Exact Combinatorial Optimization with Graph Convolutional Neural Networks

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, Andrea Lodi

Neural Information Processing Systems
33rd Annual Conference, Vancouver, Dec. 8-14 2019

POLYTECHNIQUE MONTRÉAL
GERAD GROUPE D'ÉTUDES ET DE RECHERCHE EN ANALYSE DES DÉCISIONS
DATA SCIENCE FOR REAL-TIME DECISION-MAKING
Mila

Overview

The Branching Problem

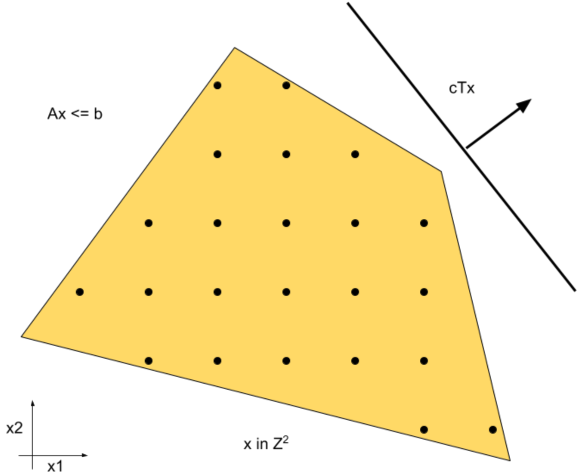
Mixed-Integer Linear Program (MILP)

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \end{aligned}$$

- $\mathbf{c} \in \mathbb{R}^n$ the objective coefficients
- $\mathbf{A} \in \mathbb{R}^{m \times n}$ the constraint coefficient matrix
- $\mathbf{b} \in \mathbb{R}^m$ the constraint right-hand-sides
- $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ the lower and upper variable bounds
- $p \leq n$ integer variables

NP-hard problem.

Mixed-Integer Linear Program (MILP)



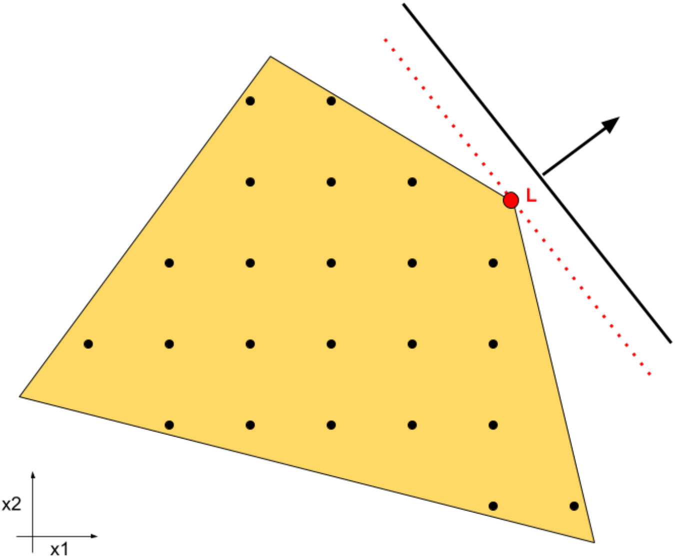
Linear Program (LP) relaxation

$$
\begin{aligned}
\arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\
\text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\
& \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\
& \mathbf{x} \in \mathbb{R}^n.
\end{aligned}
$$

Convex problem, efficient algorithms (e.g., simplex).

- $\mathbf{x}^\star \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$ (lucky) $\rightarrow$ solution to the original MILP
- $\mathbf{x}^\star \notin \mathbb{Z}^p \times \mathbb{R}^{n-p}$ $\rightarrow$ lower bound to the original MILP

Linear Program (LP) relaxation



Branch-and-Bound

Split the LP recursively over a non-integral variable, i.e. $\exists i \leq p \mid x_i^\star \notin \mathbb{Z}$

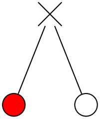
$$x_i \leq \lfloor x_i^\star \rfloor \quad \vee \quad x_i \geq \lceil x_i^\star \rceil .$$

