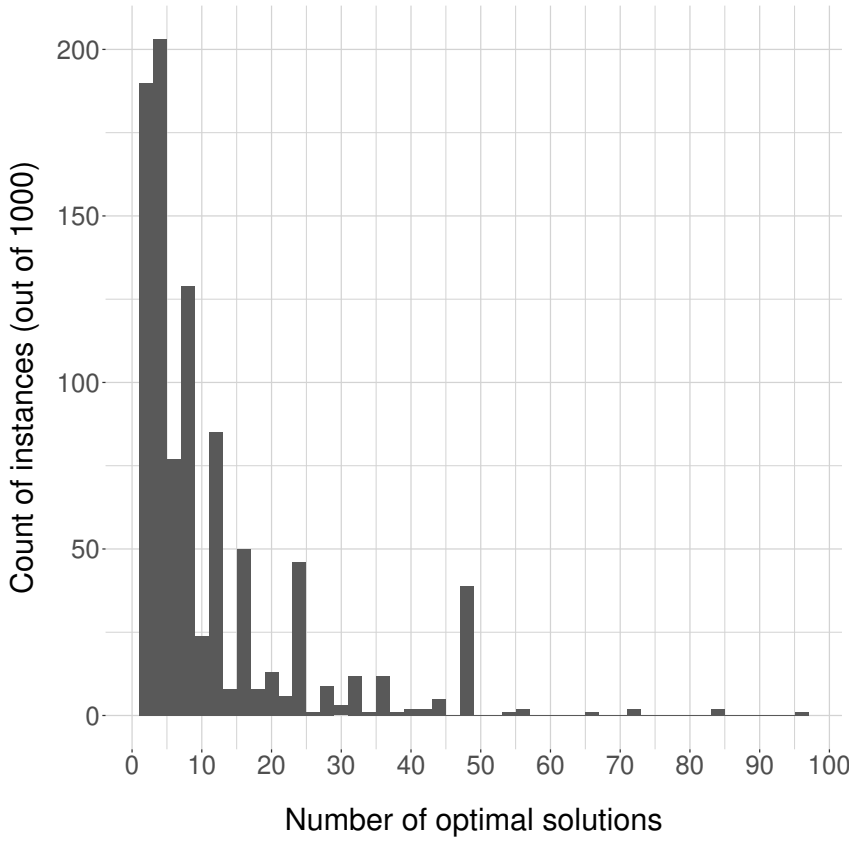
Branch. . .

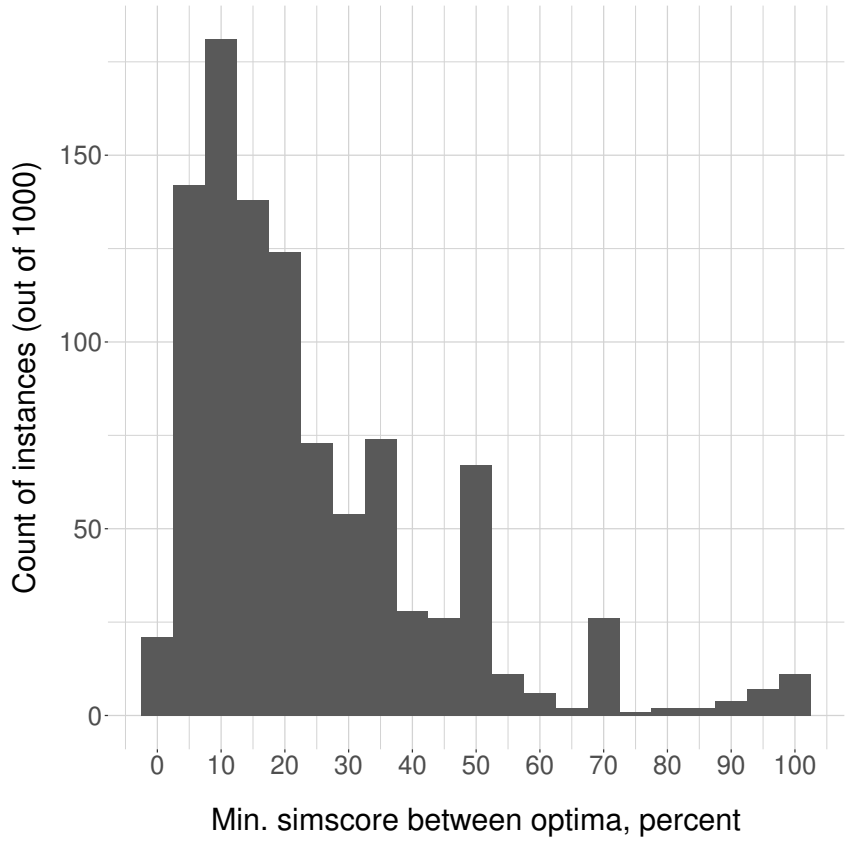
. . . and bound.

Numerical experiments: strategy

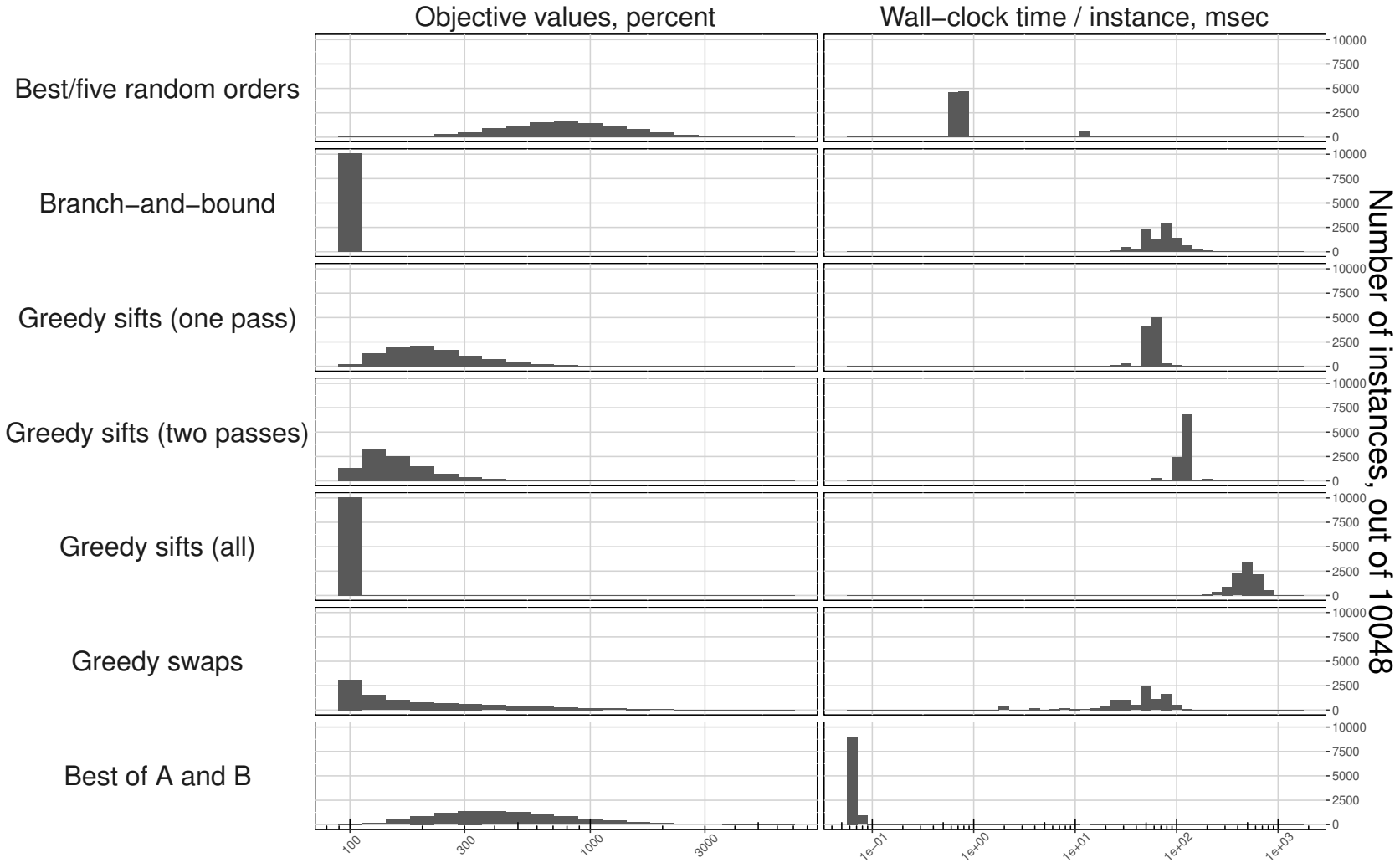
- First: consider the simplified problem in detail.
- Then: solve the original problem, benchmark vs the baseline and consider scaling.
- Analyze the solutions: original vs. simplified, problem structure.

