Minimizing Entropy to Discover Good Solutions to Recurrent Mixed Integer Programs

Charly Robinson La Rocca,

Emma Frejinger and Jean-François Cordeau

October 21, 2022

ELLIIT 2022

Université de Montréal

Background

- Our goal is to accelerate the discovery of solutions for real large-scale CO problems that cannot be efficiently solved using current general-purpose solvers.
- Our methodology uses data driven tools within the B&B to estimate the optimal solution.
- <u>Learned heuristic</u>: It produces approximate solutions to get significant speed ups.
- During the last year, we experimented with different applications to figure out how well our method would generalize.
- Key results:
    - Marginal quality loss: Below <u>1%</u> relative gap
    - LAP: <u>5 to 7x</u> runtime speed up on average
    - Network design: <u>2 to 3x</u> runtime speed up on average

Plan

1 Introduction

Structural opportunity for the applications

2 Literature review

A story of trade-offs

3 Methodology

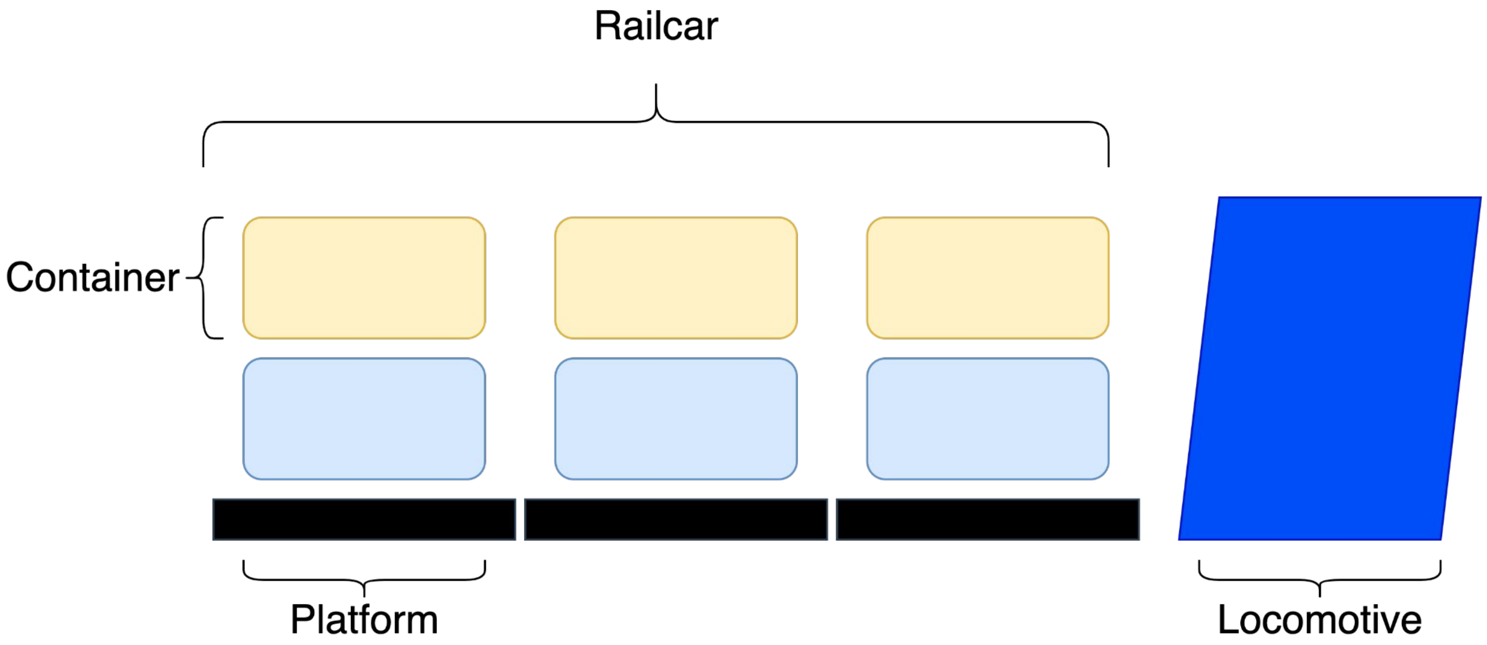
Entropy as a baseline

4 Experiments and discussion

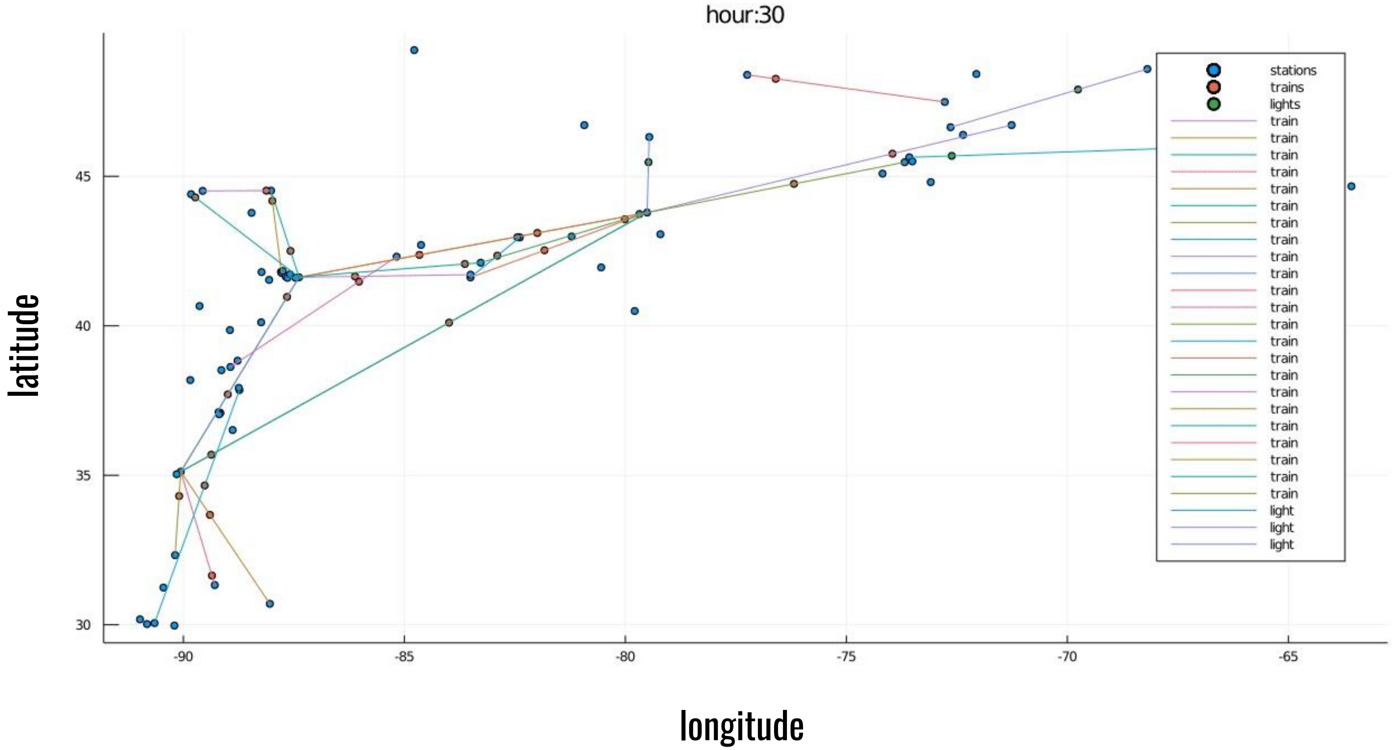
What is the speed up we can expect?

Introduction: LAP

Basic vocabulary



Locomotive assignment problem (LAP)



Locomotive assignment problem (LAP)

Assignment of consist to train

$$\sum_{c \in C_l} y_l^c = 1$$

Flow conservation

$$\sum_{l \in I[i]} x_l^q = \sum_{l \in O[i]} x_l^q$$

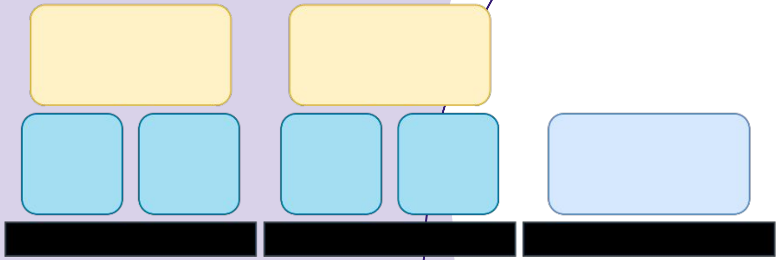
Motivations for ML

01	Compute time can be high	Fast inference time Compute is moved offline
02	Need to be solved on a recurring basis	Model is reusable
03	Large datasets available	Scales with data

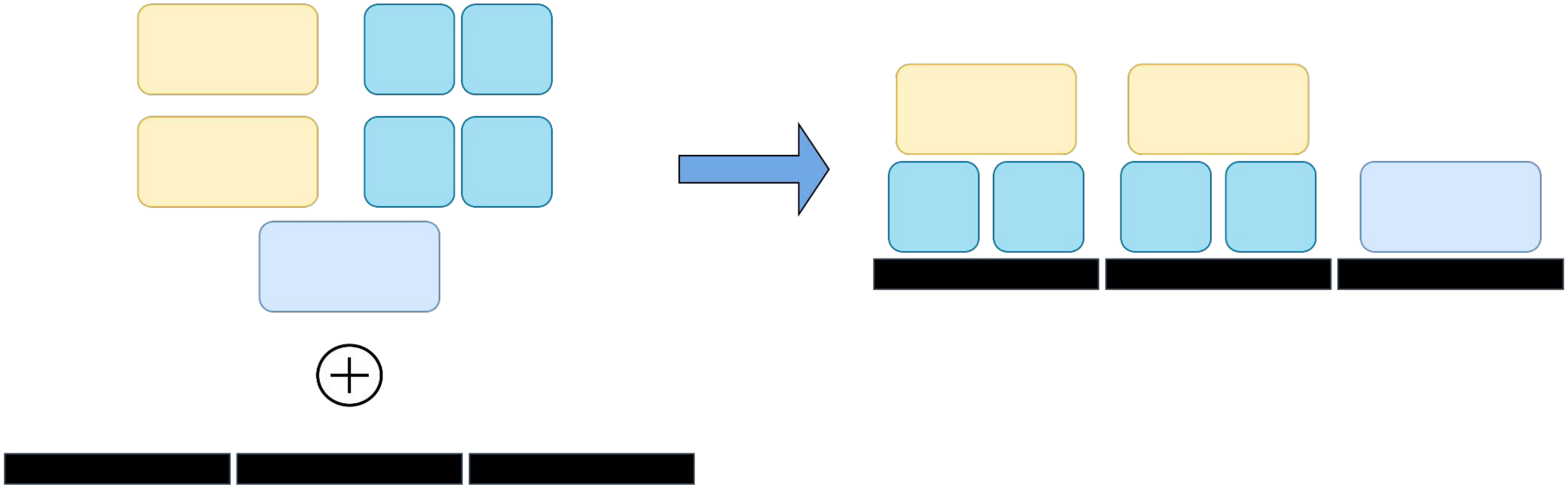
Why a large speed up matters?