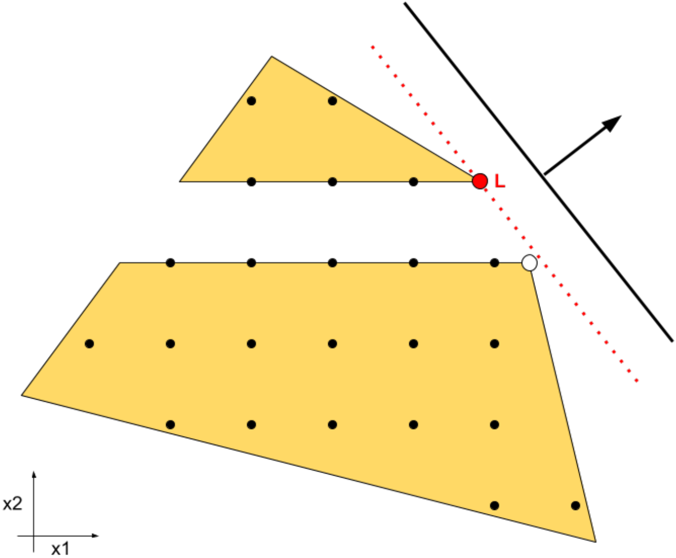
Lower bound (L): minimal among leaf nodes.
Upper bound (U): minimal among leaf nodes with integral solution.

Stopping criterion:

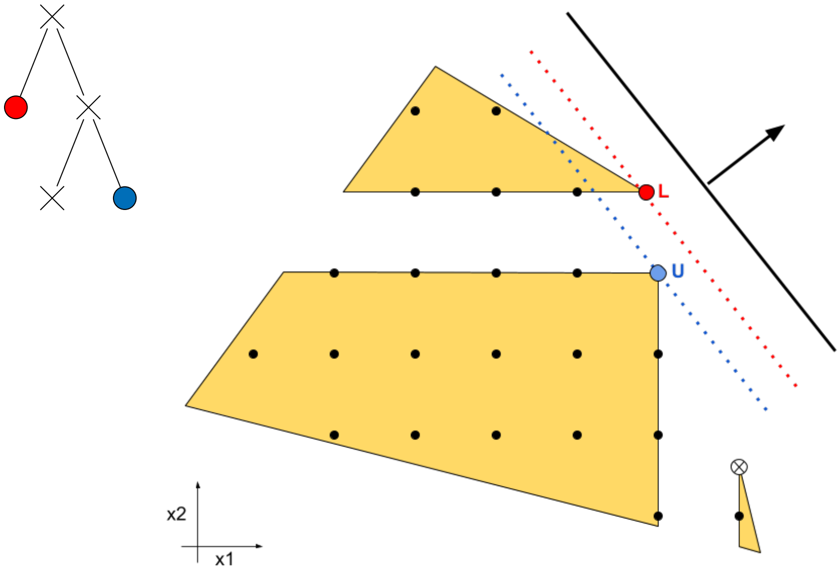
- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (early stopping)