Simplified problem: optima

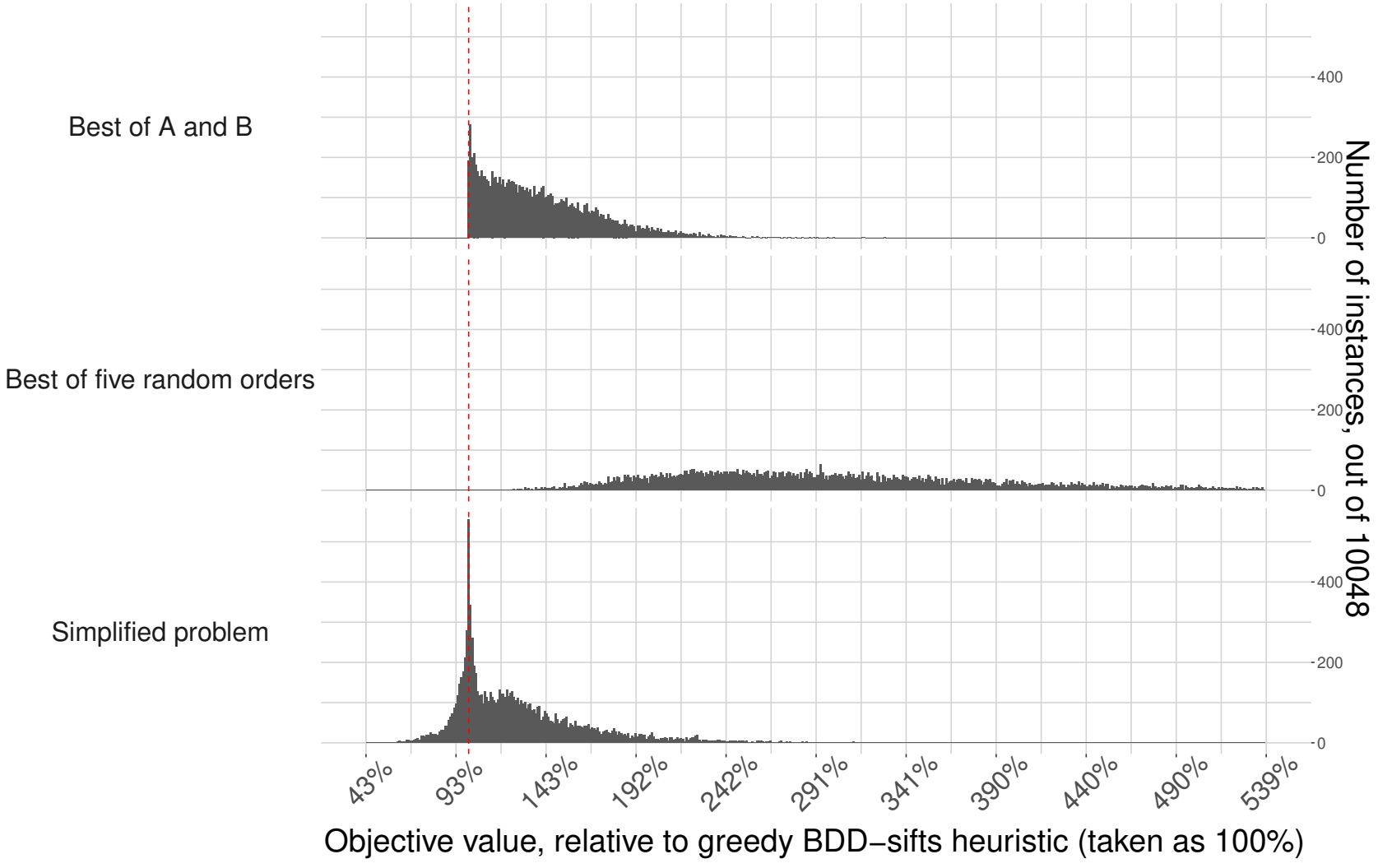




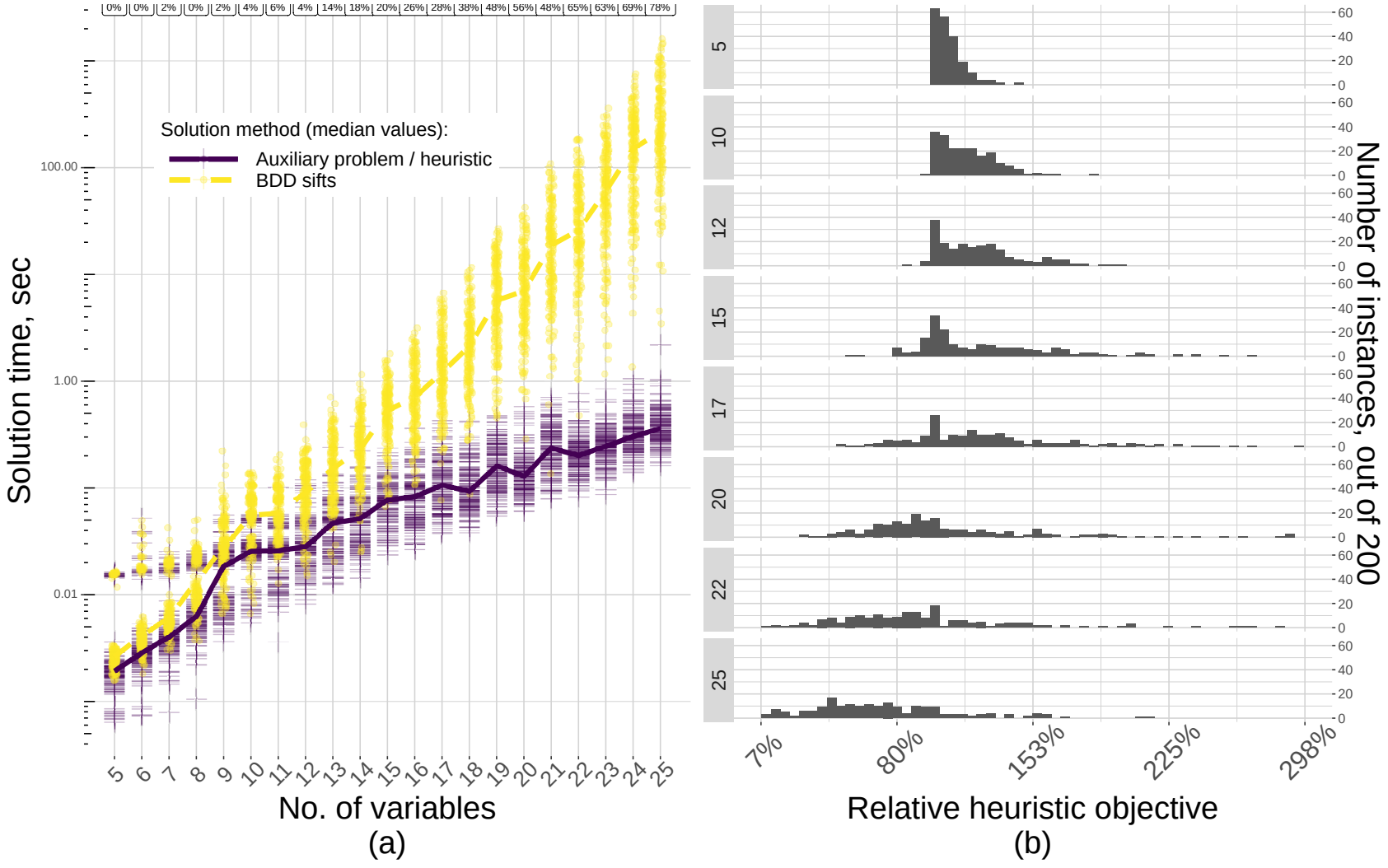
Simplified problem: heuristics



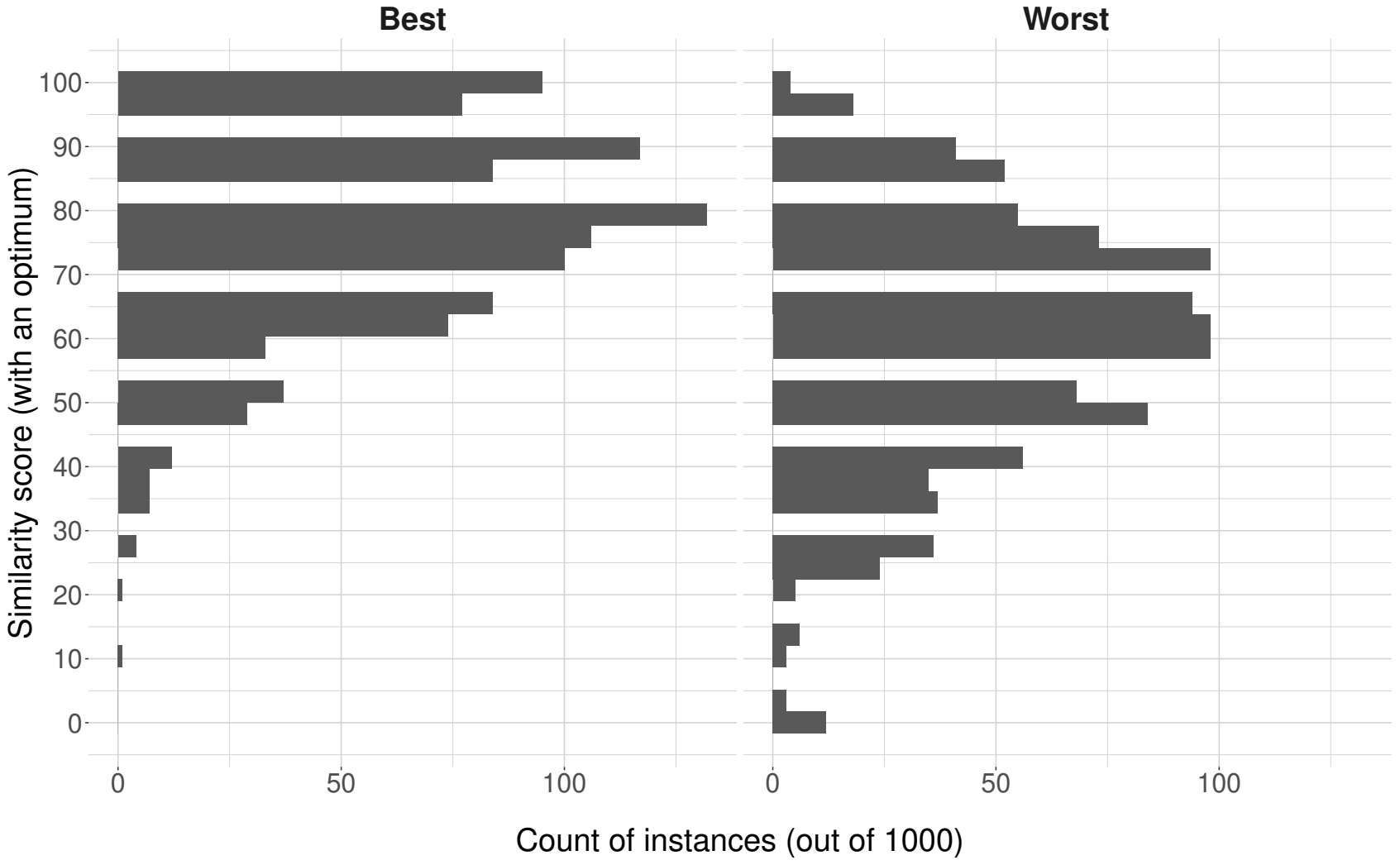
Objectives relative to the "greedy sifts"