01	Strategic or tactical level	Solve large set of instances at the exploratory phase of a project
02	Operational level	Solve instances with last-minute changes
03	Simulation	Solve instances within a simulator (e.g., relocation algorithm in a taxi fleet simulator)

Introduction: LPP



Loading pattern problem (LPP)



Loading pattern problem (LPP)

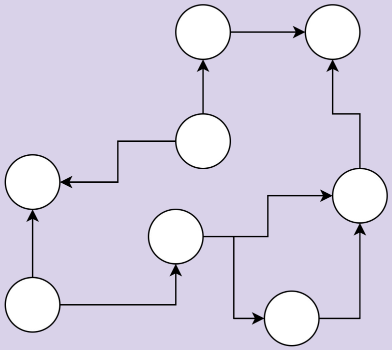
Assignment of pattern to railcar

$$\sum_{k \in K} w_j^k = 1$$

Weight capacity constraint

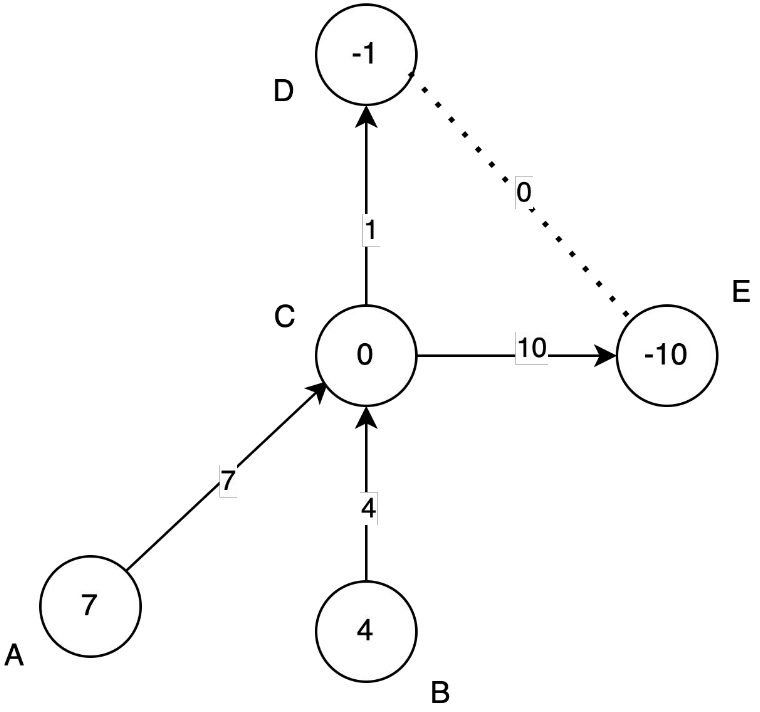
$$\sum_{i \in N} g_i y_{ip} \leq G_p$$

Introduction: Network design



Fixed-charge network design (FCN)

- Challenging
- Open source dataset available (Canad)
- Benchmarks available
- Similar structure to the LAP



Arc:
- Fixed cost
- Variable cost
- Capacity

Node:
- Sink
- Intermediate
- Source

Fixed-charge network design problem (FCN)

Capacity constraint

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} y_{ij}$$

Flow conservation

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = \delta_k$$

Literature review: A story of trade-offs

Literature review

1 Method classification

ML-augmented vs End-to-End

2 Notable trade-offs

Expressiveness vs Sample efficiency

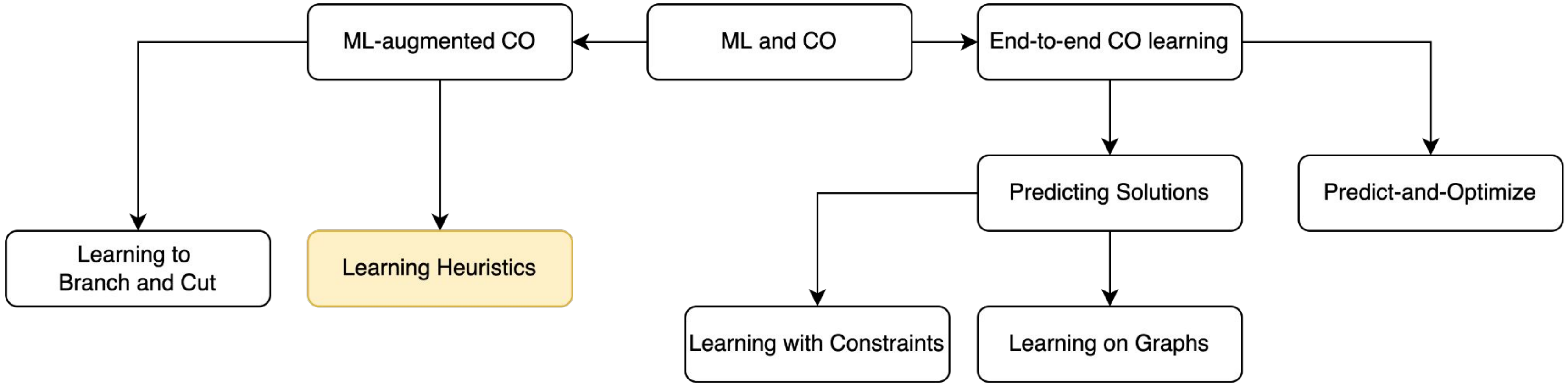
3 Underrated ideas

ML baselines and training cost discount

Noteworthy surveys

- Lodi, A., & Zarpellon, G. (2017). On learning and branching: a survey. *Top*, *25*(2), 207-236.
- Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, *290*(2), 405-421.
- Kotary, J., Fioretto, F., Van Hentenryck, P., & Wilder, B. (2021). End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*.

Machine Learning and Constrained Optimization



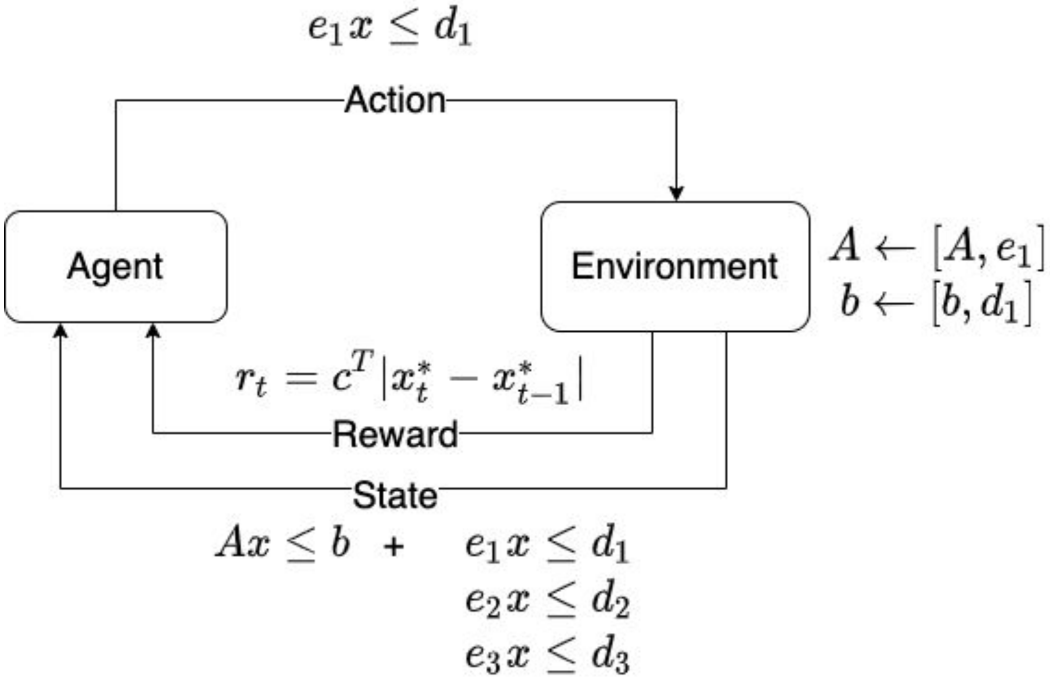
Kotary, J., Fioretto, F., Van Hentenryck, P., & Wilder, B. (2021). End-to-end constrained optimization learning: A survey. *arXiv preprint arXiv:2103.16378*.

Learning and branching

Motivation:

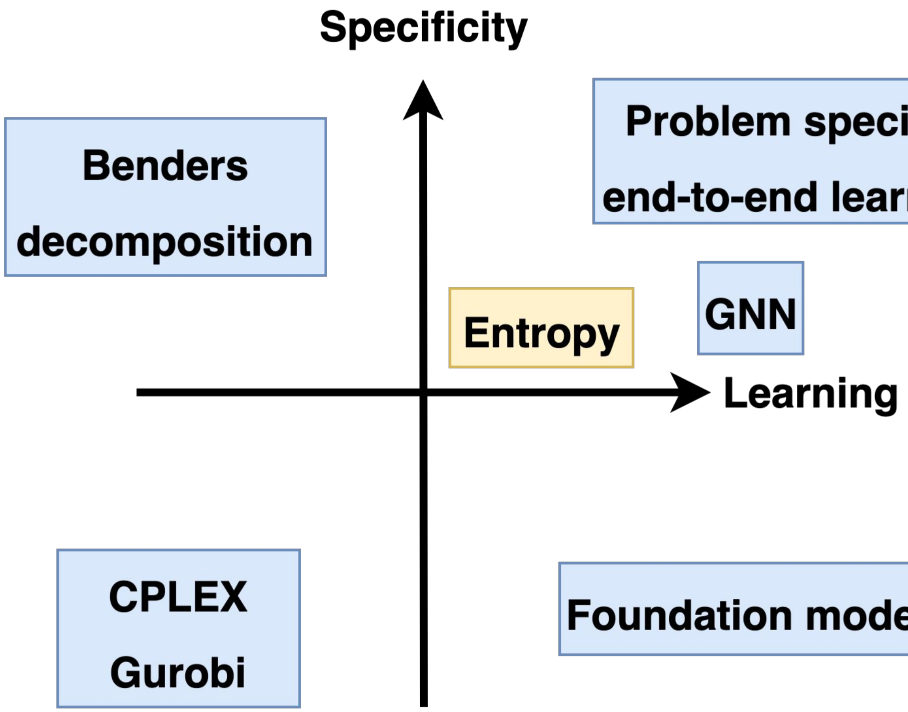
The inclination to use heuristics to deal with the branch-and-bound decisions are justified by the poor understanding from the mathematical standpoint. *There is no deep understanding of the theory underneath branching.*

Lodi, A., & Zarpellon, G. (2017). On learning and branching: a survey. Top, 25(2), 207-236.



Tang, Y., Agrawal, S., & Faenza, Y. (2020, November). Reinforcement learning for integer programming: Learning to cut. In *International conference on machine learning* (pp. 9367-9376). PMLR.