Branch-and-Bound

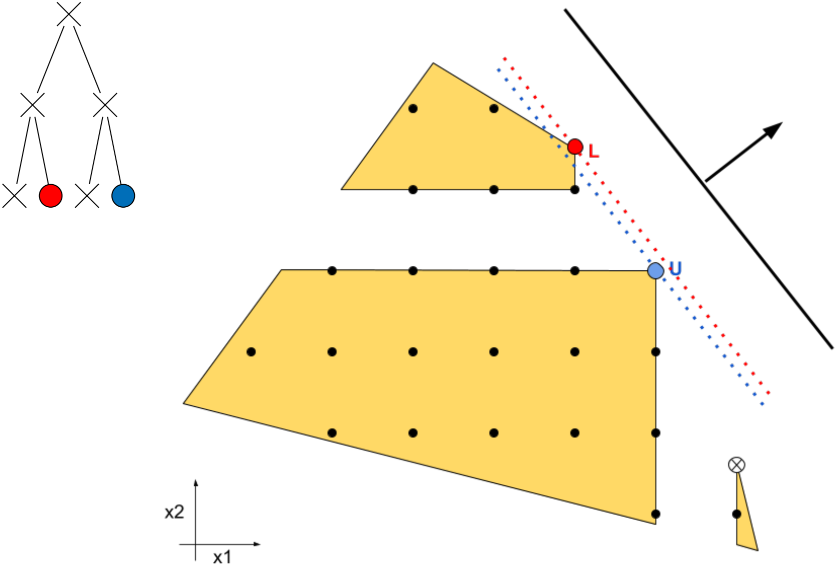




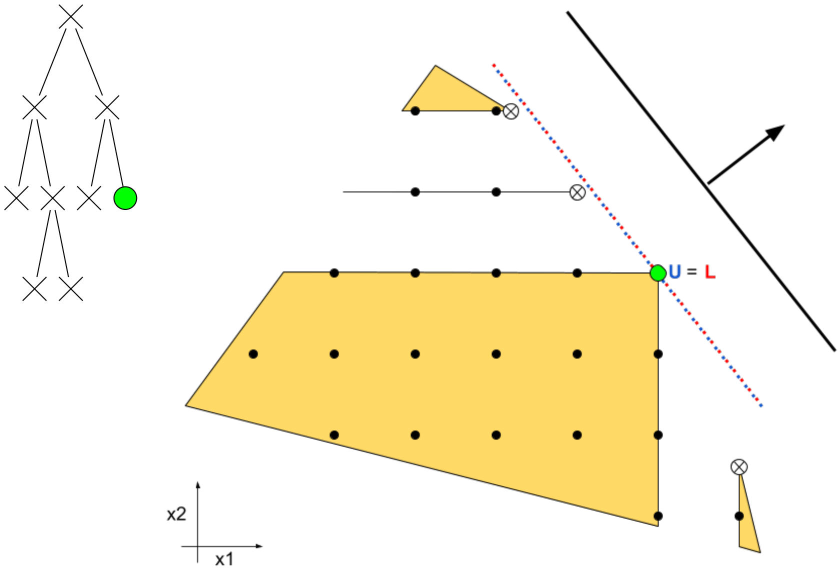
Branch-and-Bound



Branch-and-Bound



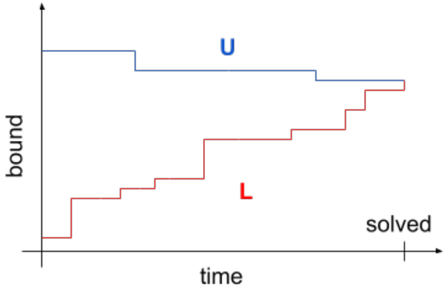
Branch-and-Bound



Branch-and-bound: a sequential process

Sequential decisions:

- variable selection (branching)
- node selection
- *cutting plane selection*
- *primal heuristic selection*
- *simplex initialization*
- ...

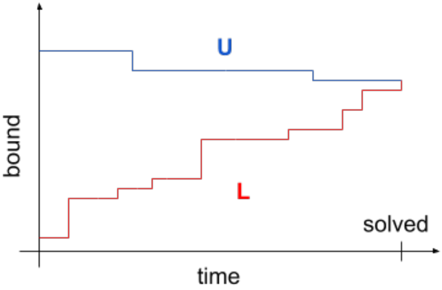


State-of-the-art in B&B solvers: expert rules

Branch-and-bound: a sequential process

Sequential decisions:

- variable selection (branching)
- node selection
- *cutting plane selection*
- *primal heuristic selection*
- *simplex initialization*
- ...



State-of-the-art in B&B solvers: expert rules

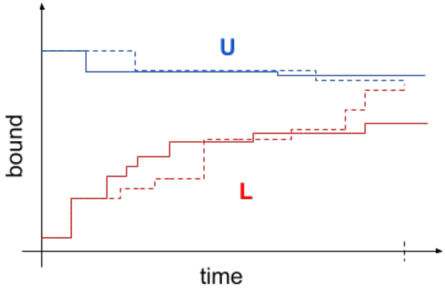
Objective: no clear consensus

- L = U fast ?
- U - L ↘ fast ?
- L ↗ fast ?
- U ↘ fast ?

Branch-and-bound: a sequential process

Sequential decisions:

- variable selection (branching)
- node selection
- *cutting plane selection*
- *primal heuristic selection*
- *simplex initialization*
- ...