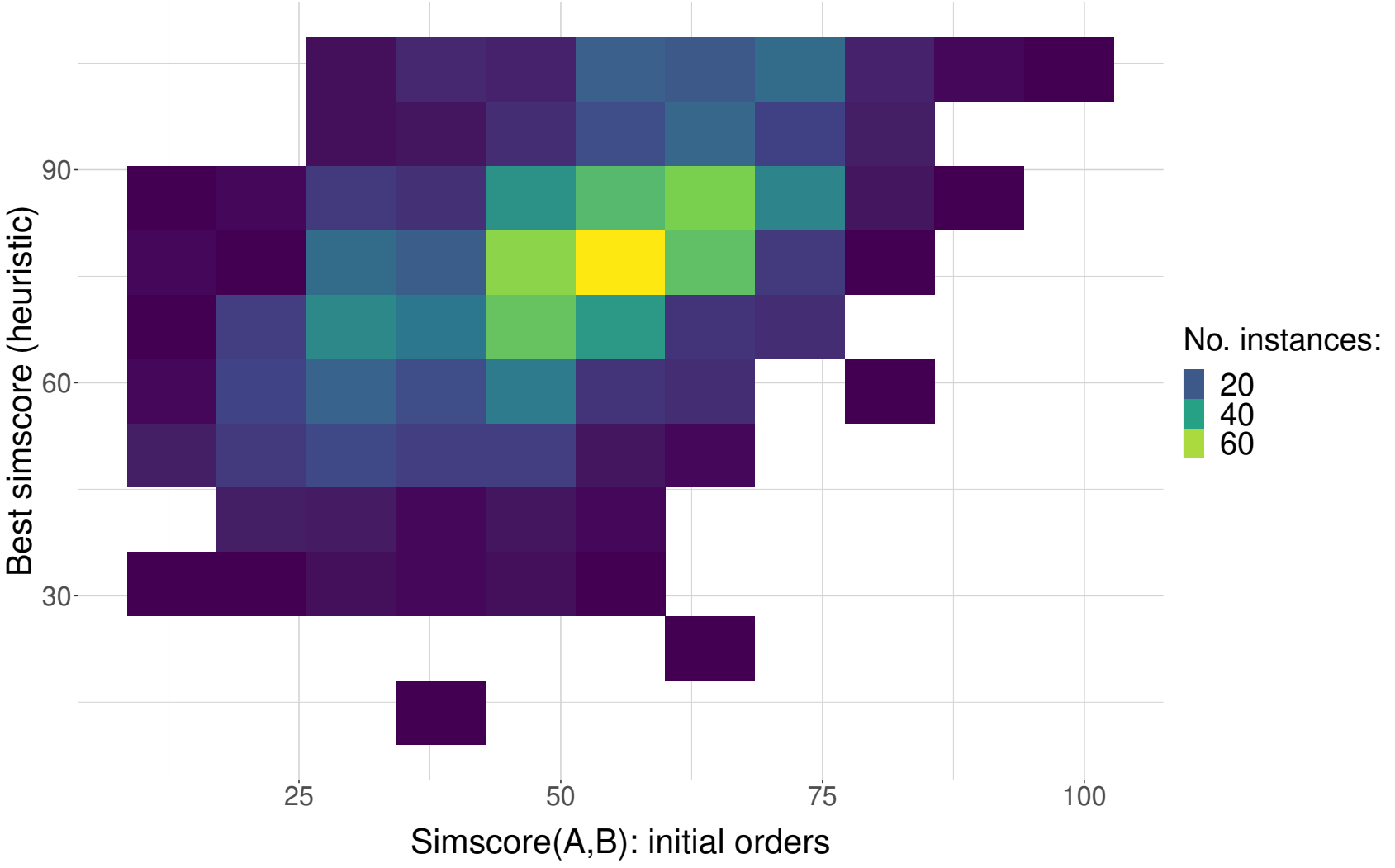
Scaling with the problem size



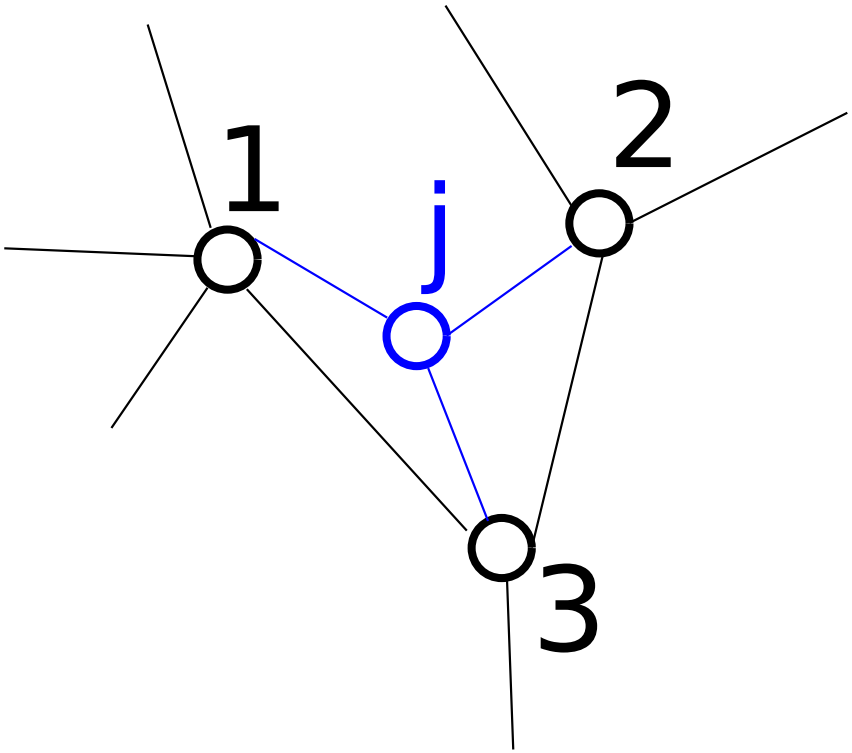
Simplified vs. Original problem solutions



Effects of the problem structure



A variant of the Facility Location Problem



- Locate facilities in a cheapest way, ...
- ... to cover all points, ...
- ... respecting the budget constraints, by facility "type".

Problem formulation

$$\min \sum_{i=1}^{M} f_i x_i$$

$$\text{s.t.} \sum_{j \in S_i} x_j \geq 1 \qquad i = 1, \ldots, M,$$

$$\sum_{j \in T_t} x_j \leq k_t \qquad t = 1, \ldots, K,$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, M.$$

Here:

- S_j – adjacency lists,
- k_t – budgets,
- f_i – location costs,
- x_i – location decisions.

Problem formulation

$$\min \sum_{i=1}^{M} f_i x_i$$

$$\text{s.t.} \sum_{j \in S_i} x_j \geq 1 \qquad i = 1, \ldots, M, \rightarrow \text{“Cover” BDD}$$

$$\sum_{j \in T_t} x_j \leq k_t \qquad t = 1, \ldots, K, \rightarrow \text{“Type” BDD}$$

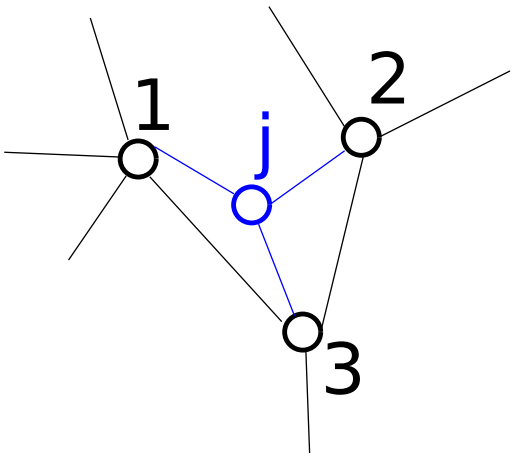
$$x_i \in \{0, 1\} \qquad i = 1, \ldots, M.$$

Here:

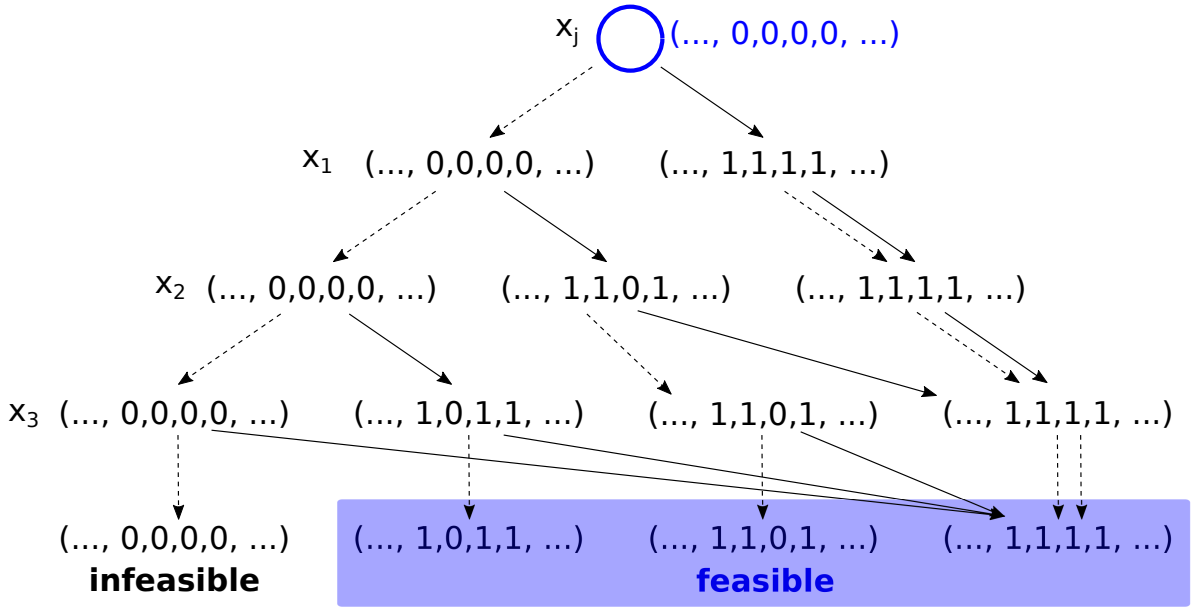
- S_j – adjacency lists,
- k_t – budgets,
- f_i – location costs,
- x_i – location decisions.

How to build the cover BDD

Original graph:



Cover BDD:

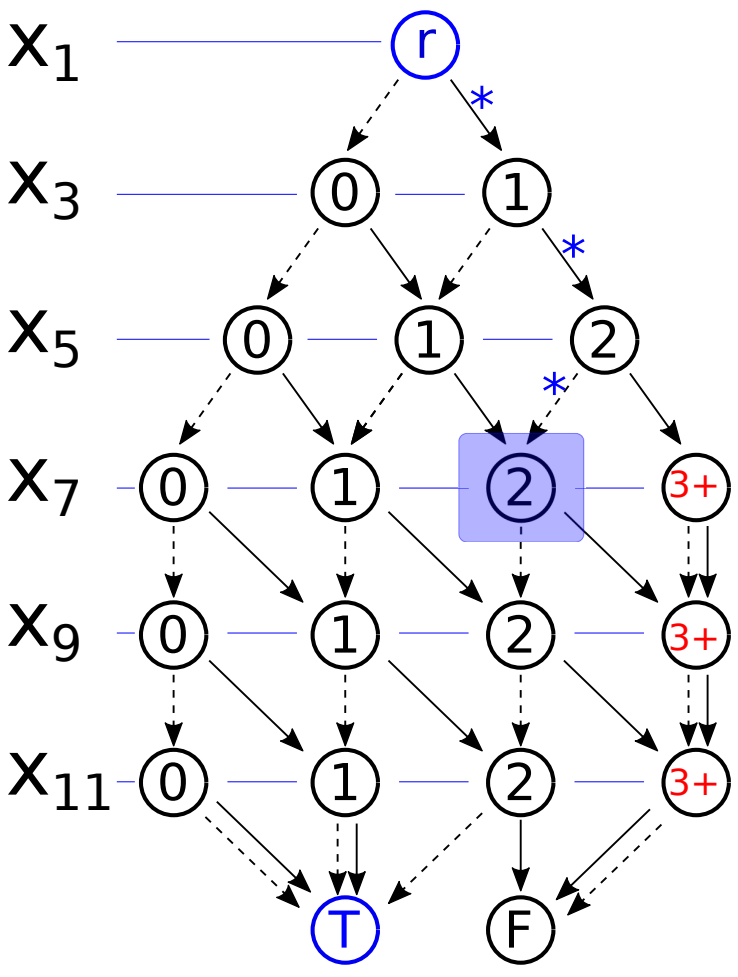


Building the diagram:

- Pick next point with the least "uncertainty": # of neighbors to be added to the BDD,
- Process its adjacency list,
- Repeat.

How to build the type BDD

Part of Type BDD:



- Consider: $\sum_{j \in T} x_j \leq k$ for $T = \{1, 3, 5, 7, 9, 11\}$ and $k = 2$.
- Stack such blocks vertically.
- Minimize the number of inversions with the Cover BDD by choosing the order of variables within each block.
- Randomize the order of blocks.

Consistent Path Problem for t-UFLP

$$\min \sum f_i v^{\mathtt{C}}_{i,\mathtt{HI(i)}},$$

$$\text{s.t.} \sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{C}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{C}}_{uj} \text{ for all } u \in L^{\mathtt{C}}_2 \cup \ldots \cup L^{\mathtt{C}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{C}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{C}}_{j\mathtt{u}} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{C}}, \mathbf{F}^{\mathtt{C}}\},$$

$$\sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{T}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{T}}_{uj} \text{ for all } u \in L^{\mathtt{T}}_2 \cup \ldots \cup L^{\mathtt{T}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{T}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{T}}_{j\mathtt{u}} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{T}}, \mathbf{F}^{\mathtt{T}}\},$$

$$\sum_{j \in L^{\mathtt{C}}_q} v^{\mathtt{C}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$\sum_{j \in L^{\mathtt{T}}_q} v^{\mathtt{T}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$v_{pq} \geq 0 \text{ for all valid } p,\ q; \quad x_i \in \{0, 1\}.$$

Consistent Path Problem for t-UFLP

$$\min \sum f_i v^{\mathtt{C}}_{i,\mathtt{HI(i)}},$$

$$\text{s.t.} \sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{C}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{C}}_{uj} \text{ for all } u \in L^{\mathtt{C}}_2 \cup \ldots \cup L^{\mathtt{C}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{C}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{C}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{C}}, \mathbf{F}^{\mathtt{C}}\},$$

$$\sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{T}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{T}}_{uj} \text{ for all } u \in L^{\mathtt{T}}_2 \cup \ldots \cup L^{\mathtt{T}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{T}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{T}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{T}}, \mathbf{F}^{\mathtt{T}}\},$$

$$\sum_{j \in L^{\mathtt{C}}_q} v^{\mathtt{C}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$\sum_{j \in L^{\mathtt{T}}_q} v^{\mathtt{T}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$v_{pq} \geq 0 \text{ for all valid } p,\ q; \quad x_i \in \{0, 1\}.$$