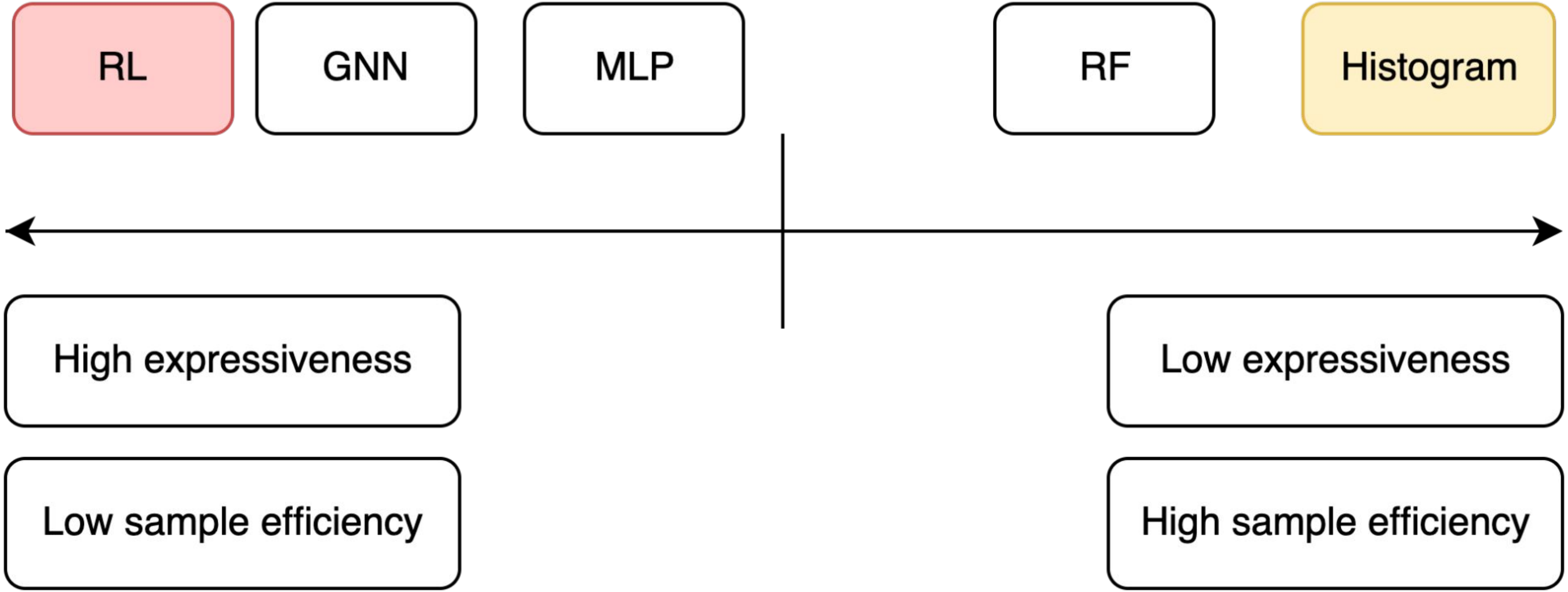
Specificity vs Learning for CO



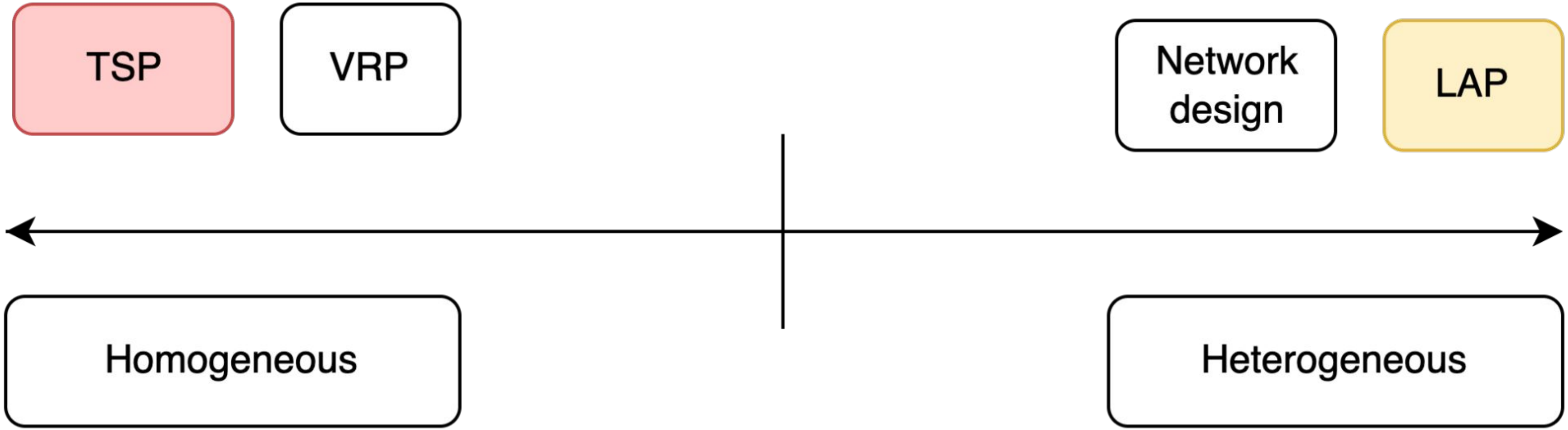
Notable trade-offs

Expressiveness vs sample efficiency

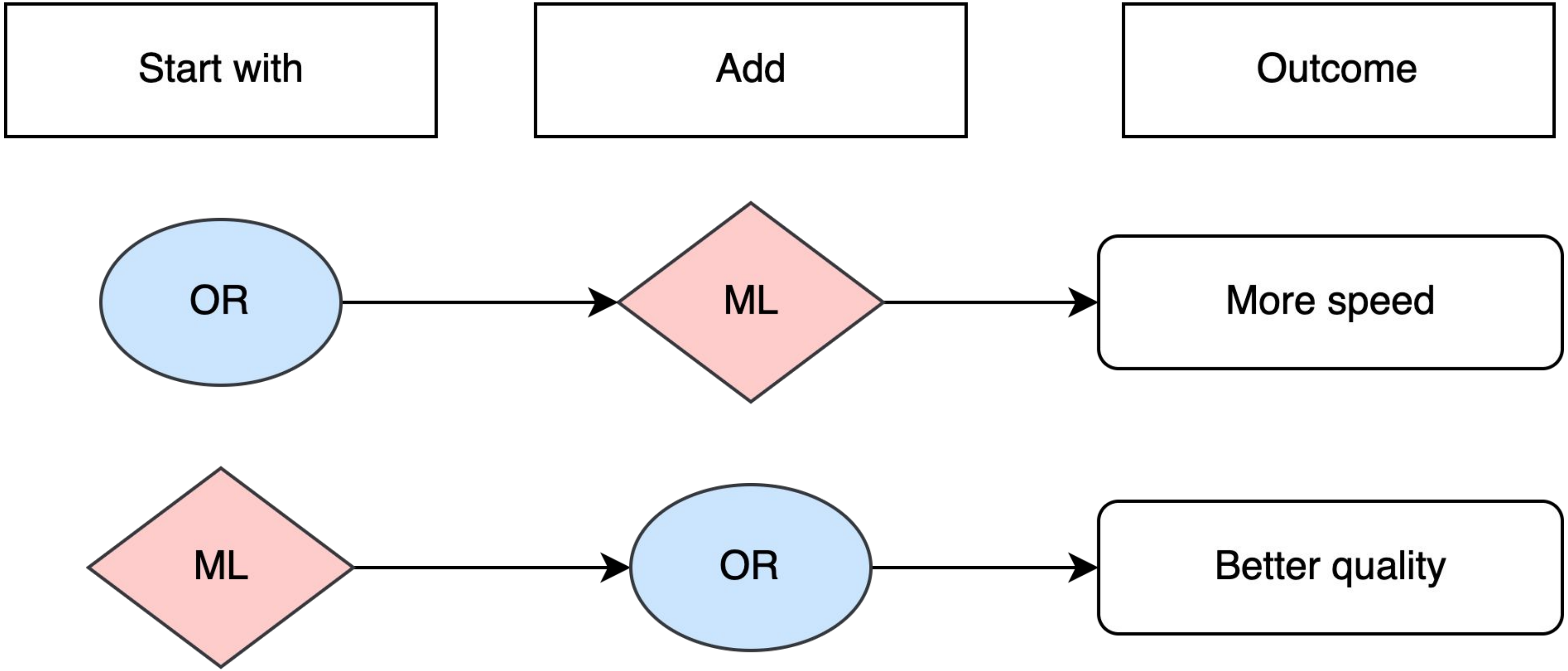
ML Models for CO



Problem instances



Hybridation goals

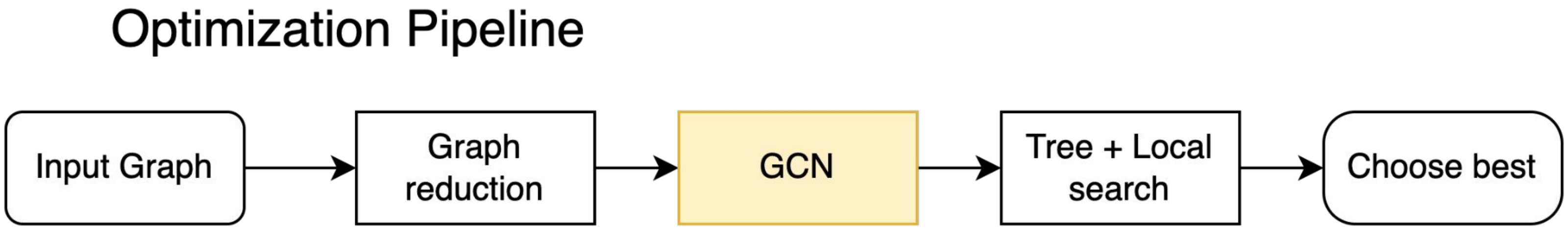


Donti, P. L., Roderick, M., Fazlyab, M., & Kolter, J. Z. (2020). Enforcing robust control guarantees within neural network policies. *arXiv preprint arXiv:2011.08105*.

Underrated ideas

ML baselines and training cost discount

Why a good ML baseline matters?

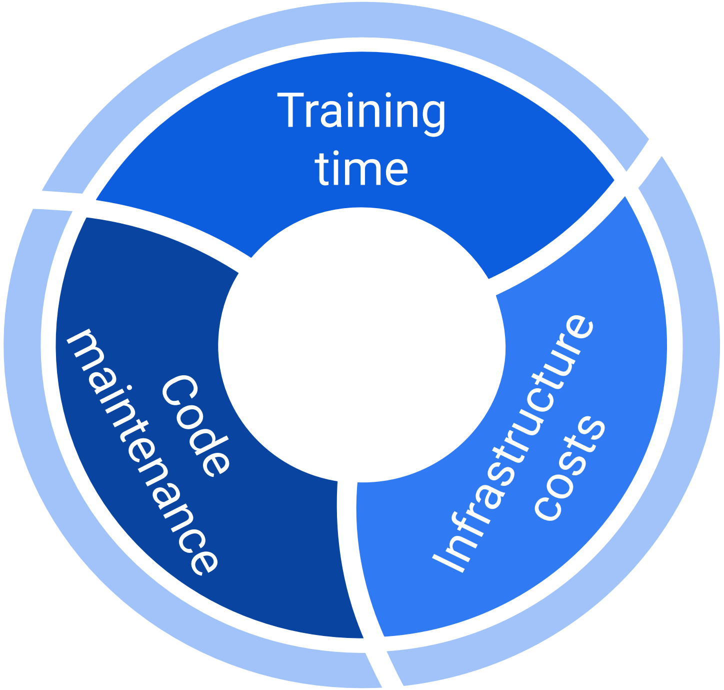


"By re-implementing their algorithm with a focus on code quality and extensibility, we show that the graph convolution network used in the tree search does not learn a meaningful representation of the solution structure, and can in fact be replaced by random values."

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., & Friedrich, T. (2022). What's Wrong with Deep Learning in Tree Search for Combinatorial Optimization. *arXiv preprint arXiv:2201.10494*.

Li, Z., Chen, Q., & Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. Advances in neural information processing systems, 31.

Why training cost discount?



Methodology: Entropy as a baseline

Methodology

1 **Mathematical formulation**

B&B and user cuts

2 **Learning tools**

Entropy and classifiers

3 **Workflows**

Training, validation and testing

Mathematical formulation

$$\min c^T \mathbf{x}$$

$$\textbf{s.t.}$$

$$A\mathbf{x} \leq b$$

$$x_j \in \{0, 1\} \qquad \forall j \in \mathcal{B}$$

$$x_j \in \mathbb{Z} \qquad \forall j \in \mathcal{Q}$$

$$x_j \geq 0 \qquad \forall j \in \mathcal{P}$$

Mathematical formulation

Special Ordered Sets of type 1 (SOS1 or S1) are a set of variables, at most one of which can take a non-zero value, all others being 0.

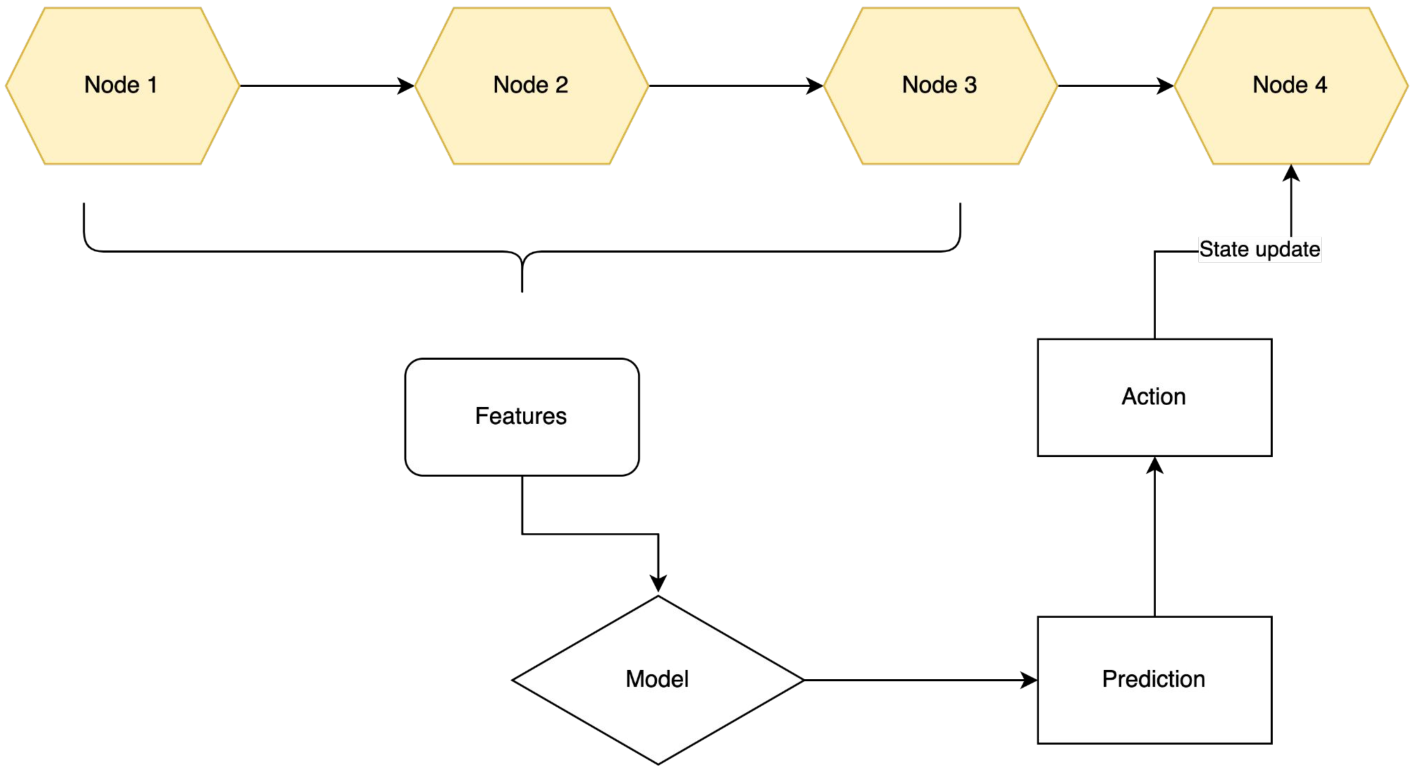
$$\sum_{k \in K} x_v^k = 1 \qquad \forall v \in V$$