Objective: no clear consensus

- L = U fast ?
- U - L ↘ fast ?
- L ↗ fast ?
- U ↘ fast ?

State-of-the-art in B&B solvers: expert rules

Expert branching rules: state-of-the-art

Strong branching: one-step forward looking (greedy)

- solve both LPs for each candidate variable
- pick the variable resulting in tightest relaxation
- \+ small trees
- − computationally expensive

Pseudo-cost: backward looking

- keep track of tightenings in past branchings
- pick the most promising variable
- \+ very fast, almost no computations
- − cold start

Reliability pseudo-cost: best of both worlds

- compute SB scores at the beginning
- gradually switches to pseudo-cost (+ other heuristics)
- \+ best overall solving time trade-off (on MIPLIB)

Markov Decision Process

Objective: take actions which maximize the long-term reward

$$\sum_{t=0}^{\infty} r(\mathbf{s}_t),$$

with $r : \mathcal{S} \to \mathbb{R}$ a reward function.

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.

Action: a branching variable, $a \in \{1, \ldots, p\}$.

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.
Action: a branching variable, $a \in \{1, \ldots, p\}$.

Trajectory: $\tau = (\mathbf{s}_0, \ldots, \mathbf{s}_T)$

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.
Action: a branching variable, $a \in \{1, \ldots, p\}$.

Trajectory: $\tau = (\mathbf{s}_0, \ldots, \mathbf{s}_T)$

- initial state $\mathbf{s}_0$: a MILP $\sim p(\mathbf{s}_0)$;

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.
Action: a branching variable, $a \in \{1, \ldots, p\}$.

Trajectory: $\tau = (\mathbf{s}_0, \ldots, \mathbf{s}_T)$

- initial state $\mathbf{s}_0$: a MILP $\sim p(\mathbf{s}_0)$;
- terminal state $\mathbf{s}_T$: the MILP is solved;

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.
Action: a branching variable, $a \in \{1, \ldots, p\}$.

Trajectory: $\tau = (\mathbf{s}_0, \ldots, \mathbf{s}_T)$

- initial state $\mathbf{s}_0$: a MILP $\sim p(\mathbf{s}_0)$;
- terminal state $\mathbf{s}_T$: the MILP is solved;
- intermediate states: branching

$$\mathbf{s}_{t+1} \sim p_\pi(\mathbf{s}_{t+1}|\mathbf{s}_t) = \sum_{a \in \mathcal{A}} \underbrace{\pi(a|\mathbf{s}_t)}_{\text{branching policy}} \underbrace{p(\mathbf{s}_{t+1}|\mathbf{s}_t, a)}_{\text{solver internals}}.$$

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.
Action: a branching variable, $a \in \{1, \ldots, p\}$.

Trajectory: $\tau = (\mathbf{s}_0, \ldots, \mathbf{s}_T)$

- initial state $\mathbf{s}_0$: a MILP $\sim p(\mathbf{s}_0)$;
- terminal state $\mathbf{s}_T$: the MILP is solved;
- intermediate states: branching

$$\mathbf{s}_{t+1} \sim p_\pi(\mathbf{s}_{t+1}|\mathbf{s}_t) = \sum_{a \in \mathcal{A}} \underbrace{\pi(a|\mathbf{s}_t)}_{\text{branching policy}} \underbrace{p(\mathbf{s}_{t+1}|\mathbf{s}_t, a)}_{\text{solver internals}}.$$

Branching problem: solve

$$\pi^\star = \arg\max_\pi \mathop{\mathbb{E}}_{\tau \sim p_\pi} [r(\tau)],$$

with $r(\tau) = \sum_{\mathbf{s} \in \tau} r(\mathbf{s})$.

Branching as a Markov Decision Process

State: the whole internal state of the solver, $\mathbf{s}$.
Action: a branching variable, $a \in \{1, \ldots, p\}$.