Consistent Path Problem for t-UFLP

$$\min \sum f_i v^{\mathtt{C}}_{i,\mathtt{HI(i)}},$$

$$\text{s.t.} \sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{C}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{C}}_{uj} \text{ for all } u \in L^{\mathtt{C}}_2 \cup \ldots \cup L^{\mathtt{C}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{C}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{C}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{C}}, \mathbf{F}^{\mathtt{C}}\},$$

$$\sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{T}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{T}}_{uj} \text{ for all } u \in L^{\mathtt{T}}_2 \cup \ldots \cup L^{\mathtt{T}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{T}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{T}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{T}}, \mathbf{F}^{\mathtt{T}}\},$$

$$\sum_{j \in L^{\mathtt{C}}_q} v^{\mathtt{C}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$\sum_{j \in L^{\mathtt{T}}_q} v^{\mathtt{T}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$v_{pq} \geq 0 \text{ for all valid } p,\ q; \quad x_i \in \{0, 1\}.$$

Consistent Path Problem for t-UFLP

$$\min \sum f_i v^{\mathtt{C}}_{i,\mathtt{HI(i)}},$$

$$\text{s.t.} \sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{C}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{C}}_{uj} \text{ for all } u \in L^{\mathtt{C}}_2 \cup \ldots \cup L^{\mathtt{C}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{C}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{C}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{C}}, \mathbf{F}^{\mathtt{C}}\},$$

$$\sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{T}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{T}}_{uj} \text{ for all } u \in L^{\mathtt{T}}_2 \cup \ldots \cup L^{\mathtt{T}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{T}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{T}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{T}}, \mathbf{F}^{\mathtt{T}}\},$$

$$\sum_{j \in L^{\mathtt{C}}_q} v^{\mathtt{C}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$\sum_{j \in L^{\mathtt{T}}_q} v^{\mathtt{T}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$v_{pq} \geq 0 \text{ for all valid } p,\ q; \quad x_i \in \{0, 1\}.$$

Consistent Path Problem for t-UFLP

$$\min \sum f_i v^{\mathtt{C}}_{i,\mathtt{HI(i)}},$$

$$\text{s.t.} \sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{C}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{C}}_{uj} \text{ for all } u \in L^{\mathtt{C}}_2 \cup \ldots \cup L^{\mathtt{C}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{C}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{C}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{C}}, \mathbf{F}^{\mathtt{C}}\},$$

$$\sum_{i:\mathtt{HI(i)}=u \text{ or } \mathtt{LO(i)}=u} v^{\mathtt{T}}_{iu} = \sum_{j:\mathtt{HI(u)}=j \text{ or } \mathtt{LO(u)}=j} v^{\mathtt{T}}_{uj} \text{ for all } u \in L^{\mathtt{T}}_2 \cup \ldots \cup L^{\mathtt{T}}_{(M-1)},$$

$$\sum_{j:\mathtt{HI(r)}=j \text{ or } \mathtt{LO(r)}=j} v^{\mathtt{T}}_{\mathtt{r}j} = 1,$$

$$\sum_{j:\mathtt{HI(j)}=u \text{ or } \mathtt{LO(j)}=u} v^{\mathtt{T}}_{ju} = 1 \text{ for } u \in \{\mathbf{T}^{\mathtt{T}}, \mathbf{F}^{\mathtt{T}}\},$$

$$\sum_{j \in L^{\mathtt{C}}_q} v^{\mathtt{C}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

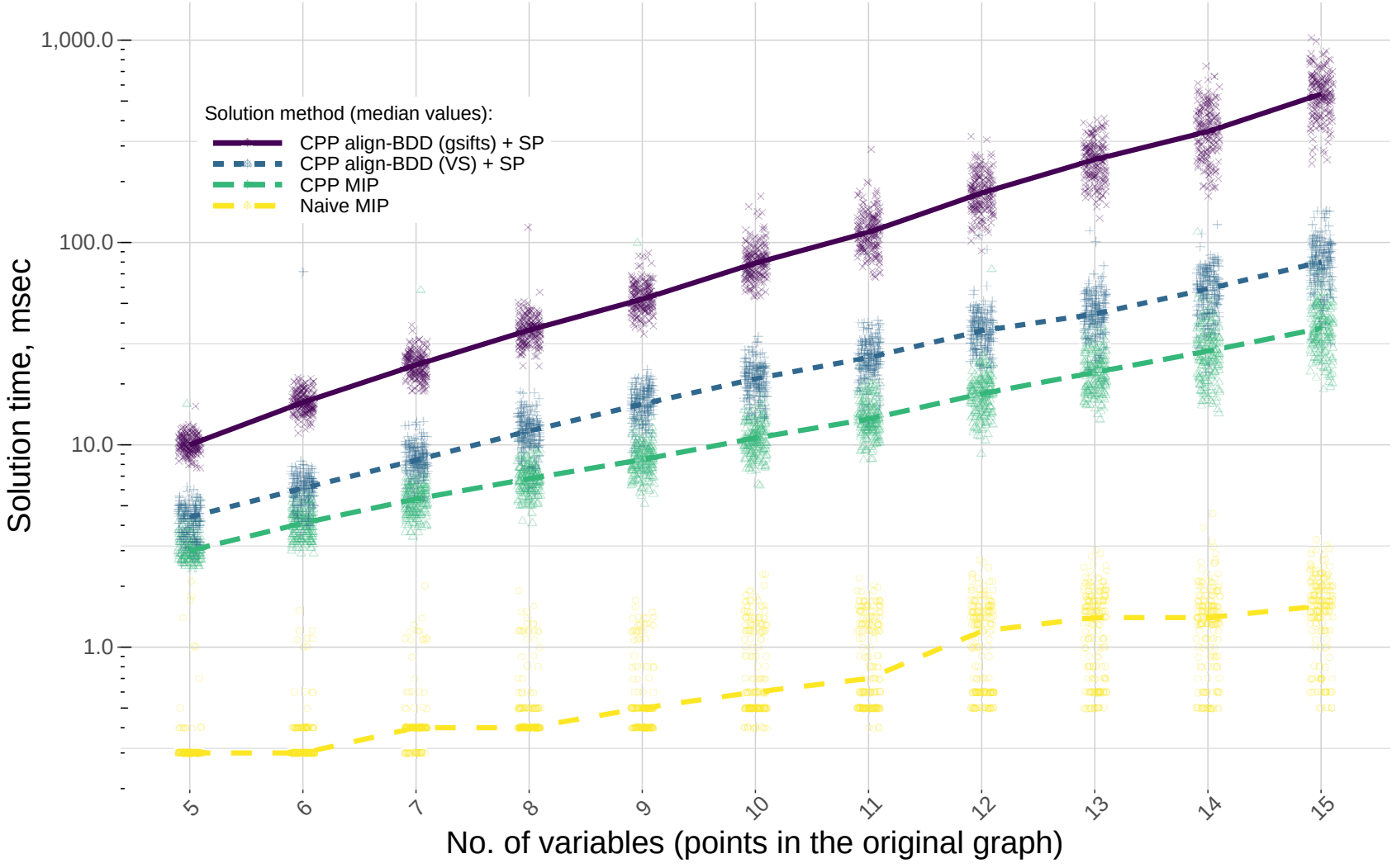
$$\sum_{j \in L^{\mathtt{T}}_q} v^{\mathtt{T}}_{j\mathtt{HI(j)}} = x_q \text{ for all } q = 1, \ldots, M,$$

$$v_{pq} \geq 0 \text{ for all valid } p,\ q; \quad x_i \in \{0, 1\}.$$

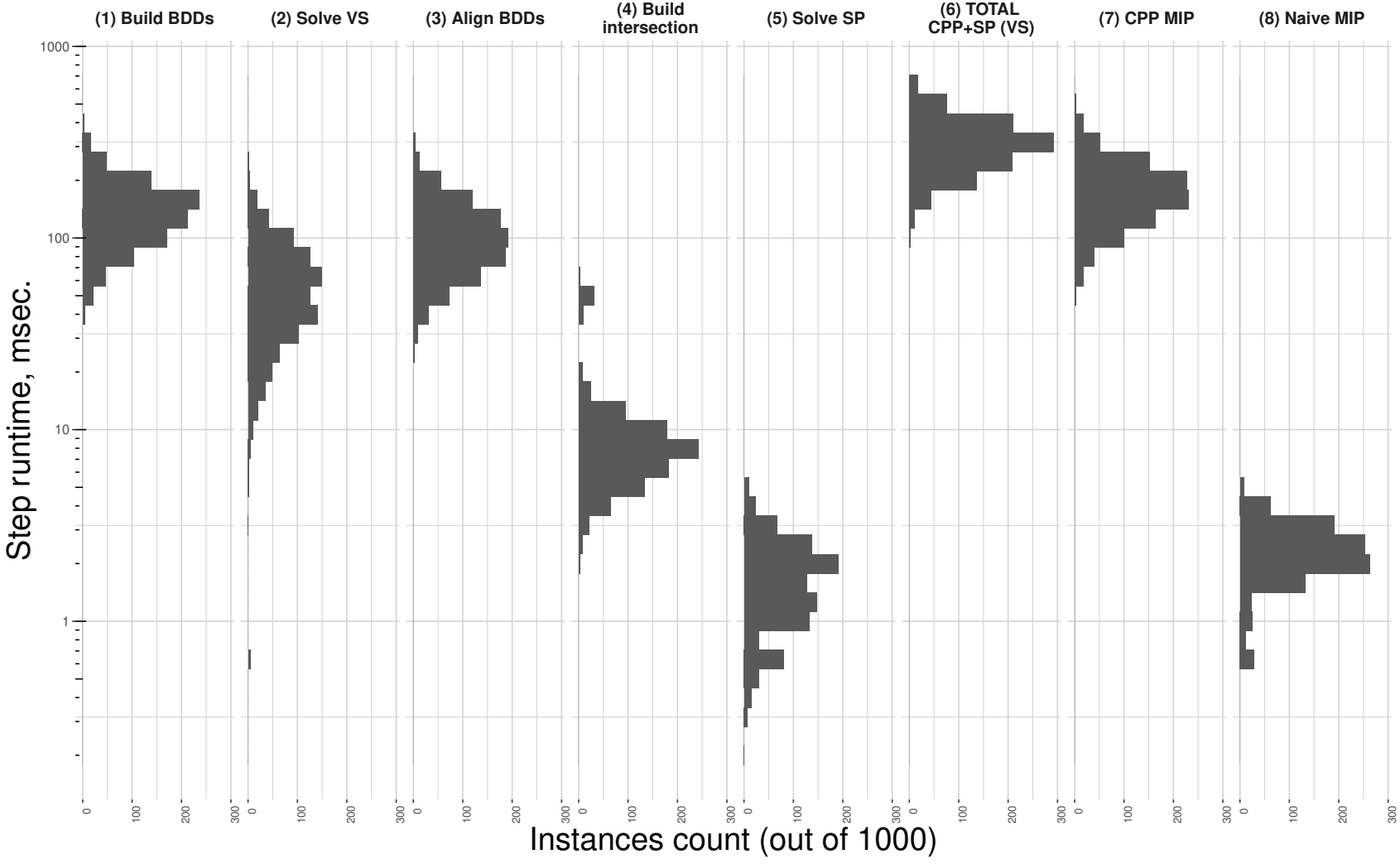
Approaches to solve the problem

- Simple MIP (with $x_i \in \{0, 1\}$ as decision variables),
- CPP + MIP (with continuous BDD flows + x_i as variables),
- CPP $\rightarrow$ Align-BDD $\rightarrow$ shortest-path (no binary variables!)