K the set of classes available

V the set of objects in the instance

x_v^k binary variable that models the assignment of the object to the class

B&B integration



B&B integration: Code example

```
function algorithm_callback(state::StateType, algo::MIPAlgorithm)
    if should_trigger(state, scheduler(algo))
        actions = make_actions!(state, algo)
        map(actions) do action
            apply!(state, action)
        end
    end
end
```

User cut specification

<u>Action</u>: adding a constraint to limit the search space

$$x_v^k = 1$$

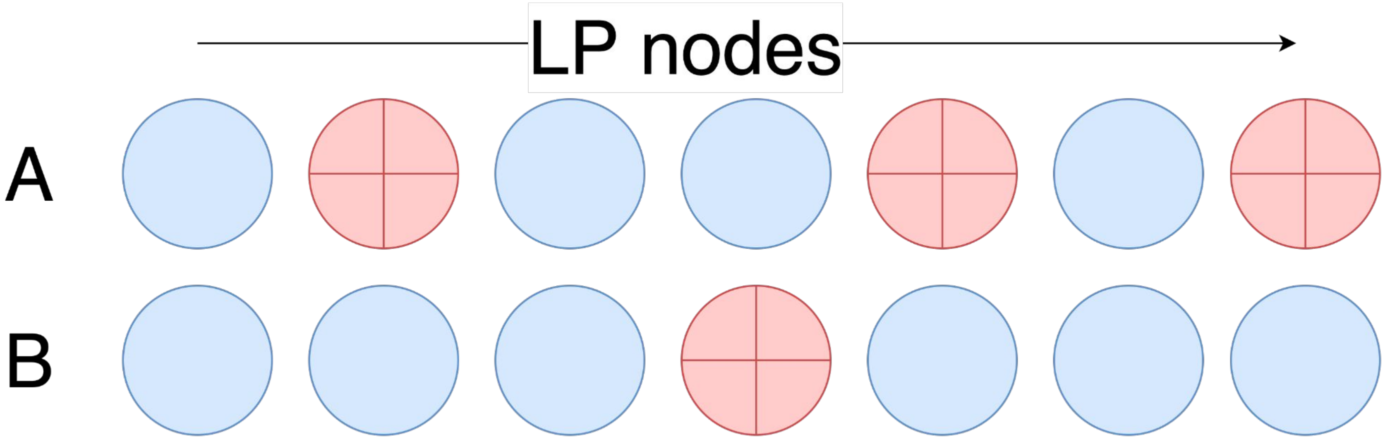
```
constraint = @build_constraint(state.model[:x][v, k] == 1)
MOI.submit(state.model, MOI.UserCut(state.cb_data), constraint)
```

JUMP https://jump.dev/JuMP.jl/stable/

Learning tools

Entropy and classifiers

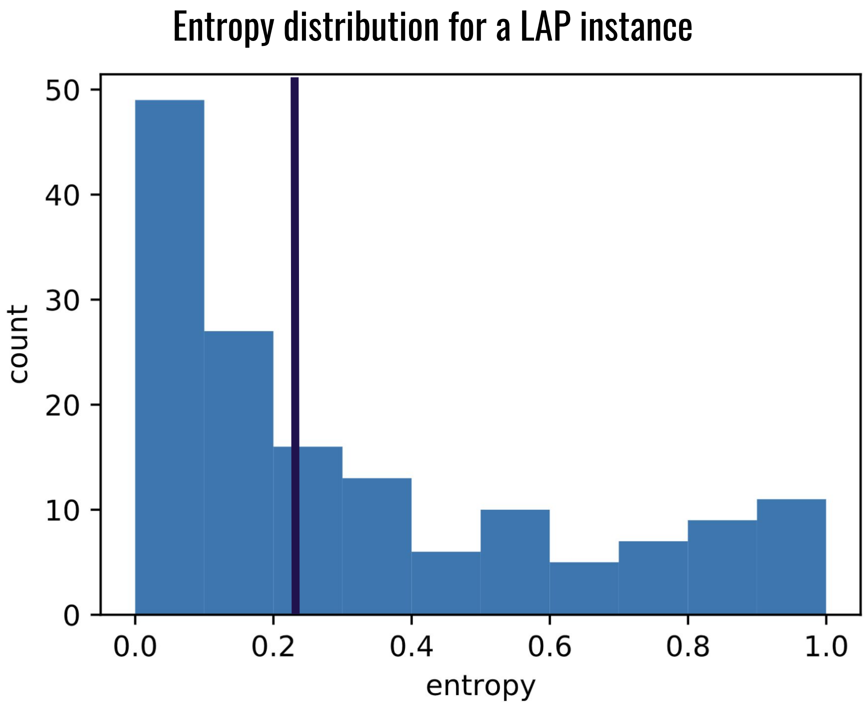
Minimize entropy to reduce assignment risks



$$H(Z) = -\sum_i P(z_i) \log P(z_i)$$

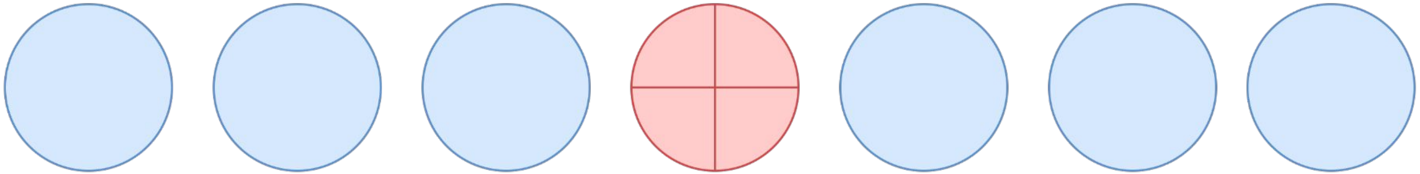
$H(A) > H(B)$ ➡ B is more <u>stable</u> than A.

Empirical motivation for entropy



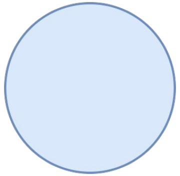
ML baseline: Histogram

B



$$P(Blue) = 6/7$$
$$P(Red) = 1/7$$

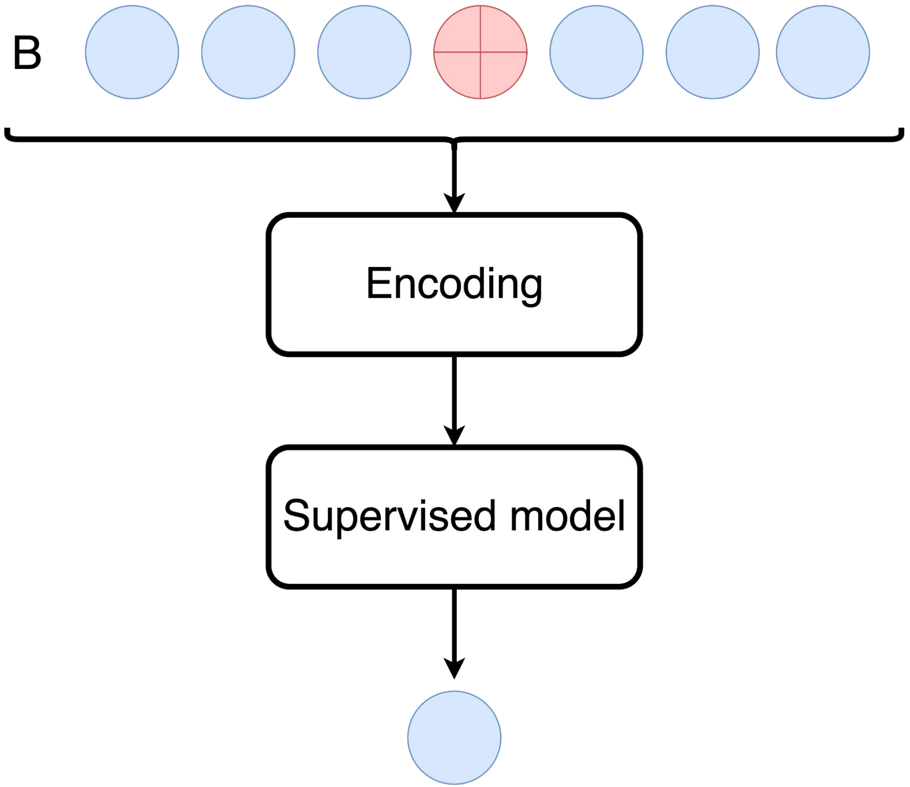
$$\Rightarrow \; argmax(P(Blue), P(Red)) \; \Rightarrow$$



Supervised learning model

Why Random Forest (RF) vs DNN?

- Little tuning required
- Small footprint
- No GPU required
- Reaches good accuracy while trained on a single instance
- No gradient available



Online features

- Feature vector updates at every node
- Contains relevant data from the visited solutions

$$\phi_t(v) = [\mathrm{mean}(K_{vt}), \mathrm{var}(K_{vt}), \max(K_{vt}), \min(K_{vt})]$$

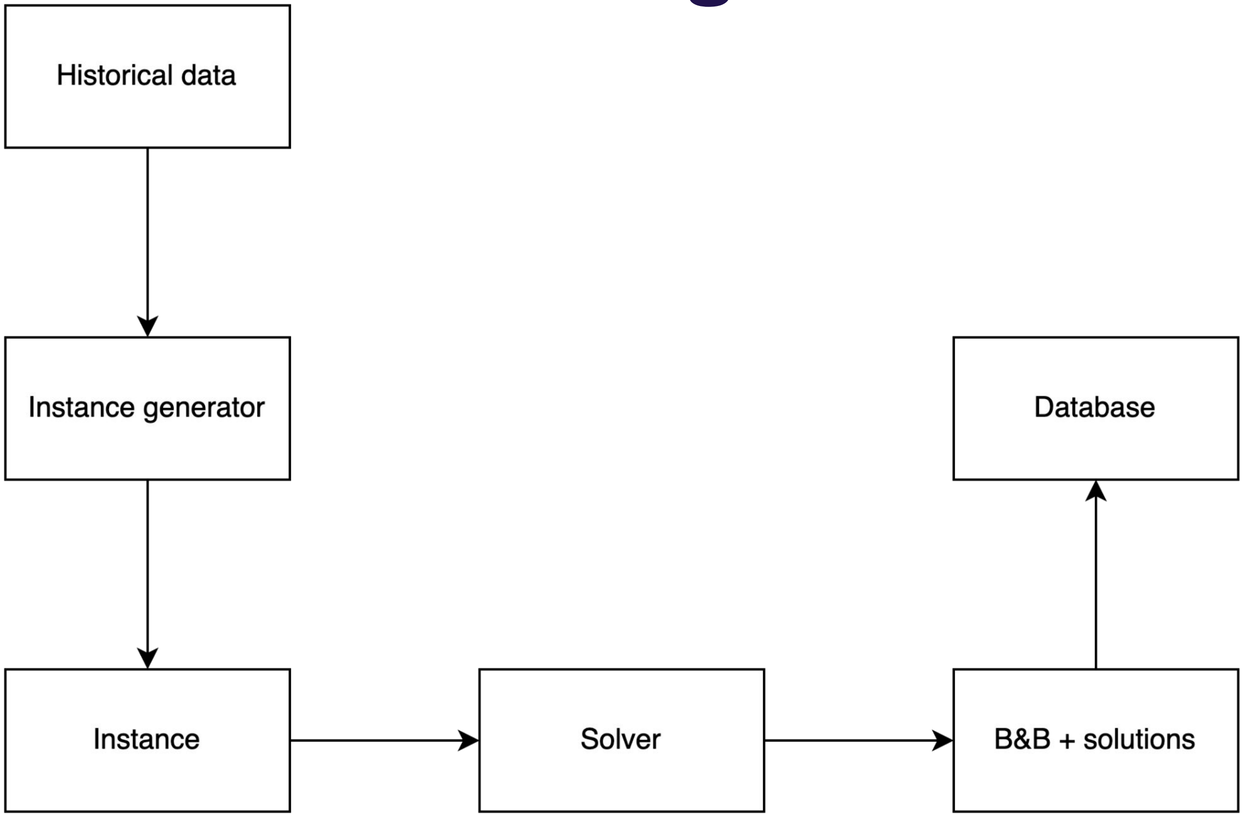
$$K_{vt} = [k_{v1}, k_{v2}, ..., k_{vt}]$$