Trajectory: $\tau = (\mathbf{s}_0, \ldots, \mathbf{s}_T)$

- initial state $\mathbf{s}_0$: a MILP $\sim p(\mathbf{s}_0)$;
- terminal state $\mathbf{s}_T$: the MILP is solved;
- intermediate states: branching

$$\mathbf{s}_{t+1} \sim p_\pi(\mathbf{s}_{t+1}|\mathbf{s}_t) = \sum_{a \in \mathcal{A}} \underbrace{\pi(a|\mathbf{s}_t)}_{\text{branching policy}} \underbrace{p(\mathbf{s}_{t+1}|\mathbf{s}_t, a)}_{\text{solver internals}}.$$

Branching problem: solve

$$\pi^\star = \arg\max_\pi \mathop{\mathbb{E}}_{\tau \sim p_\pi} [r(\tau)],$$

with $r(\tau) = \sum_{\mathbf{s} \in \tau} r(\mathbf{s})$.

A policy $\pi^\star$ may not be optimal in two distinct configurations.

Challenges

MDP $\implies$ Reinforcement learning (RL) ?

Challenges

MDP $\Longrightarrow$ Reinforcement learning (RL) ?

State representation: $\mathbf{s}$

- global level: original MILP, tree, bounds, focused node...
- node level: variable bounds, LP solution, simplex statistics...

Challenges

MDP $\implies$ Reinforcement learning (RL) ?

State representation: $\mathbf{s}$

- global level: original MILP, tree, bounds, focused node...
- node level: variable bounds, LP solution, simplex statistics...
- – dynamically growing structure (tree)
- – variable-size instances (cols, rows) $\implies$ Graph Neural Network

Challenges

MDP $\Longrightarrow$ Reinforcement learning (RL) ?

State representation: $\mathbf{s}$

- global level: original MILP, tree, bounds, focused node...
- node level: variable bounds, LP solution, simplex statistics...
- dynamically growing structure (tree)
- variable-size instances (cols, rows) $\Longrightarrow$ Graph Neural Network

Sampling trajectories: $\tau \sim p_\pi$

- collect one τ = solving a MILP (with π likely not optimal)
- expensive $\Longrightarrow$ train on small instances

Challenges

MDP $\implies$ Reinforcement learning (RL) ?

State representation: **s**

- global level: original MILP, tree, bounds, focused node...
- node level: variable bounds, LP solution, simplex statistics...
- – dynamically growing structure (tree)
- – variable-size instances (cols, rows) $\implies$ Graph Neural Network

Sampling trajectories: $\tau \sim p_\pi$

- collect one τ = solving a MILP (with π likely not optimal)
- – expensive $\implies$ train on small instances

Reward function: r

- no consensus
- \+ a strong expert exists $\implies$ imitation learning

Machine learning approaches

Node selection

- He et al., 2014
- Song et al., 2018

Variable selection (branching)

- Khalil, Le Bodic, et al., 2016 $\Longrightarrow$ "online" imitation learning
- Hansknecht et al., 2018 $\Longrightarrow$ offline imitation learning
- Balcan et al., 2018 $\Longrightarrow$ theoretical results

Cut selection

- Baltean-Lugojan et al., 2018
- Tang et al., 2019

Primal heuristic selection

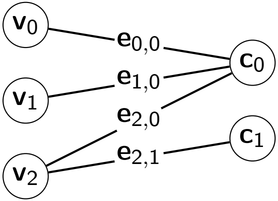
- Khalil, Dilkina, et al., 2017
- Hendel et al., 2018

The Graph Convolution Neural Network Model

Node state encoding

Natural representation : variable / constraint bipartite graph

$$
\begin{aligned}
\arg\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\
\text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\
& \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \\
& \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}.
\end{aligned}
$$

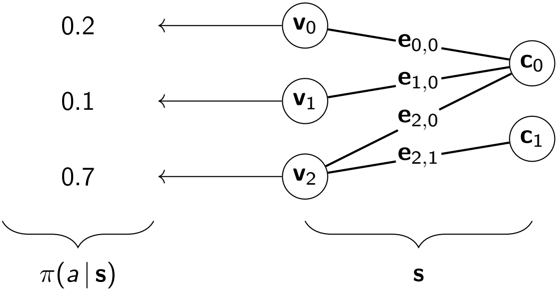


- $\mathbf{v}_i$: variable features (type, coef., bounds, LP solution...)
- $\mathbf{c}_j$: constraint features (right-hand-side, LP slack...)
- $\mathbf{e}_{i,j}$: non-zero coefficients in $\mathbf{A}$