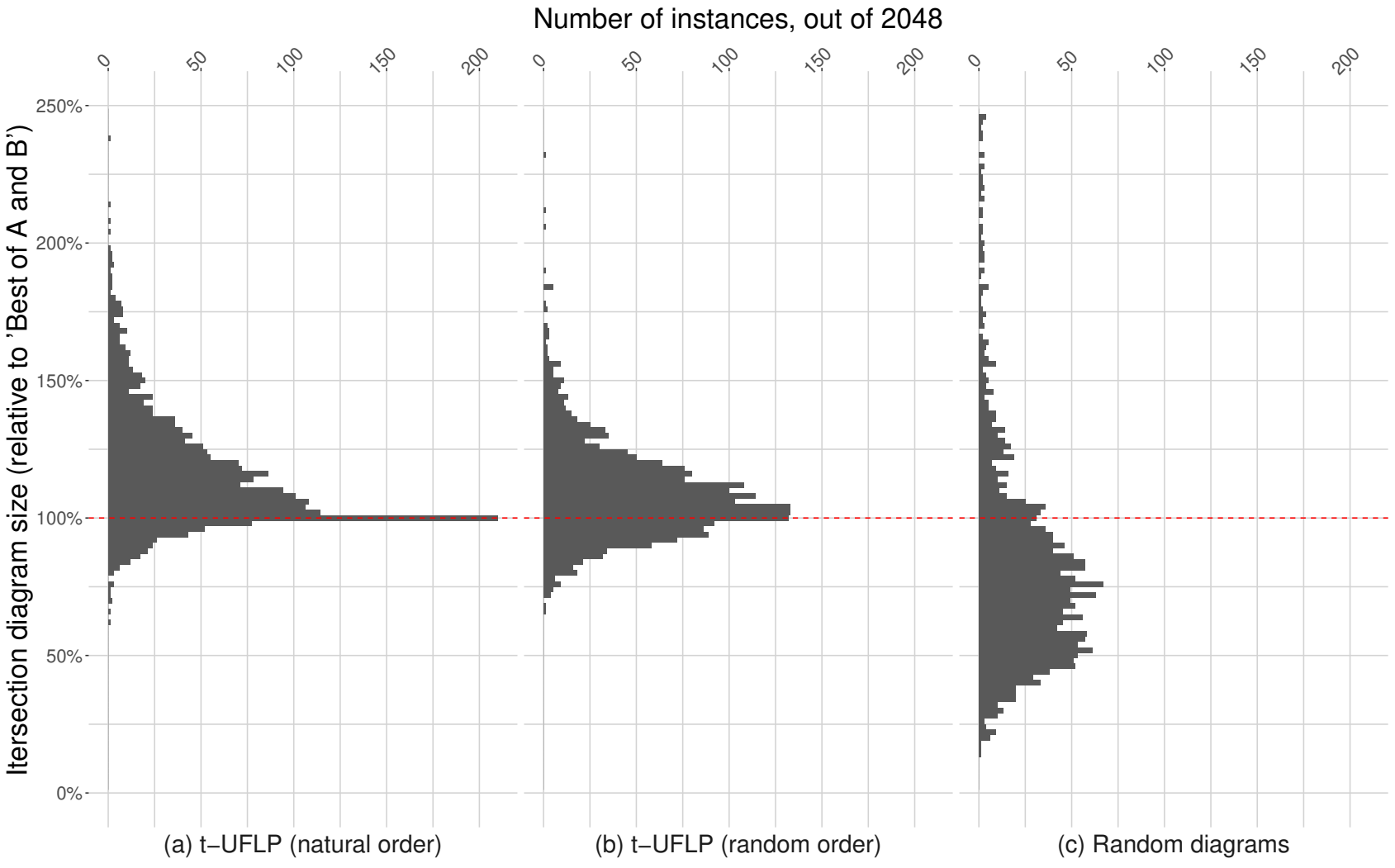
Comparing runtimes (random instances)



Runtimes breakdown



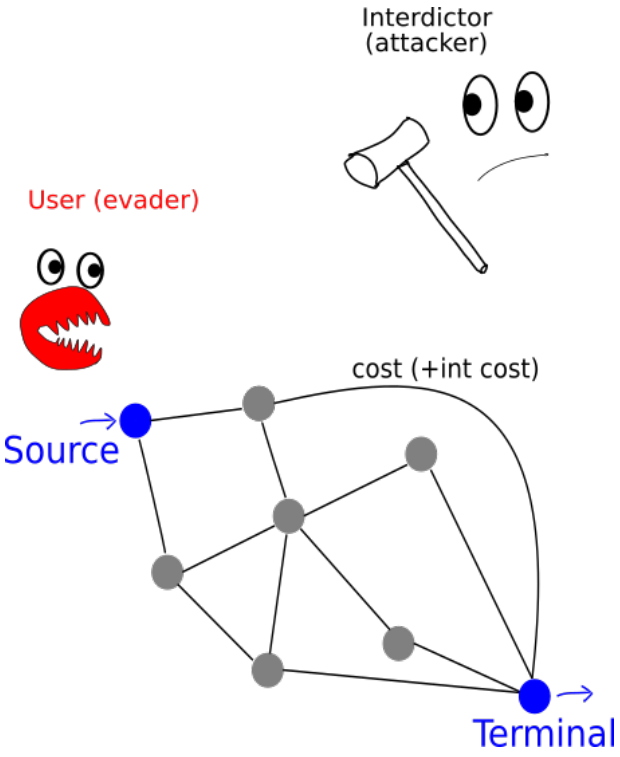
Alignment heuristic performance vs. problem structure



Summary on t-UFLP

- present an illustration for the Align-BDD heuristic (simplified problem),
- introduce a problem, propose a CPP reformulation,
- apply the heuristic to align the diagrams and obtain an LP,
- demonstrate the runtimes in numerical experiments,
- highlight the limits / effects of the problem structure.

A game of "interdiction": intro



- Network: a directed graph with two special nodes (source ⓢ and terminal ⓣ), and a pair of "costs" associated to each edge.
- User: seeks to run through the graph, ⓢ to ⓣ, at min cost.
- Attacker: maximizes the User's cost by "attacking" the arcs, having a limited "budget".

We consider a dynamic version of the game, following [Sefair and Smith(2016)]. (NP-hard)

A game of "interdiction": formulation

The Interdictor's optimal objective z^* can be expressed as:

$$z^*(S,i) = \max_{S' \subseteq \mathrm{FS}(i) \setminus S \,:\, |S \cup S'| \leq b} \left\{ \min_{j \in \mathrm{FS}(i)} \{ z^*(S \cup S', j) + \widetilde{c}_{ij}(S \cup S') \} \right\},$$

where:

- S: interdiction set,
- i: current Evader's node, $\mathrm{FS}(i)$ – forward star of node i,
- $\widetilde{c}_{ij}$: arc traversal costs (given the interdiction),
- b: Interdictor's budget.

A game of "interdiction": formulation

The Interdictor's optimal objective z^* can be expressed as:

$$z^*(S,i) = \max_{S' \subseteq \mathrm{FS}(i) \setminus S \,:\, |S \cup S'| \leq b} \left\{ \min_{j \in \mathrm{FS}(i)} \{ z^*(S \cup S', j) + \widetilde{c}_{ij}(S \cup S') \} \right\},$$

where:

- S: interdiction set,
- i: current Evader's node, $\mathrm{FS}(i)$ – forward star of node i,
- $\widetilde{c}_{ij}$: arc traversal costs (given the interdiction),
- b: Interdictor's budget.

Existing algorithms (by Sefair & Smith)

- polynomial DP algorithm for a DAG
- exact DP algorithm, exp time for general case.

Monte Carlo Tree Search

- Maintain the game tree,
- Try not to create all the nodes,
- Prune the definitely suboptimal ones,
- Drive the tree growth by a computationally cheap objective estimate (e.g., based on simulated games).

The "game tree"

Create a "game tree", where nodes contain the following information.

- Current status
  - $\mathrm{p}(j) \in \mathcal{N}$: where is the Evader,
  - $S_j \subseteq \mathcal{A}$: what is interdicted,
  - $\tau(j)$: who's turn is it (Interdictor/Evader)

The "game tree"

Create a "game tree", where nodes contain the following information.

- Current status
  - $\mathrm{p}(j) \in \mathcal{N}$: where is the Evader,
  - $S_j \subseteq \mathcal{A}$: what is interdicted,
  - $\tau(j)$: who's turn is it (Interdictor/Evader)
- Possible further development
  - $\texttt{children}(j)$: child game tree nodes,
  - $\texttt{actions}(j)$: available actions.

The "game tree"

Create a "game tree", where nodes contain the following information.

- Current status
  - $\mathrm{p}(j) \in \mathcal{N}$: where is the Evader,
  - $S_j \subseteq \mathcal{A}$: what is interdicted,
  - $\tau(j)$: who's turn is it (Interdictor/Evader)
- Possible further development
  - $\texttt{children}(j)$: child game tree nodes,
  - $\texttt{actions}(j)$: available actions.
- Costs info, to drive the search and prune the tree
  - $\widehat{Q}_j$: cost-to-go (starting from this node),
  - $LB(j)$, $UB(j)$: bounds on the true cost-to-go,
  - $N_j \in \mathbb{N}$: how many times the node was visited

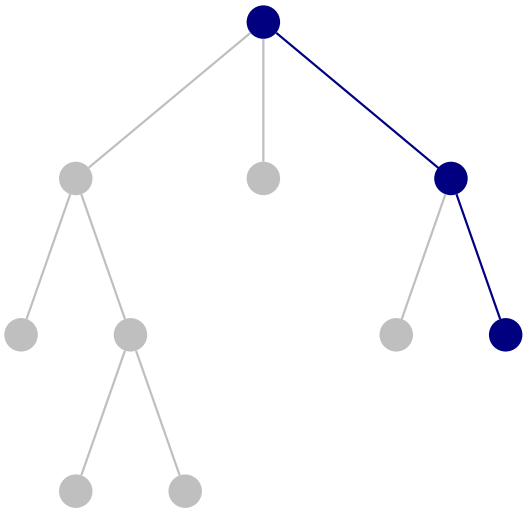
The "game tree"

Create a "game tree", where nodes contain the following information.

- Current status
  - $\mathrm{p}(j) \in \mathcal{N}$: where is the Evader,
  - $S_j \subseteq \mathcal{A}$: what is interdicted,
  - $\tau(j)$: who's turn is it (Interdictor/Evader)
- Possible further development
  - $\texttt{children}(j)$: child game tree nodes,
  - $\texttt{actions}(j)$: available actions.
- Costs info, to drive the search and prune the tree
  - $\widehat{Q}_j$: cost-to-go (starting from this node),
  - $LB(j)$, $UB(j)$: bounds on the true cost-to-go,
  - $N_j \in \mathbb{N}$: how many times the node was visited

  So, we iterate through episodes, each one implying a "full cycle" of the game tree update in four phases.

Phase 1. Selection



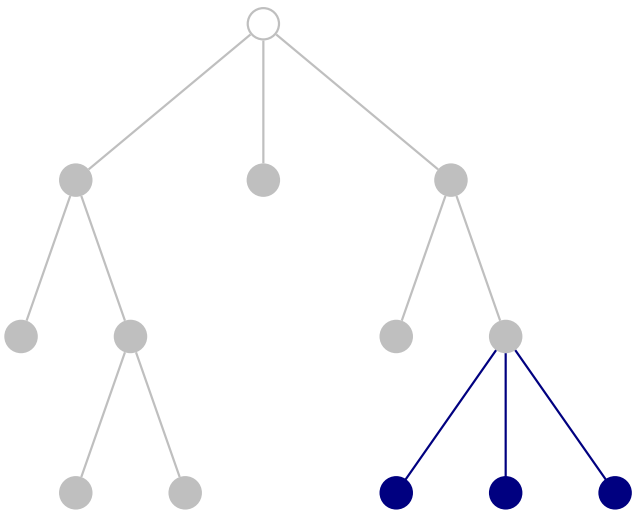
What's happening:

- Start at the root node,
- Use *tree policy* to choose the next node recursively...
- ... pruning nodes as we go, when possible ...
- ... until we reach a leaf.