k_{vt} the class of the object at the current iteration

Workflows

Training, validation and testing

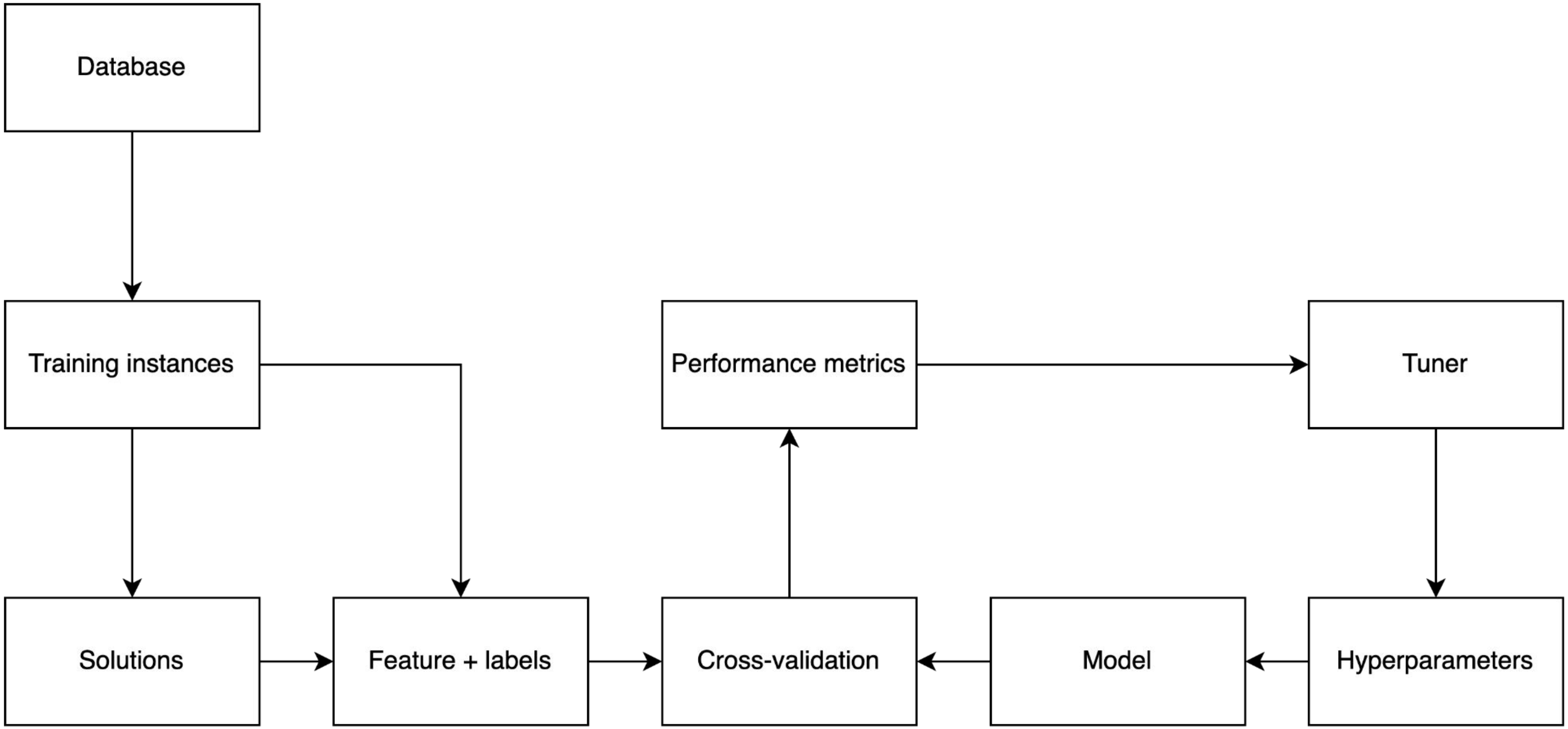
Data generation



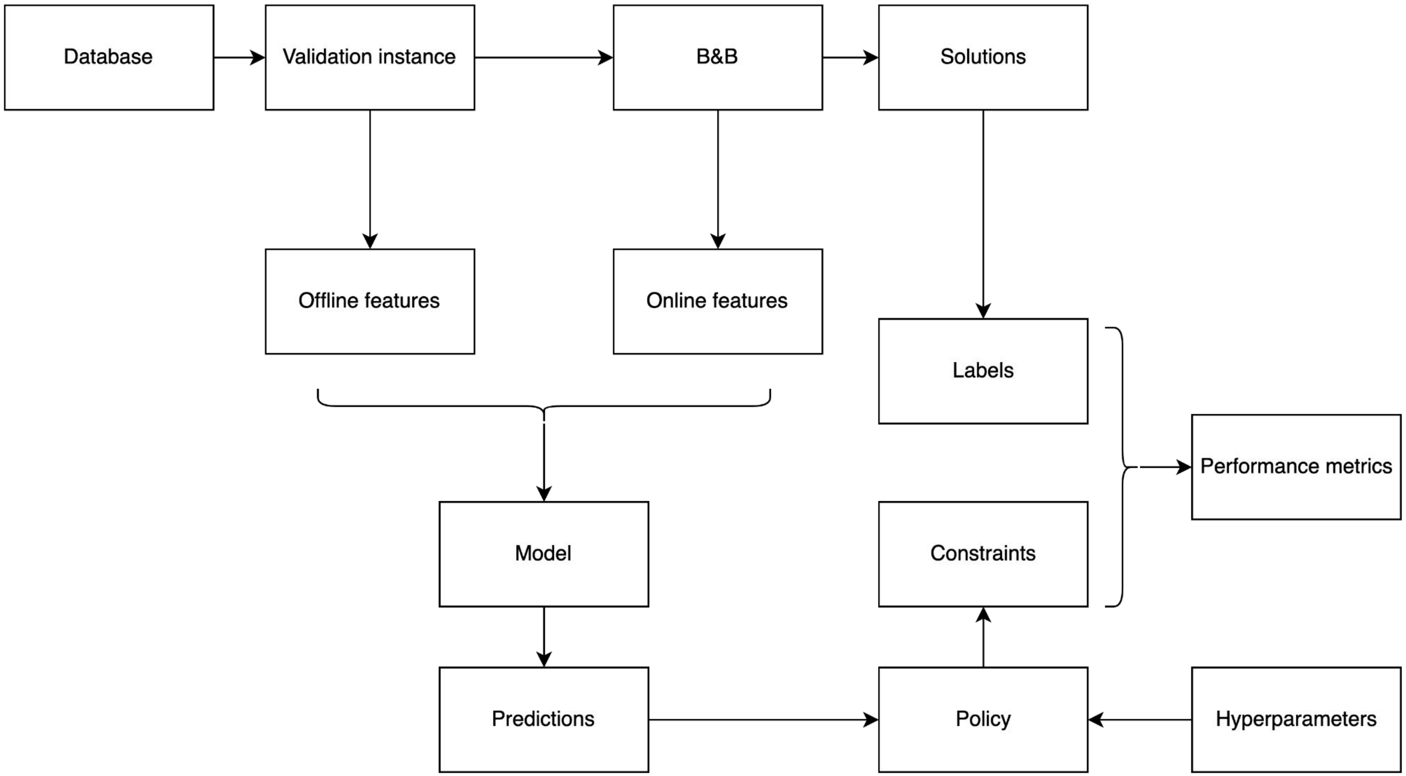
Notes:

- LAP uses historical data from CN
- FCN uses Canad instances
- LPP uses synthetic instances

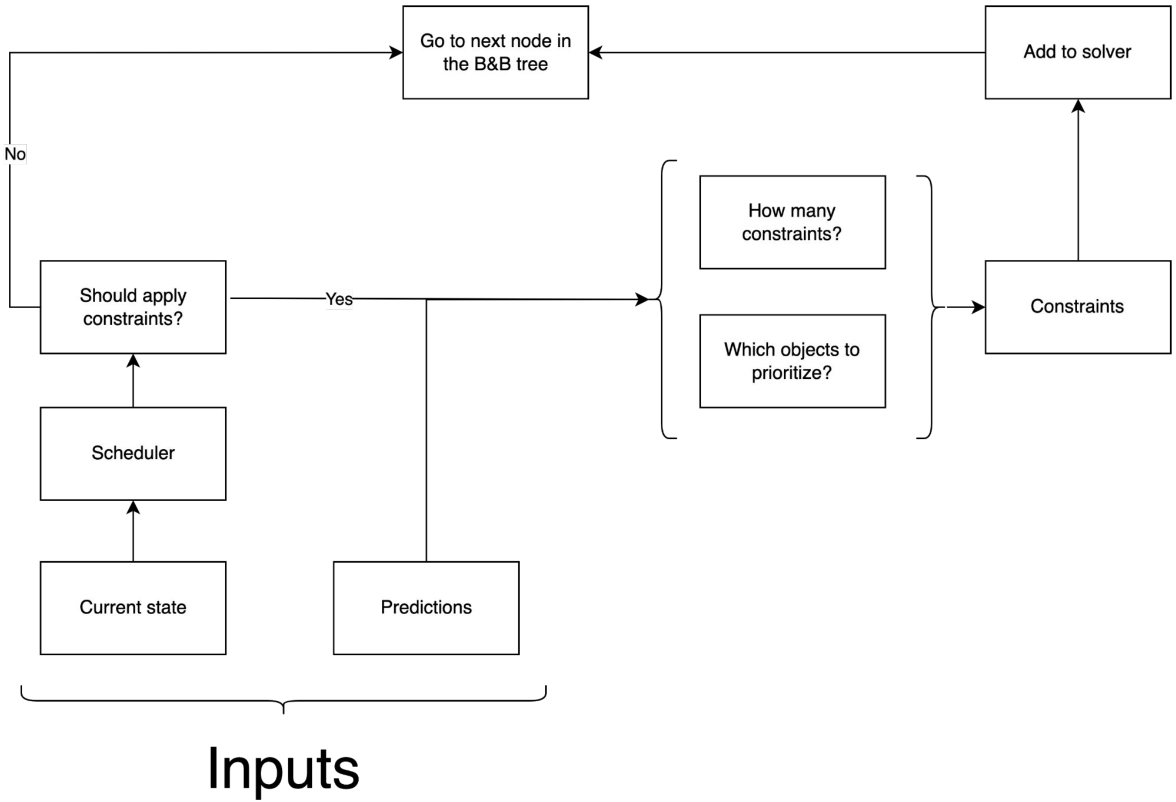
Training


Database
Training instances
Performance metrics
Tuner
Solutions
Feature + labels
Cross-validation
Model
Hyperparameters

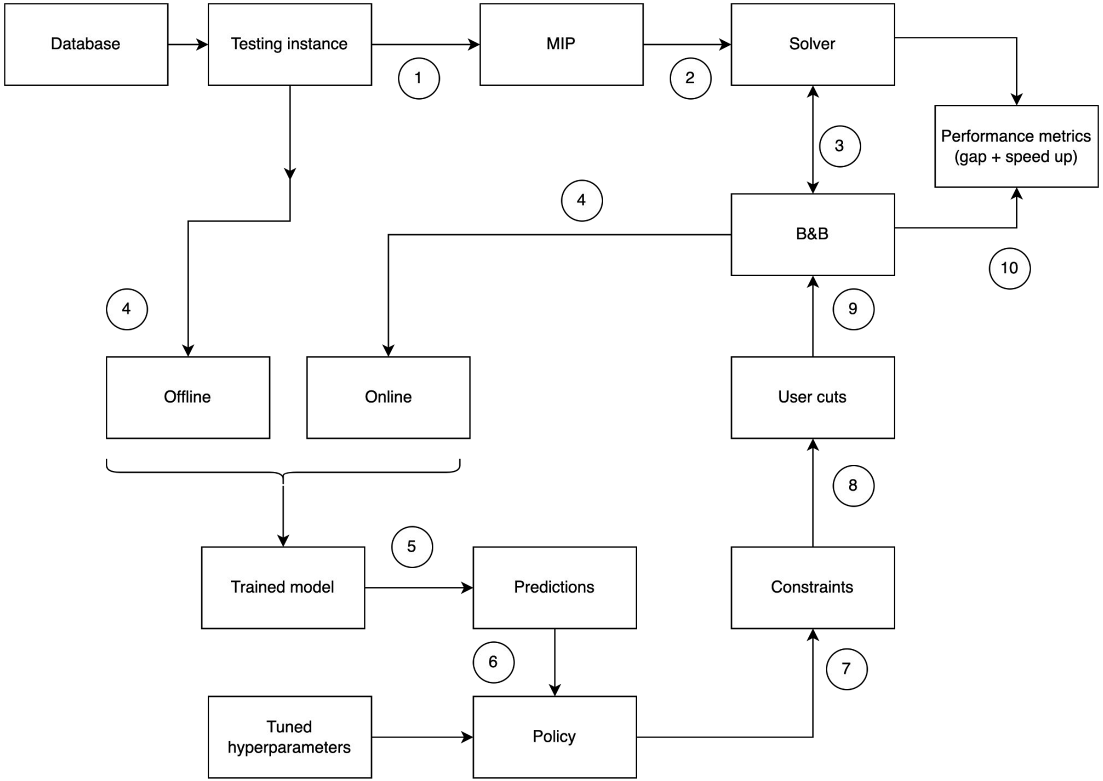
Validation/Simulation



Policy

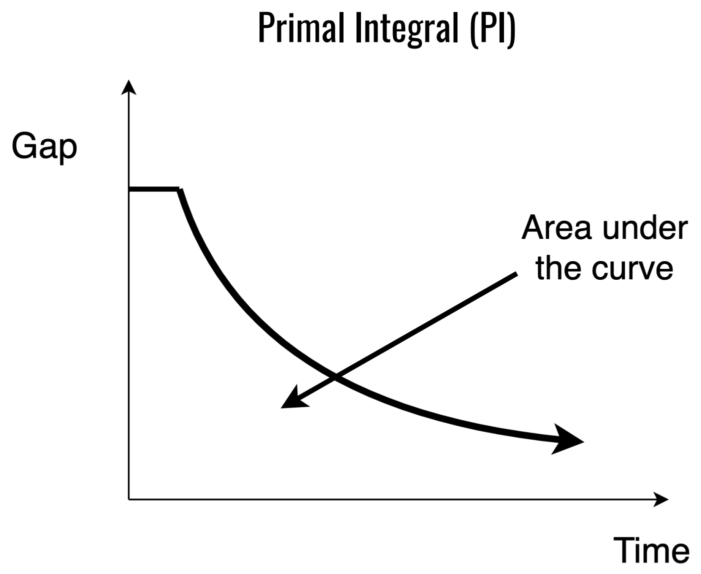

Go to next node in the B&B tree
Add to solver
No
How many constraints?
Should apply constraints?
Yes
Constraints
Which objects to prioritize?
Scheduler
Current state
Predictions
Inputs