D. Selsam et al. (2019). Learning a SAT Solver from Single-Bit Supervision.

Branching Policy as a GCNN Model

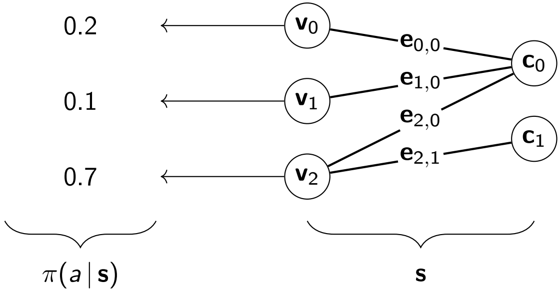
Neighbourhood-based updates: $\mathbf{v}_i \leftarrow \sum_{j \in \mathcal{N}_i} \mathbf{f}_\theta(\mathbf{v}_i, \mathbf{e}_{i,j}, \mathbf{c}_j)$



T. N. Kipf et al. (2016). Semi-Supervised Classification with Graph Convolutional Networks.

Branching Policy as a GCNN Model

Neighbourhood-based updates: $\mathbf{v}_i \leftarrow \sum_{j \in \mathcal{N}_i} \mathbf{f}_\theta(\mathbf{v}_i, \mathbf{e}_{i,j}, \mathbf{c}_j)$



Natural model choice for graph-structured data

- permutation-invariance
- benefits from sparsity

T. N. Kipf et al. (2016). Semi-Supervised Classification with Graph Convolutional Networks.

Experiments

Imitation learning

Full Strong Branching (FSB): good branching rule, but expensive.
Can we learn a fast, good-enough approximation ?

[1] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 6.

Imitation learning

Full Strong Branching (FSB): good branching rule, but expensive.
Can we learn a fast, good-enough approximation ?

Behavioural cloning

- collect $\mathcal{D} = \{(\mathbf{s}, a^\star), \dots\}$ from the expert agent (FSB)
- estimate $\pi^\star(a \mid \mathbf{s})$ from $\mathcal{D}$
- \+ no reward function, supervised learning, well-behaved
- − will never surpass the expert...

Implementation with the open-source solver SCIP[1]

[1] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 6.

Imitation learning

Full Strong Branching (FSB): good branching rule, but expensive.
Can we learn a fast, good-enough approximation ?

Behavioural cloning

- collect $\mathcal{D} = \{(\mathbf{s}, a^\star), \dots\}$ from the expert agent (FSB)
- estimate $\pi^\star(a \,|\, \mathbf{s})$ from $\mathcal{D}$
- \+ no reward function, supervised learning, well-behaved
- − will never surpass the expert...

Implementation with the open-source solver SCIP[1]

Not a new idea

- Alvarez et al., 2017 predict SB scores, XTrees model
- Khalil, Le Bodic, et al., 2016 predict SB rankings, SVMrank model
- Hansknecht et al., 2018 do the same, λ-MART model

[1] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 6.

Minimum set covering[2]

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	17.30	0 / 100	17	411.34	0 / 90	171	3600.00	0 / 0	n/a
RPB	8.98	0 / 100	**54**	60.07	0 / 100	1741	1677.02	4 / 65	47 299
XTrees	9.28	0 / 100	187	92.47	0 / 100	2187	2869.21	0 / 35	59 013
SVMrank	8.10	1 / 100	165	73.58	0 / 100	1915	2389.92	0 / 47	42 120
λ-MART	7.19	14 / 100	167	59.98	0 / 100	1925	2165.96	0 / 54	45 319
GCNN	**6.59**	**85** / 100	134	**42.48**	**100** / 100	**1450**	**1489.91**	**66** / 70	**29 981**

3 problem sizes

- 500 rows, 1000 cols (easy), training distribution
- 1000 rows, 1000 cols (medium)
- 2000 rows, 1000 cols (hard)

[2] E. Balas et al. (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study.