What's updated:

- Nothing in the tree.
- Along the way: bounds for pruning (more momentarily!) + path costs.

Phase 2. Expansion



What's happening:

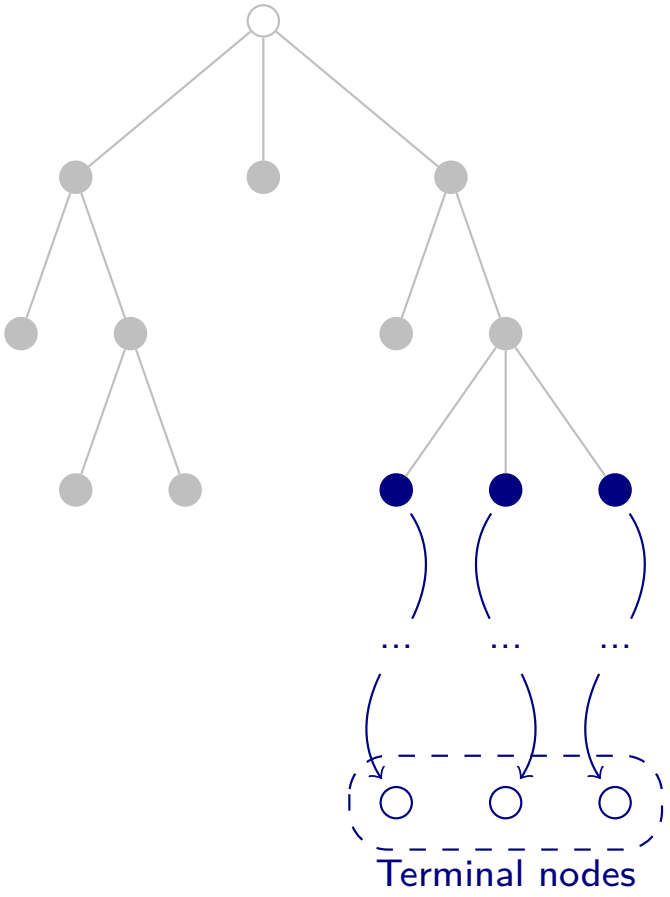
- Create child nodes for possible actions.

What's updated:

- New nodes are created,
- UBs and LBs are calculated

Note: Some inconsistencies can be introduced here, between child and parent nodes.

Phase 3. Roll-outs



What's happening:

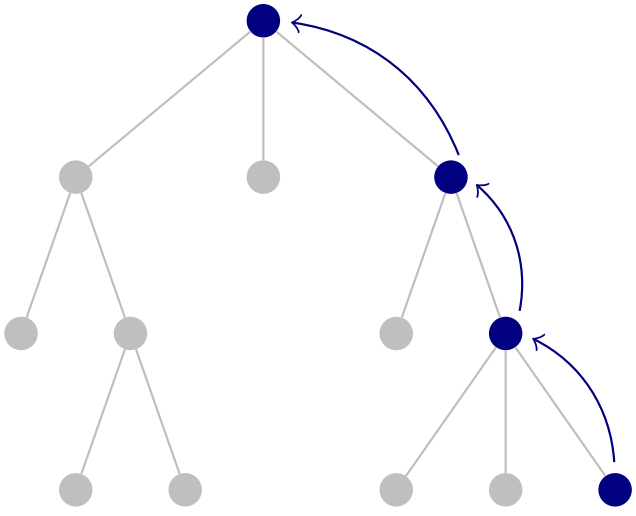
- Run a random simulated game from each node,
- Calculate cost-to-go estimate $\widehat{Q}_j$ as the simulated game cost.

What's updated

- Cost-to-go for each new node.

Note: We do not record the intermediate game states occured during roll-outs!

Phase 4. Backpropagation



What's happening:

- Start at the selected node,
- Recursively update ("propagate") node information for parents ...
- ... until we reach the root.

What's updated: Information in each parent node, using the child nodes:

- UBs and LBs
- Cost-to-go estimate: the best value (given the turn).

The Algorithm

The algorithm can perform actions for both players. Each turn involves two steps:

Step 1. THINK.

We iteratively improve the tree (while we have budget):

FOR $k = 1, \ldots, K$ **DO**

- Selection
- Expansion
- Roll-outs
- Backpropagation

END.

Step 2. ACT.

. . . then pick an action corresponding to the "most attractive" child node of the root.

There are several secret ingredients

SI-1. How to select?

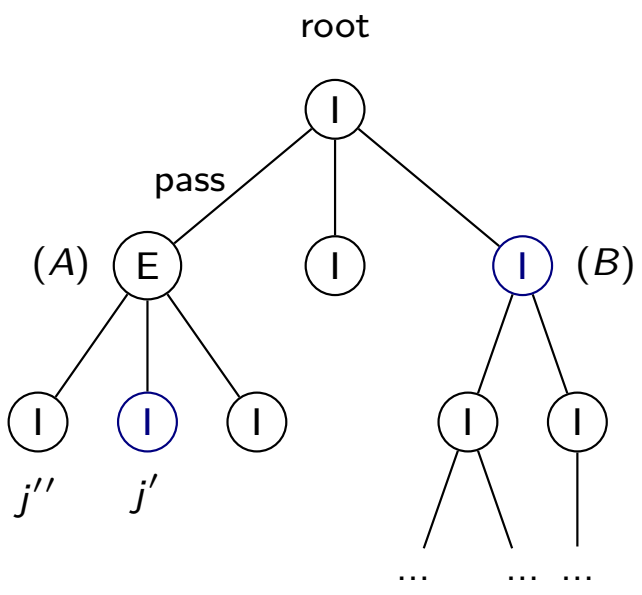
- **with probability** ε: choose at random;
- **otherwise**, a child node with the best score:

$$R_j = \underbrace{\sigma_i(\widetilde{C}_{ij} + \widehat{Q}_j)}_{\text{best cost-to-go}} + \underbrace{C_p\sqrt{\log(N_i)/N_j}}_{\text{encourage exploration}}, \quad \text{for all } j \in \texttt{children}(i)$$

"Best" here depends on the turn (the Evader will choose the smallest cost estimate, the Interdictor — the largest).

SI-2. How to prune?

We leverage the classic idea of alpha–beta pruning:



Maintain two running numbers (bounds):

- α: the worst (minimum) alternative cost achievable by the Interdictor,
- β: the worst (maximum) alternative cost achievable by the Evader.

Pruning condition: $\beta \leq \widehat{\alpha}_j$ or $\widehat{\beta}_j \leq \alpha$, where

- $\widehat{\alpha}_j = \pi_n + LB(j)$ (Interdictor's turns), and
- $\widehat{\beta}_j = \pi_n + \widetilde{C}_{nj} + UB(j)$ (Evader's turns)

SI-3. How to back-propagate?

Assuming the Evader's turn, and i being the current game tree node:

- Update the bounds:

$$UB(i) \leftarrow \min_{j \in \texttt{children}(i)} \left\{ \widetilde{C}_{ij}(S_i) + UB(j) \right\},$$

$$LB(i) \leftarrow \min_{j \in \texttt{children}(i)} \left\{ \widetilde{C}_{ij}(S_i) + LB(j) \right\}.$$

- Update the cost-to-go estimate:

$$\widehat{Q}_i \leftarrow \min_{j \in \texttt{children}(i)} \left\{ \widetilde{C}_{ij}(S_i) + \widehat{Q}_j \right\}.$$

SI-4. Bounds

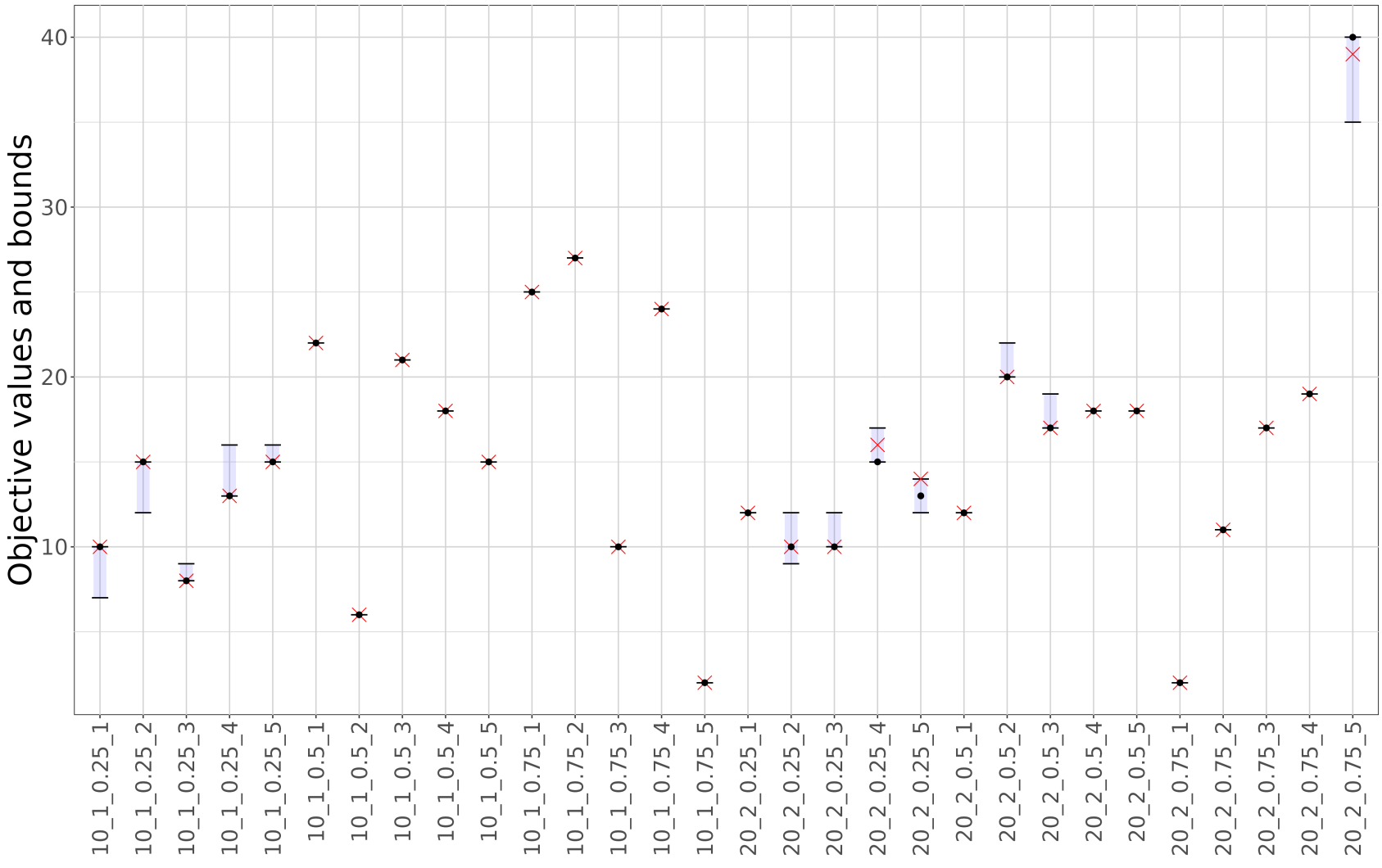
Are inspired by Sefair & Smith 2016:

- **Lower bound:** restrict the Attacker. Make attacks "expire" after a single turn).
- **Upper bound:** restrict the Interdictor. Remove some arcs so that G is a DAG ($\Rightarrow$ DSPI is simple to solve.) Repeat multiple times and choose the best bound.