Testing

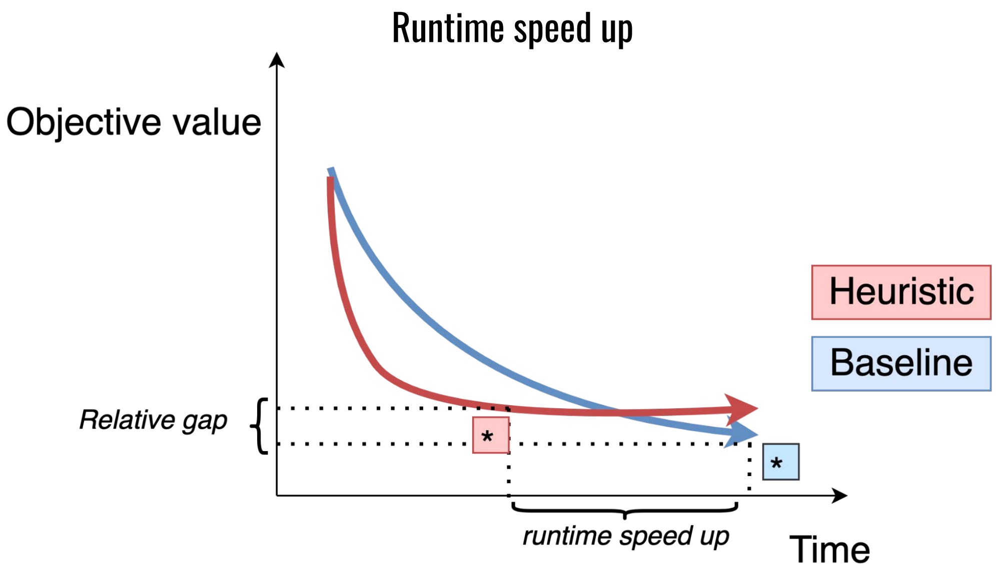


Database
Testing instance
MIP
Solver
1
2
3
Performance metrics
(gap + speed up)
4
B&B
10
4
9
Offline
Online
User cuts
8
5
Trained model
Predictions
Constraints
6
7
Tuned
hyperparameters
Policy

Metrics



Primal Integral (PI)
Gap
Area under
the curve
Time



Runtime speed up
Objective value
Heuristic
Baseline
Relative gap
*
*
runtime speed up
Time

Metrics

01	Primal integral ratio (PIR)	It takes into account the whole solution process by computing the integral of the primal gap over time.
02	**Runtime speed up**	Relative solution time for the best solution found within the time limit.
03	**Relative gap**	Gap relative to the best solution found by CPLEX.

Experimental results:

What speed up can we expect?

Experimental results

1 **Preliminary results**

Optimality gaps and accuracy

2 **Scatter plots**

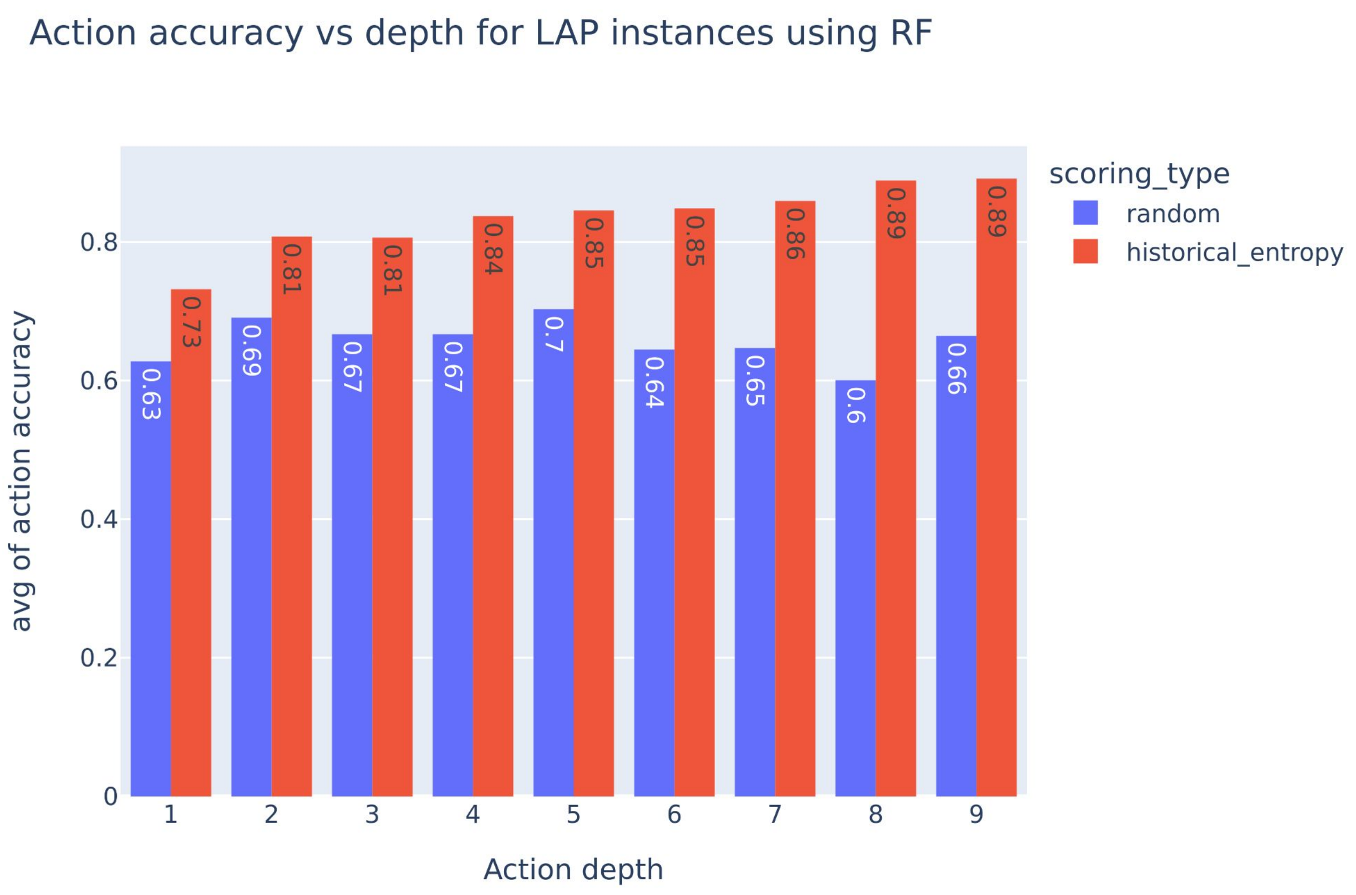
Relative gap vs speed up

3 **Summary results**

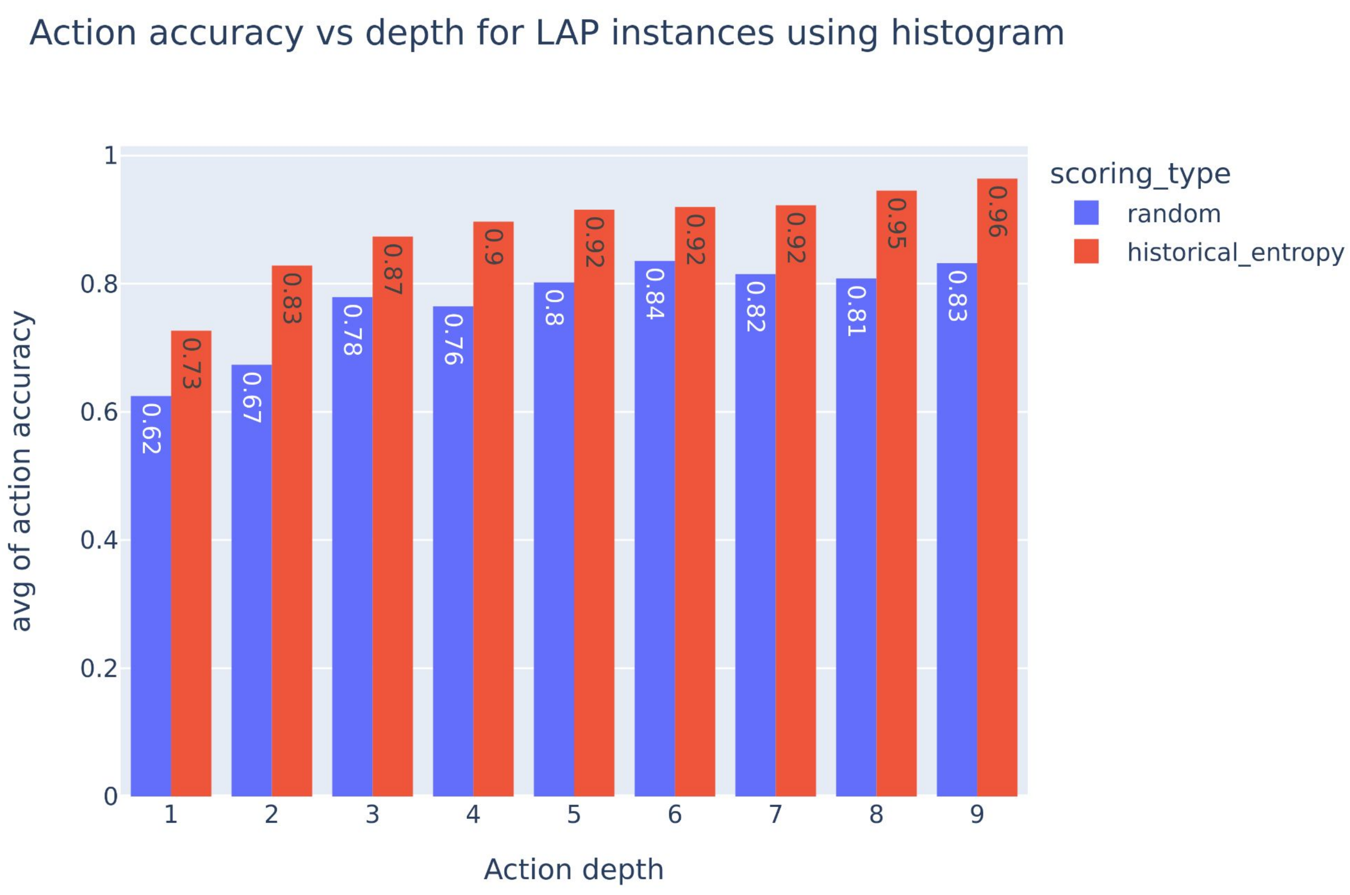
Quantiles and averages

Table 1: Optimility gap (%) distributions using CPLEX

Problem family	Quantiles			Mean
	0.05	0.5	0.95	
LAPEasy	0.00480	0.00946	0.00999	0.00847
LAPHard	0.00991	0.12673	0.81034	0.26699
FCN	0.00273	0.76846	3.31074	1.10769



Action accuracy vs depth for LAP instances using RF
avg of action accuracy
Action depth
scoring_type
random
historical_entropy
0
0.2
0.4
0.6
0.8
1
2
3
4
5
6
7
8
9
0.63
0.73
0.69
0.81
0.67
0.81
0.67
0.84
0.7
0.85
0.64
0.85
0.65
0.86
0.6
0.89
0.66
0.89



Action accuracy vs depth for LAP instances using histogram
avg of action accuracy
Action depth
scoring_type
random
historical_entropy
0
0.2
0.4
0.6
0.8
1
1
2
3
4
5
6
7
8
9
0.62
0.73
0.67
0.83
0.78
0.87
0.76
0.9
0.8
0.92
0.84
0.92
0.82
0.92
0.81
0.95
0.83
0.96

Scatter plots

Speed up vs relative gap

Oracle

What?