Minimum set covering[2]

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	17.30	0 / 100	17	411.34	0 / 90	171	3600.00	0 / 0	n/a
RPB	8.98	0 / 100	**54**	60.07	0 / 100	1741	1677.02	4 / 65	47 299
XTrees	9.28	0 / 100	187	92.47	0 / 100	2187	2869.21	0 / 35	59 013
SVMrank	8.10	1 / 100	165	73.58	0 / 100	1915	2389.92	0 / 47	42 120
λ-MART	7.19	14 / 100	167	59.98	0 / 100	1925	2165.96	0 / 54	45 319
GCNN	**6.59**	**85** / 100	134	**42.48**	**100** / 100	**1450**	**1489.91**	**66** / 70	**29 981**

3 problem sizes

- 500 rows, 1000 cols (easy), training distribution
- 1000 rows, 1000 cols (medium)
- 2000 rows, 1000 cols (hard)

Pays off: better than SCIP's default in terms of solving time.

[2] E. Balas et al. (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study.

Minimum set covering[2]

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	17.30	0 / 100	17	411.34	0 / 90	171	3600.00	0 / 0	n/a
RPB	8.98	0 / 100	**54**	60.07	0 / 100	1741	1677.02	4 / 65	47 299
XTrees	9.28	0 / 100	187	92.47	0 / 100	2187	2869.21	0 / 35	59 013
SVMrank	8.10	1 / 100	165	73.58	0 / 100	1915	2389.92	0 / 47	42 120
λ-MART	7.19	14 / 100	167	59.98	0 / 100	1925	2165.96	0 / 54	45 319
GCNN	**6.59**	**85** / 100	134	**42.48**	**100** / 100	**1450**	**1489.91**	**66** / 70	**29 981**

3 problem sizes

- 500 rows, 1000 cols (easy), training distribution
- 1000 rows, 1000 cols (medium)
- 2000 rows, 1000 cols (hard)

Pays off: better than SCIP's default in terms of solving time.
Generalizes to harder problems !

[2] E. Balas et al. (1980). Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study.

Combinatorial auction[3]

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	4.11	0 / 100	6	86.90	0 / 100	72	1813.33	0 / 68	400
RPB	2.74	0 / 100	10	17.41	0 / 100	689	136.17	13 / 100	5511
XTrees	2.47	0 / 100	86	23.70	0 / 100	976	451.39	0 / 95	10 290
SVMrank	2.31	0 / 100	77	23.10	0 / 100	867	364.48	0 / 98	6329
λ-MART	**1.79**	**75** / 100	77	14.42	1 / 100	873	222.54	0 / 100	7006
GCNN	1.85	25 / 100	**70**	**10.29**	**99** / 100	**657**	**114.16**	**87** / 100	**5169**

3 problem sizes

- 100 items, 500 bids (easy), training distribution
- 200 items, 1000 bids (medium)
- 300 items, 1500 bids (hard)

[3] K. Leyton-Brown et al. (2000). Towards a Universal Test Suite for Combinatorial Auction Algorithms.

Capacitated facility location[4]

Model	Easy			Medium			Hard		
	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	30.36	4 / 100	14	214.25	1 / 100	76	742.91	15 / 90	55
RPB	26.55	9 / 100	**22**	156.12	8 / 100	**142**	631.50	14 / 96	**110**
XTrees	28.96	3 / 100	135	159.86	3 / 100	401	671.01	1 / 95	381
SVMrank	23.58	11 / 100	117	130.86	13 / 100	348	586.13	21 / 95	321
λ-MART	23.34	16 / 100	117	128.48	23 / 100	349	582.38	15 / 95	314
GCNN	**22.10**	**57** / 100	107	**120.94**	**52** / 100	339	**563.36**	**30** / 95	338

3 problem sizes

- 100 facilities, 100 customers (easy), training distribution
- 100 facilities, 200 customers (medium)
- 100 facilities, 400 customers (hard)

[4] G. Cornuejols et al. (1991). A comparison of heuristics and relaxations for the capacitated plant location problem.