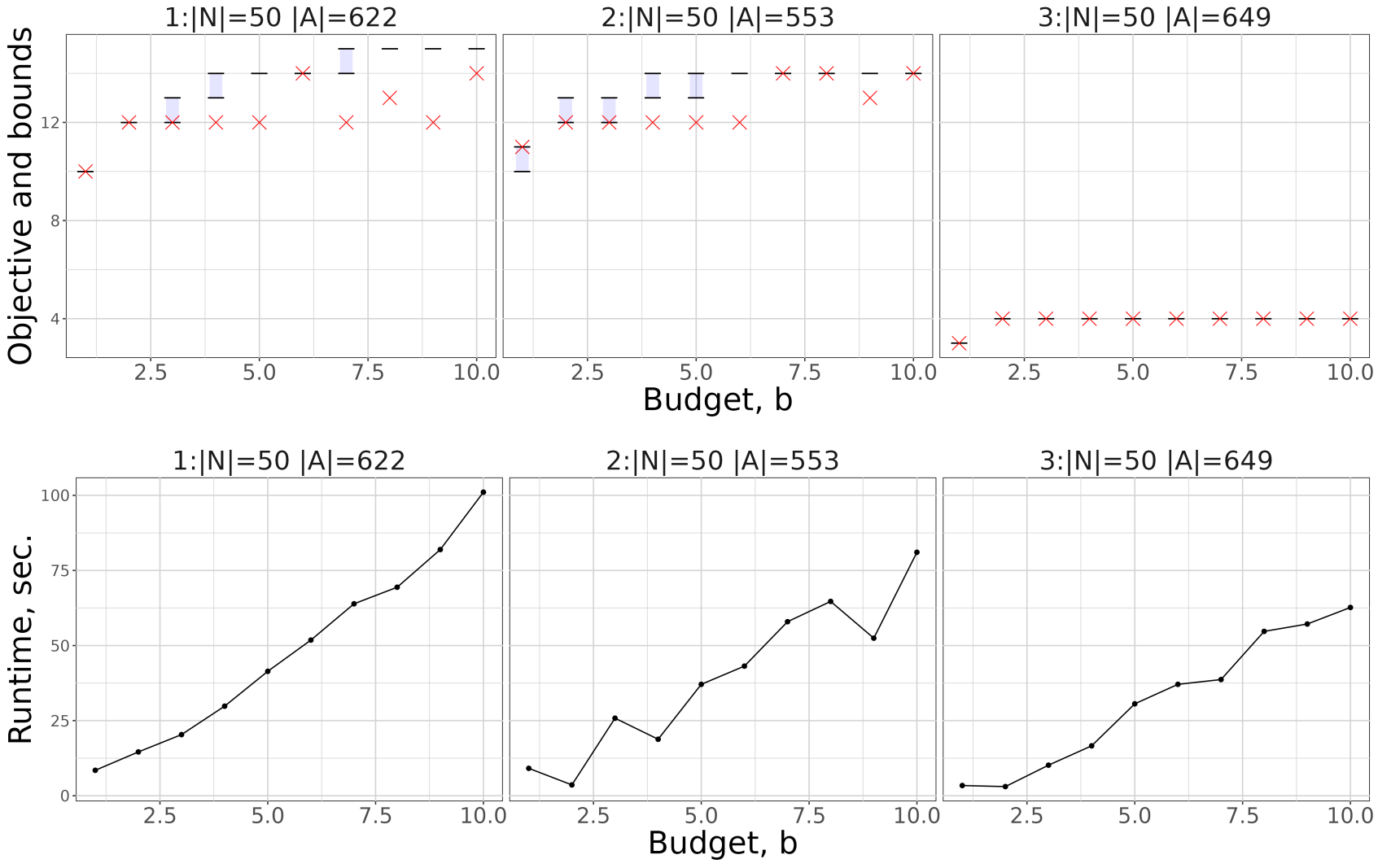
Numerical experiments: strategy

- How does it perform on pre-defined instances? (relative to the known optimum, and to the bounds)
- How does it perform on randomly generated instances with different budgets?
- What's the dynamics of the tree construction? How does the algorithm work?
- What's the point of "playing out", i.e., changing root nodes?

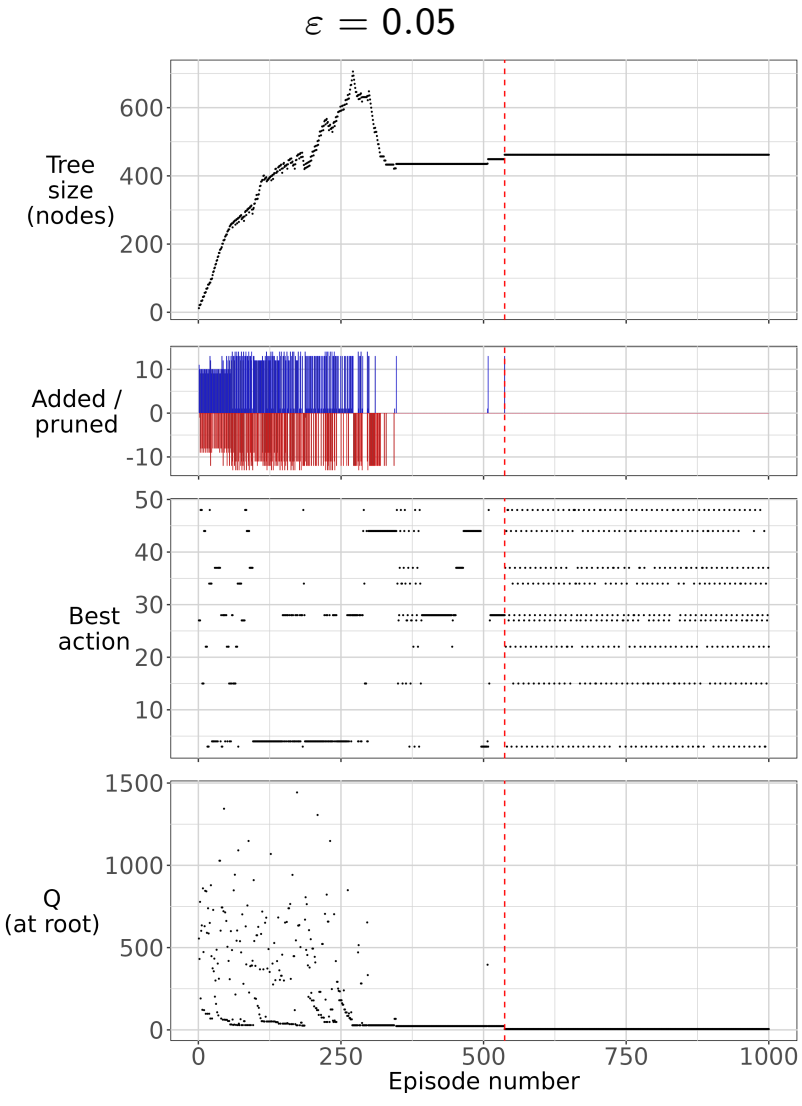
Pre-defined instances

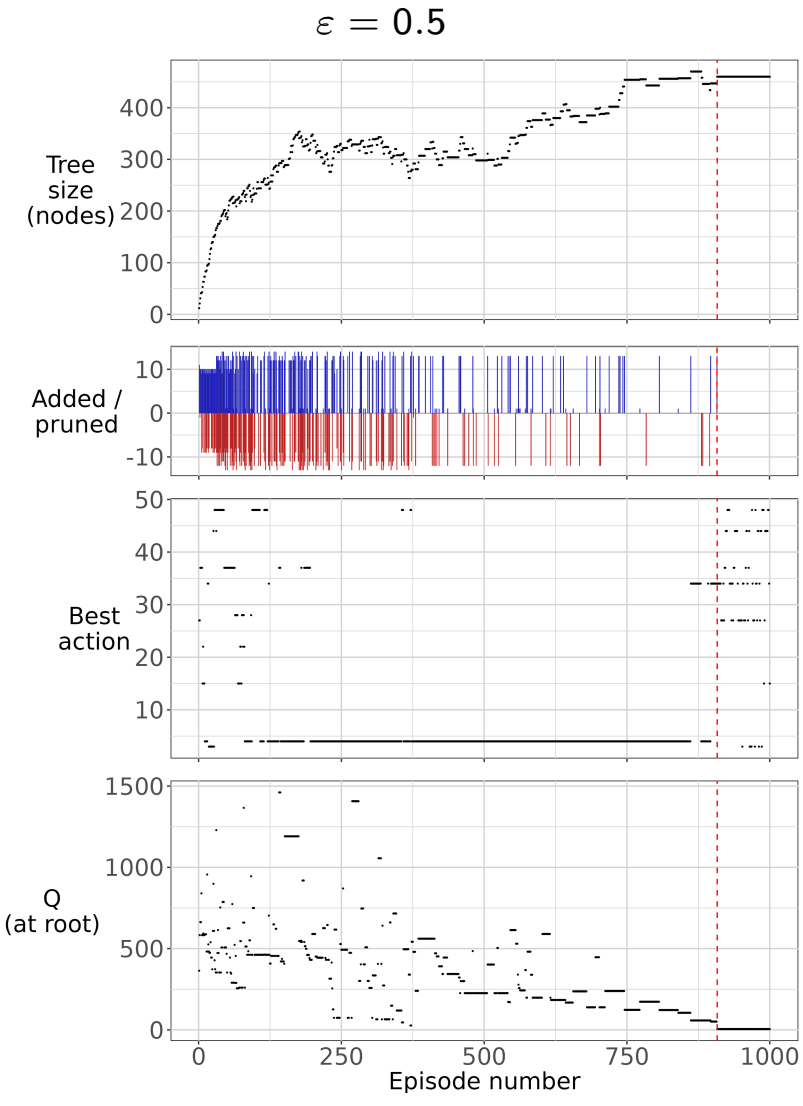


Random instances (snapshot)