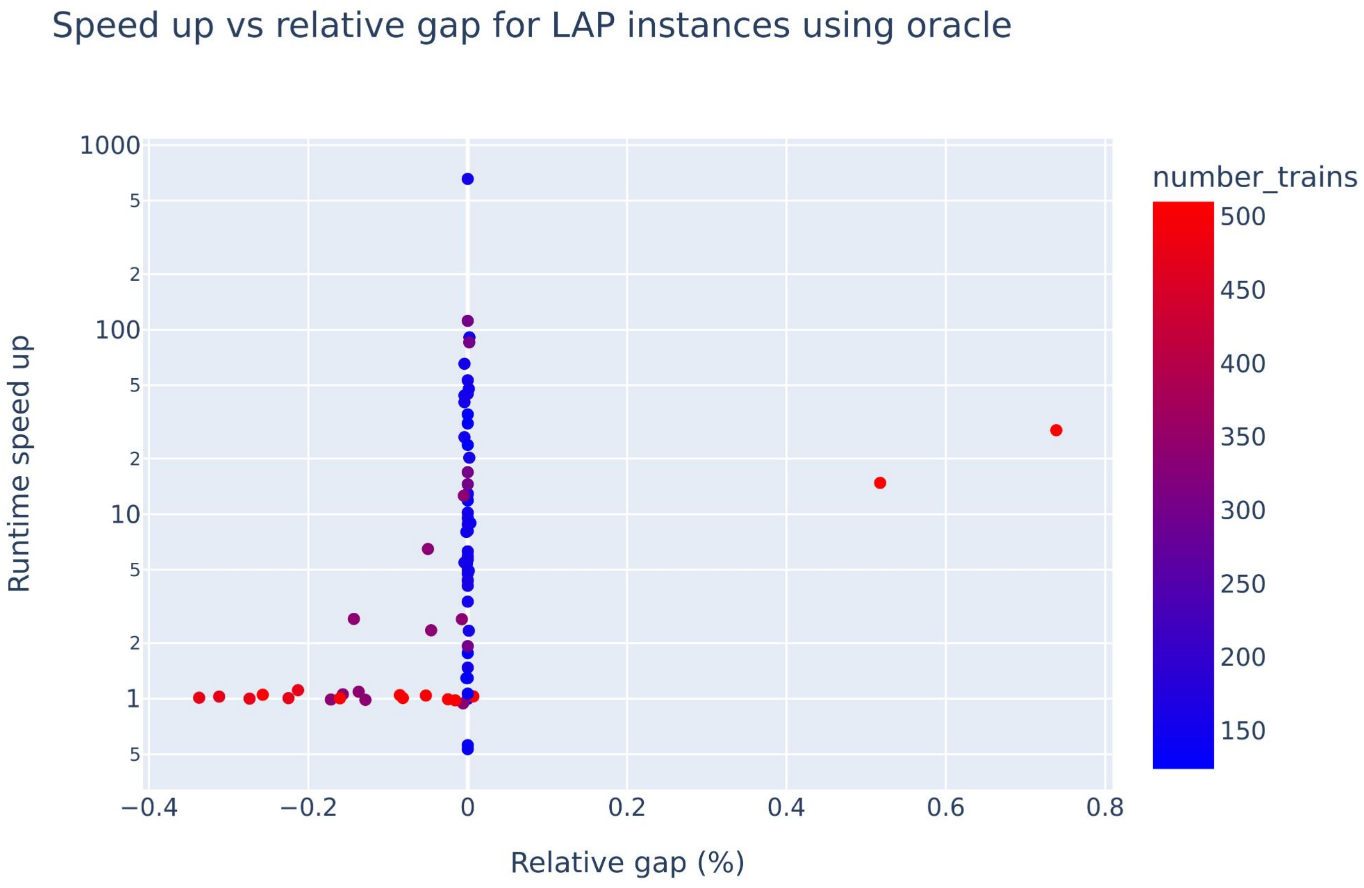
Agent that knows the best known solution.

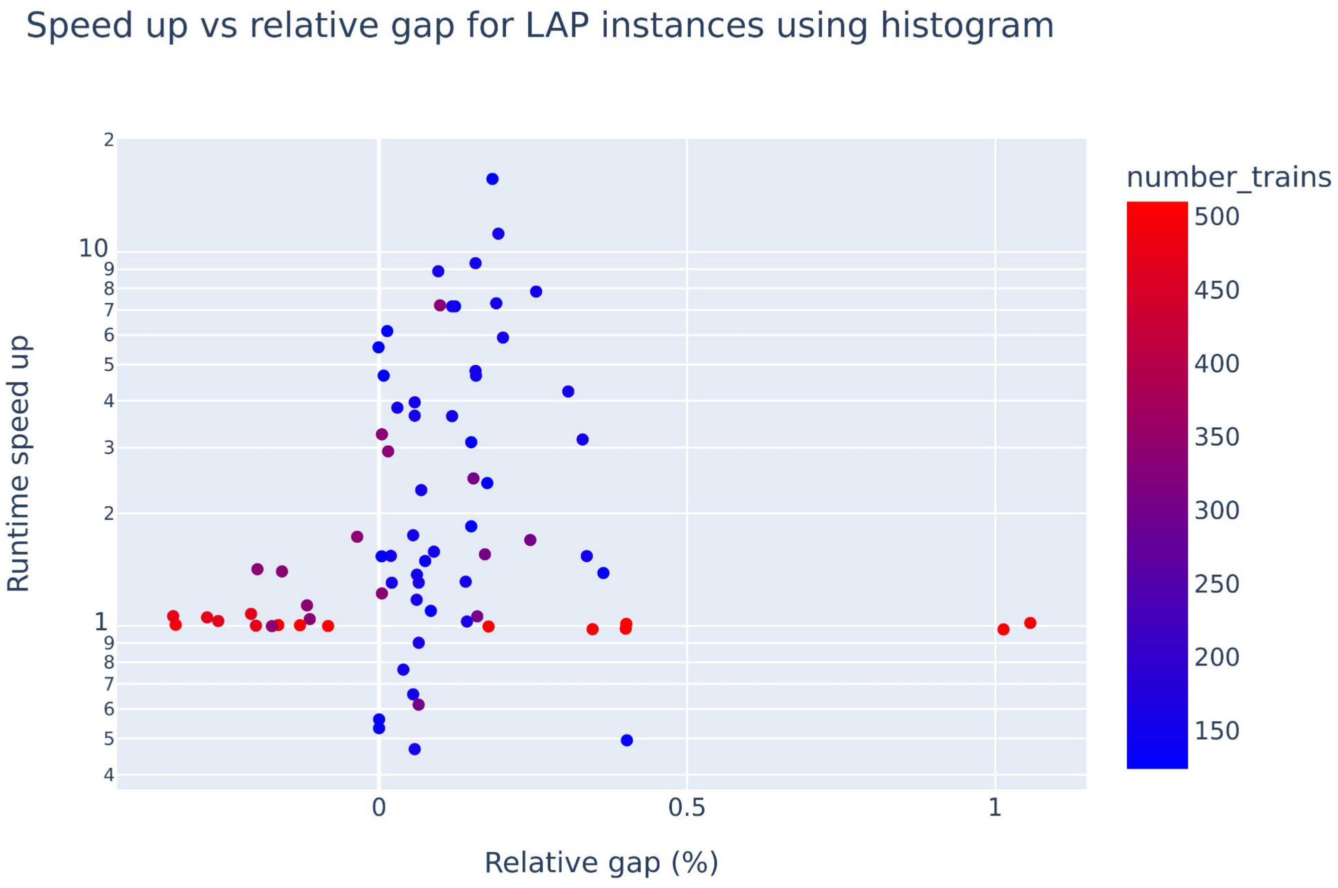
How?

Collects the best solution from baseline and applies constraints accordingly.

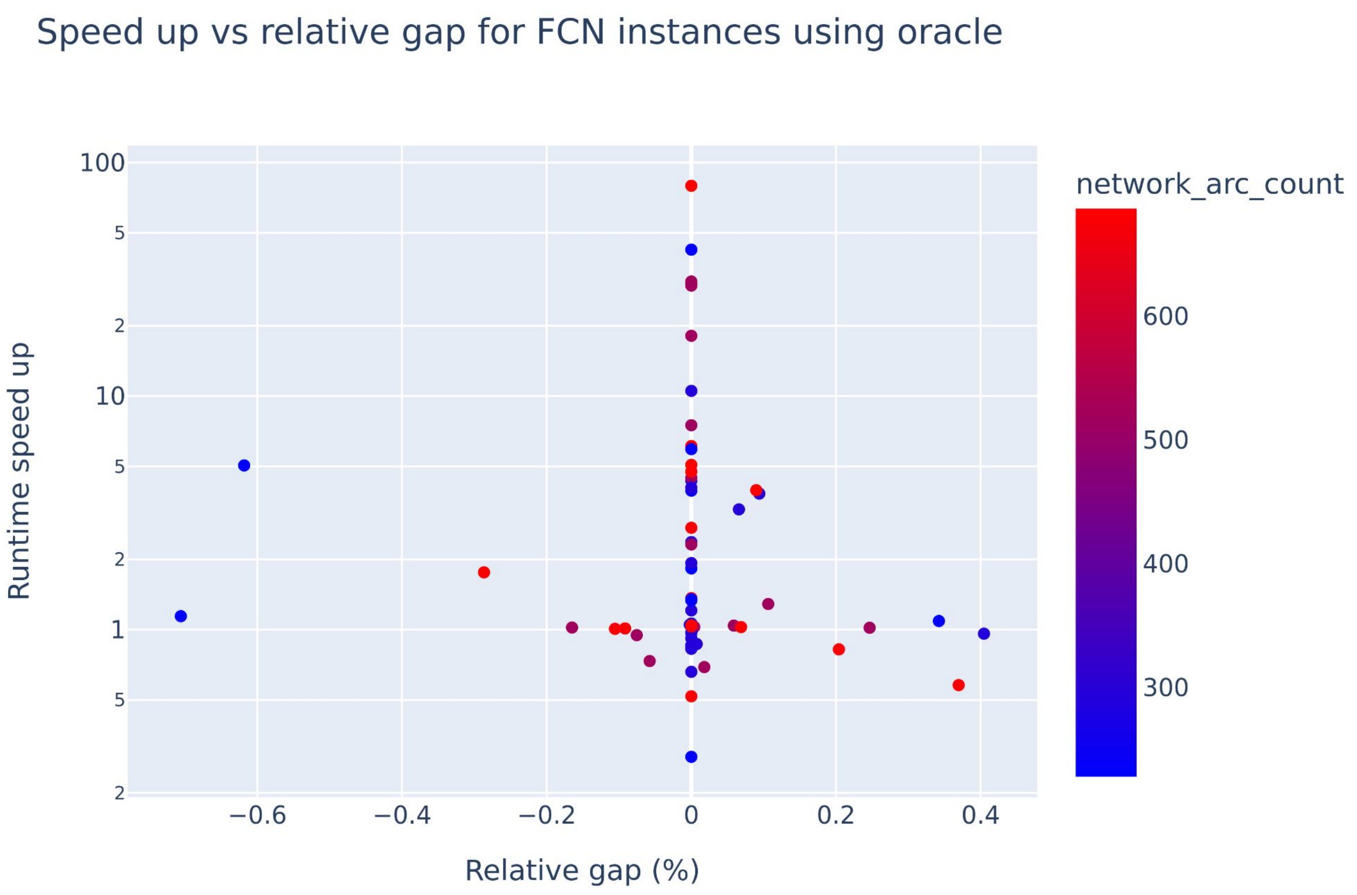
Why?