Maximum independent set[5]

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	23.58	9 / 100	7	1503.55	0 / 74	38	3600.00	0 / 0	n/a
RPB	8.77	7 / 100	**20**	**110.99**	41 / 100	729	2045.61	22 / 42	**2675**
XTrees	10.75	1 / 100	76	1183.37	1 / 47	4664	3565.12	0 / 3	38 296
SVMrank	8.83	2 / 100	46	242.91	1 / 96	**546**	2902.94	1 / 18	6256
λ-MART	7.31	30 / 100	52	219.22	15 / 91	747	3044.94	0 / 12	8893
GCNN	**6.43**	51 / 100	43	192.91	**42** / 82	1841	**2024.37**	**25** / 29	2997

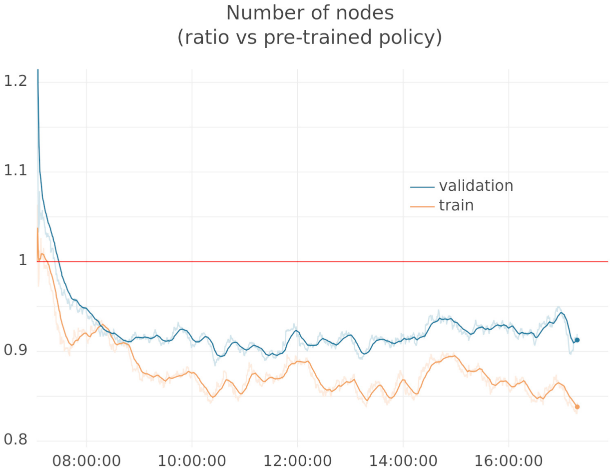
3 problem sizes, Barabási-Albert graphs (affinity=4)

- 500 nodes (easy), training distribution
- 1000 nodes (medium)
- 1500 nodes (hard)

[5] D. Chalupa et al. (2014). On the Growth of Large Independent Sets in Scale-Free Networks.

Reinforcement learning

Early results: set covering problem



Reward: negative number of nodes

Proximal Policy Optimization (PPO)

Challenging. . . but promising !

Conclusion

Heuristic vs data-driven branching:

+ tune B&B to your problem of interest automatically
– no guarantees outside of the training distribution
– requires training instances

Conclusion

Heuristic vs data-driven branching:

+ tune B&B to your problem of interest automatically
– no guarantees outside of the training distribution
– requires training instances

What next:

- real-world problems
- other solver components: node selection, cut selection...
- reinforcement learning: still a lot of challenges
- interpretation: which variables are chosen ? Why ?
- provide an clean API + benchmarks for MILP adaptive solving (based on the open-source SCIP solver)

Paper: `https://arxiv.org/abs/1906.01629` M. Gasse et al. (2019). Exact Combinatorial Optimization with Graph Convolutional Neural Networks.

Code: `https://github.com/ds4dm/learn2branch`

Exact Combinatorial Optimization with Graph Convolutional Neural Networks

Thank you!

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, Andrea Lodi

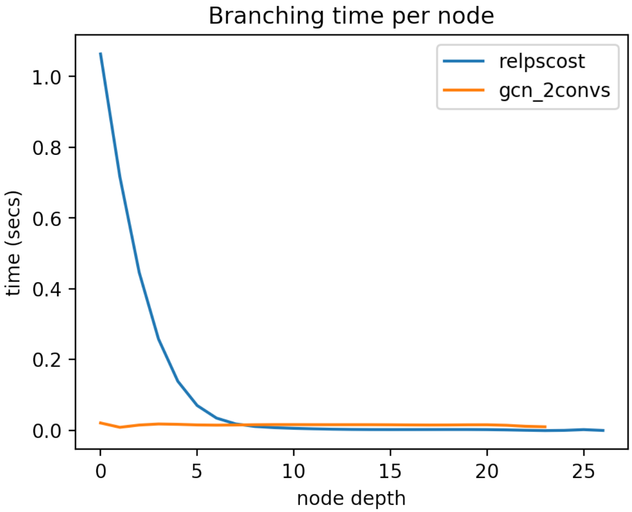
POLYTECHNIQUE MONTRÉAL

GERAD GROUPE D'ÉTUDES ET DE RECHERCHE EN ANALYSE DES DÉCISIONS

DATA SCIENCE FOR REAL-TIME DECISION-MAKING

Mila

Learned Policy vs Reliability Pseudocost (SCIP default)



Time delta:
- python overhead
- data extraction (**s**)
- model evaluation

Close the gap:
- engineering ?
- efficient heuristics (reliability) ?