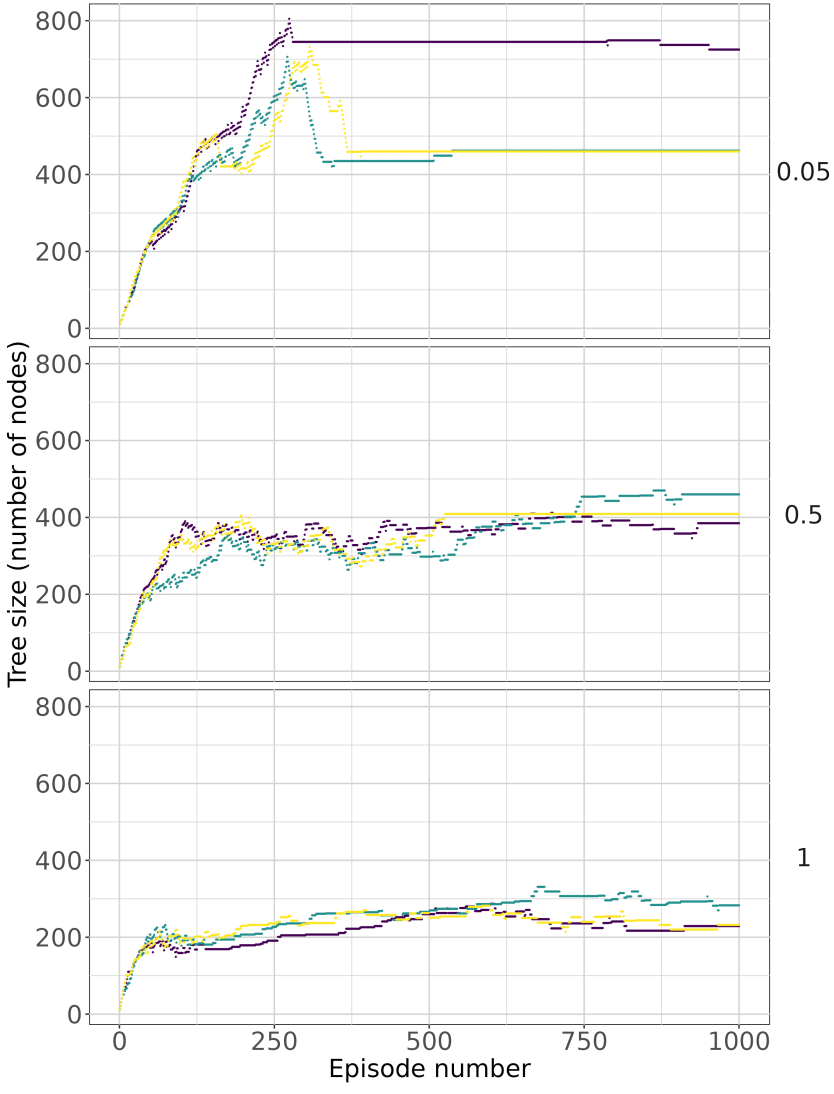


Convergence profiles



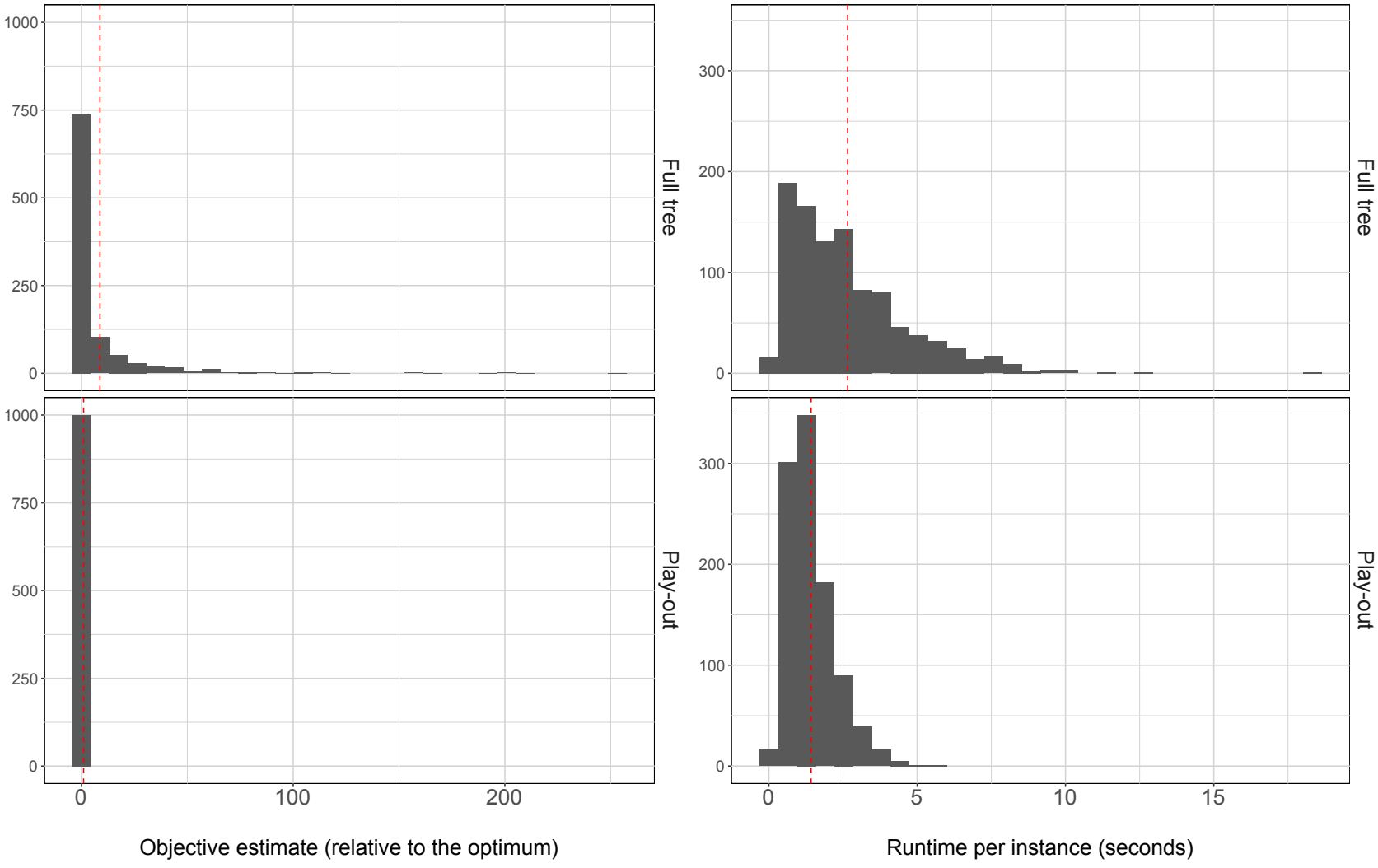


A remark: that's not just different runs



For each value of ε (0.05, 0.5, and 1) we performed three runs, to confirm these are indeed different "modes" of the algorithm.

Play-out vs. first-move strategy



Why does it work? (A sketch on "correctness")

Theorem (Proposition)

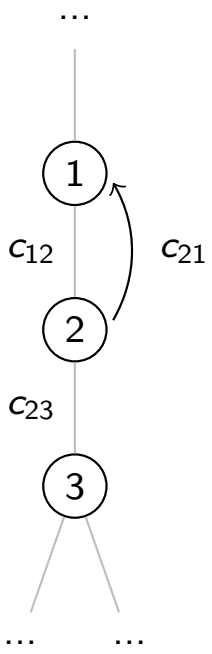
$$\lim_{k\to\infty} \mathbb{P}\{Q^k_{root} = true\ optimum\} = 1$$

A sketch of the proof:

- The game tree has finite number of nodes (there is a bound independent from K).
- We never cut off all the optima $\Rightarrow$ the tree contains at least one.
- What is left is a finite-size minimax tree, containing an optimum.
- As $K \to \infty$, probability to select every node for expansion converges to 1.

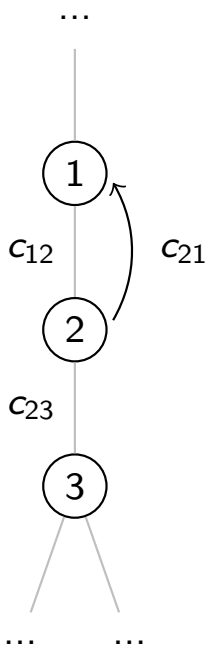
Why in the world the tree is finite?

Network:

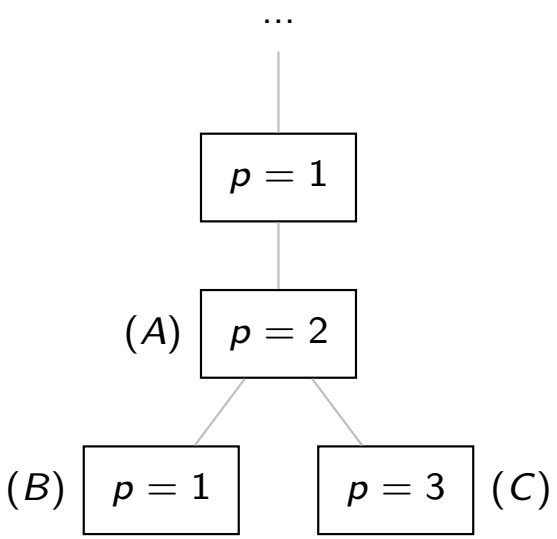


Why in the world the tree is finite?

Network:

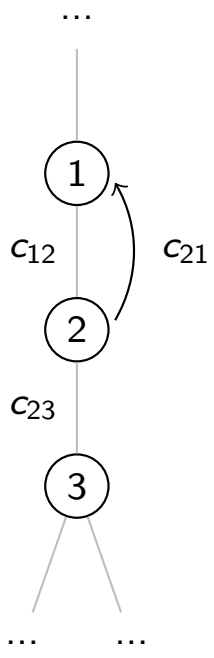


After the first expansion:

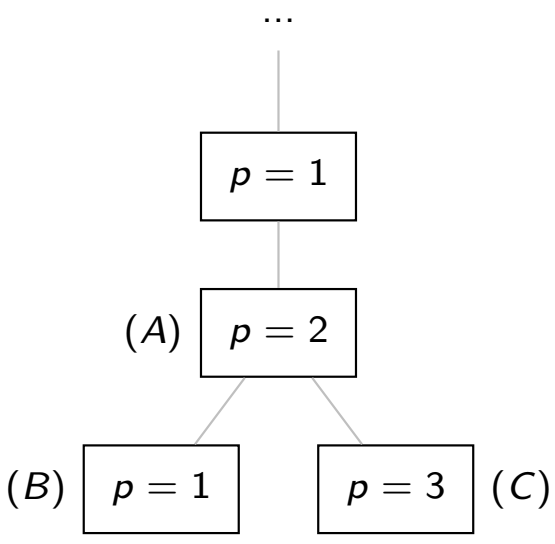


Why in the world the tree is finite?

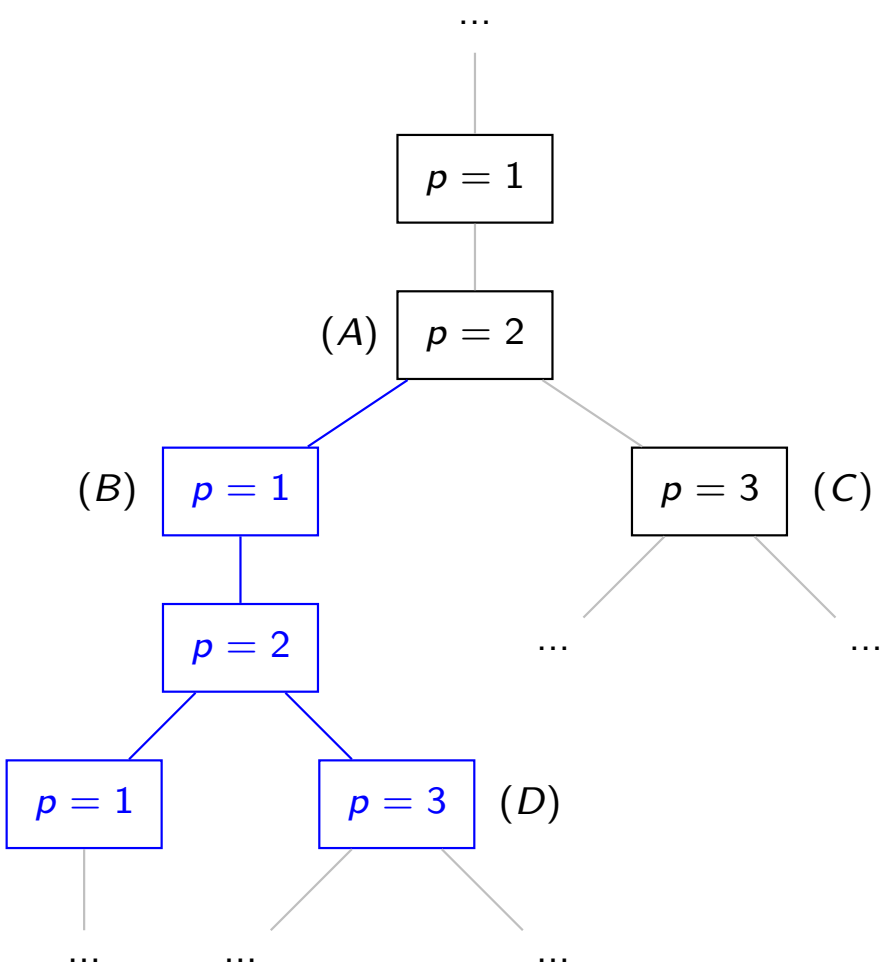
Network:



After the first expansion:



After the second expansion:



Mentioned sources

Leonardo Lozano, David Bergman, and J. Cole Smith.
On the Consistent Path Problem.
Operations Research, 68(6):1913–1931, October 2020.
ISSN 0030-364X.
doi: 10.1287/opre.2020.1979.

Jorge A. Sefair and J. Cole Smith.
Dynamic shortest-path interdiction.
Networks, 68(4):315–330, 2016.
ISSN 1097-0037.
doi: 10.1002/net.21712.