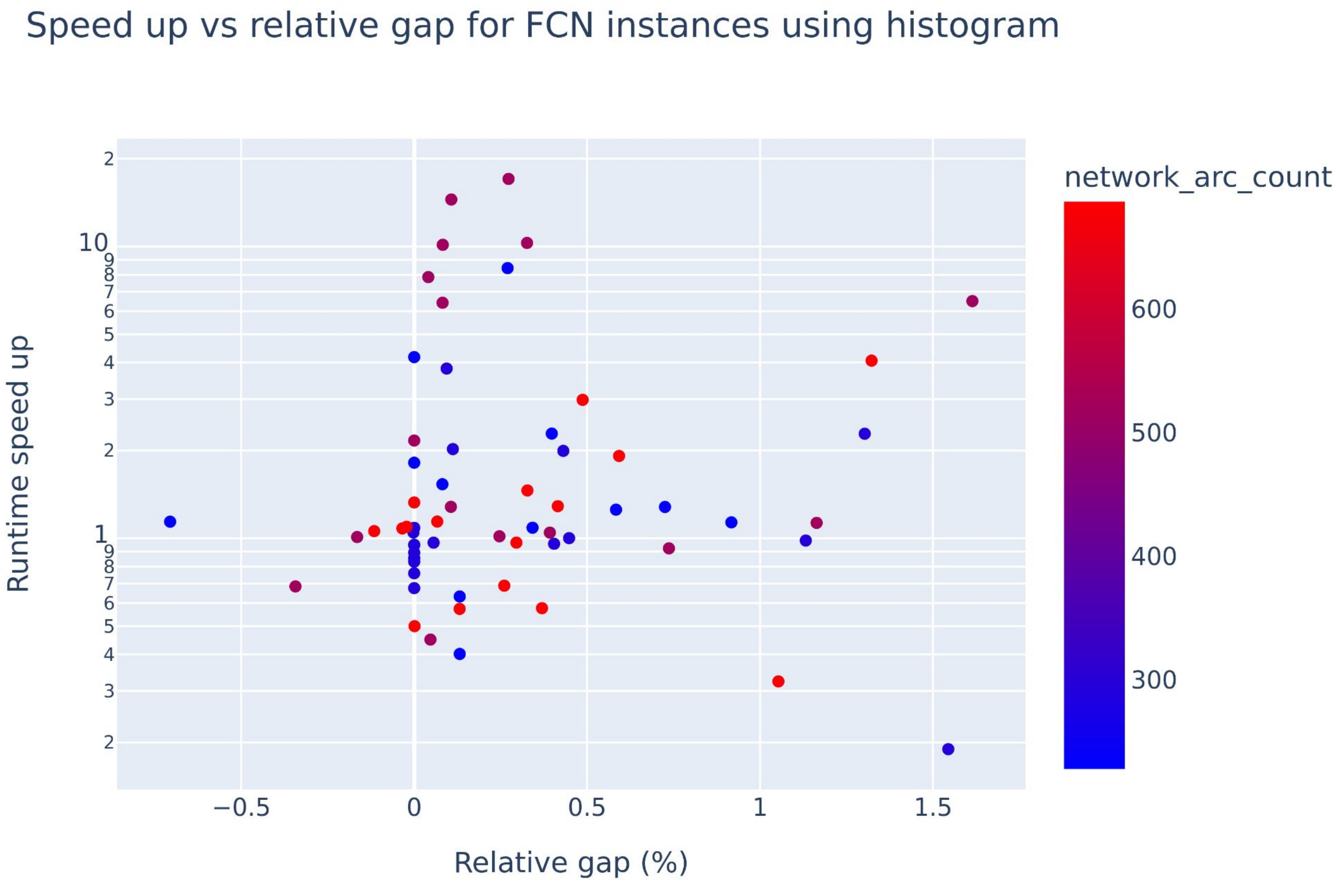
It gives an estimation of the best case scenario.


Speed up vs relative gap for LAP instances using oracle
Runtime speed up
Relative gap (%)
number_trains
1000
5
2
100
5
2
10
5
2
1
5
−0.4
−0.2
0
0.2
0.4
0.6
0.8
500
450
400
350
300
250
200
150



Speed up vs relative gap for LAP instances using histogram
Runtime speed up
Relative gap (%)
number_trains
500
450
400
350
300
250
200
150
2
10
9
8
7
6
5
4
3
2
1
9
8
7
6
5
4
0
0.5
1


Speed up vs relative gap for FCN instances using oracle
Runtime speed up
Relative gap (%)
network_arc_count
100
5
2
10
5
2
1
5
2
−0.6
−0.4
−0.2
0
0.2
0.4
600
500
400
300



Speed up vs relative gap for FCN instances using histogram
Runtime speed up
Relative gap (%)
network_arc_count
600
500
400
300
−0.5
0
0.5
1
1.5
10
1
2
3
4
5
6
7
8
9

Summary results

Quantiles and averages

Table 2: Runtime speed up distributions for LAPEasy

Learner	Pruning	Quantiles			Mean
		0.05	0.5	0.95	
oracle	High	3.76	25.22	160.76	48.51
oracle	Low	0.87	2.66	22.83	5.81
RF	High	0.63	3.60	24.04	7.26
RF	Low	0.52	1.27	4.06	1.85
histogram	High	0.63	4.16	21.42	7.31
histogram	Low	0.51	1.23	3.99	1.83

Table 3: Relative gap (%) distributions for LAPEasy

Learner	Pruning	Quantiles			Mean
		0.05	0.5	0.95	
oracle	High	-0.00044	0.00000	0.00000	0.00005
oracle	Low	-0.00240	0.00000	0.00094	-0.00023
RF	High	0.06186	0.39592	0.96963	0.46510
RF	Low	-0.00216	0.01308	0.15983	0.04116
histogram	High	0.06186	0.37970	0.96963	0.43463
histogram	Low	-0.00216	0.01308	0.15983	0.04116

Table 4: Runtime speed up distributions for LAPHard

Learner	Pruning	Quantiles			Mean
		0.05	0.5	0.95	
oracle	High	1.16	35.94	304.46	96.61
oracle	Low	0.99	2.52	35.17	9.27
histogram	High	1.00	1.54	20.58	5.88
histogram	Low	1.02	1.15	4.75	1.88

Table 5: Relative gap (%) distributions for LAPHard

Learner	Pruning	Quantiles			Mean
		0.05	0.5	0.95	
oracle	High	-0.24041	0.00000	0.00000	-0.06477
oracle	Low	-0.24178	0.00000	0.24294	-0.01079
histogram	High	-0.12548	0.29598	0.42804	0.21083
histogram	Low	-0.29987	0.05789	0.33848	0.02245

Table 6: Runtime speed up distributions for FCN

Learner	Pruning	Quantiles			Mean
		0.05	0.5	0.95	
oracle	Low	0.55	1.09	6.81	2.87
oracle	High	0.78	2.31	36.65	9.40
histogram	Low	0.47	1.02	7.15	1.98
histogram	High	0.45	1.28	9.36	2.89

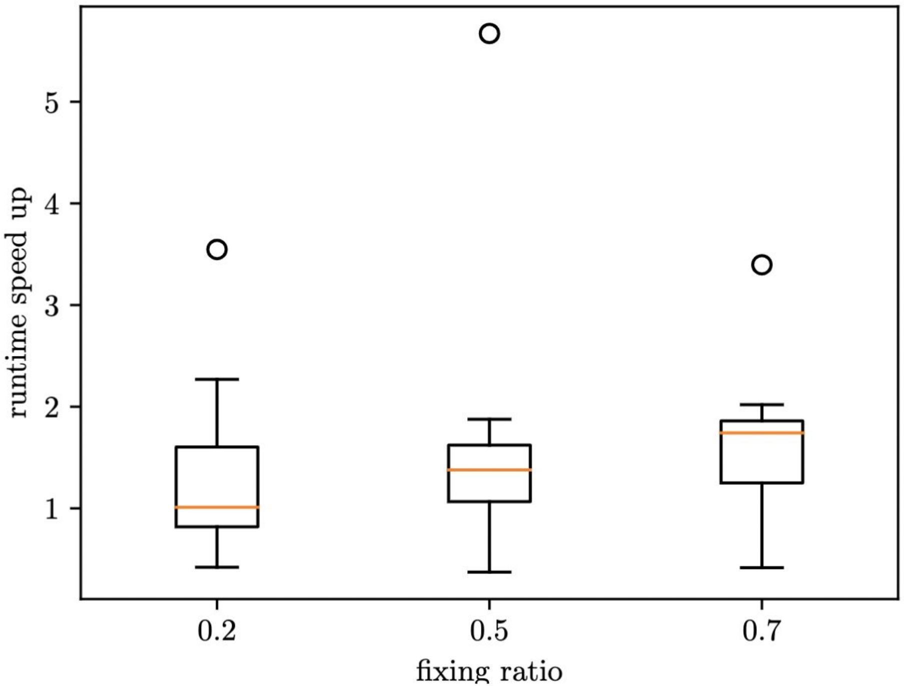
Table 7: Relative gap (%) distributions for FCN

Learner	Pruning	Quantiles			Mean
		0.05	0.5	0.95	
oracle	Low	-0.22575	0.00000	0.35587	0.01367
oracle	High	-0.09047	-0.00000	0.06723	-0.01458
histogram	Low	-0.25409	0.00109	0.38701	0.05700
histogram	High	-0.01091	0.39780	1.43325	0.53195

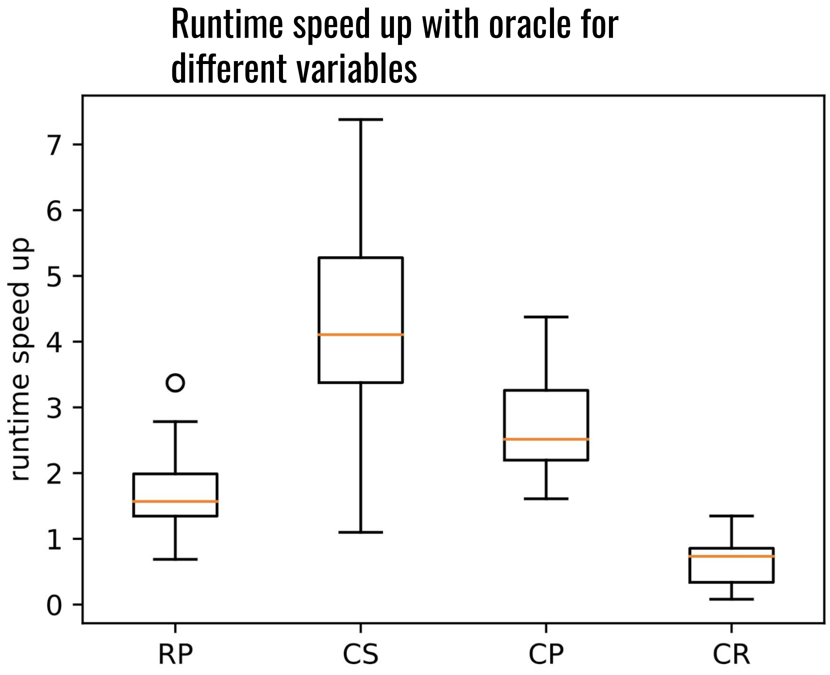
Conclusion: There is no free lunch

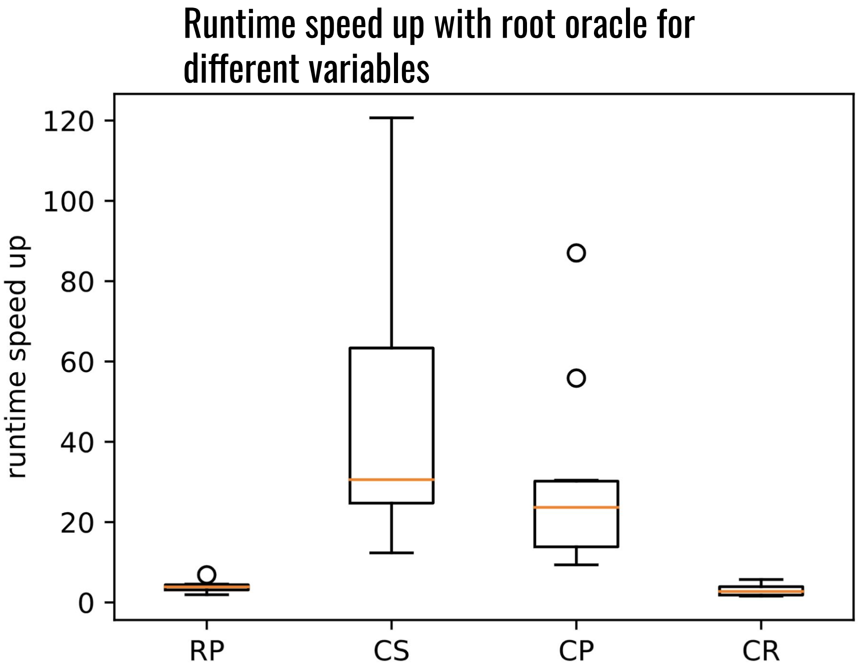
Oracle on LPP

Runtime speed up with oracle for different fixing ratios



Oracle on the LPP





RP: railcar - pattern, CS: container - slot, CP: container - platform, CR: container - railcar

Conclusion: there is no free lunch 🥙

- Demonstrated the importance of a reliable ML baseline; we can often do more with less.
- Significant speed up with marginal quality loss (<1%). With high pruning: 5 to 7x speed up on LAP, 2 to 3x on FCN.
- Revealed that the potential speed up is not as interesting on a non-graph based problem (LPP).
- Paths forwards:
    - More features and data to meaningfully outperform the ML baseline
    - Restarting strategy: pruning the problem before presolve