Lookback for Learning to Branch

Prateek Gupta*, Elias B. Khalil, Didier Chételat, Maxime Gasse,
M. Pawan Kumar, Andrea Lodi, Yoshua Bengio

SIAM Conference on Optimization, June 2nd 2023

To **improve** the extent to which neural networks can **imitate** a computationally expensive but accurate heuristic to solve mixed-integer linear programming (MILP) problems.

Outline

Problem formulation

Our solution

Conclusion

Outline

- Problem formulation
  - Discrete Optimization
  - Branch-and-Bound
  - The Branching Problem
  - Learning to branch
  - Lookback property
- Our solution
- Conclusion

Outline

Mixed-Integer Linear Program (MILP)

$$\arg\min_{x} \quad c^\top x$$

- $c \in \mathbb{R}^n$ the objective coefficients

Mixed-Integer Linear Program (MILP)

$$\begin{aligned} \arg\min_{x} \quad & c^{\top}x \\ \text{subject to} \quad & Ax \leq b, \end{aligned}$$

- $c \in \mathbb{R}^{n}$ the objective coefficients
- $A \in \mathbb{R}^{m \times n}$ the constraint coefficient matrix
- $b \in \mathbb{R}^{m}$ the constraint right-hand-sides

Mixed-Integer Linear Program (MILP)

$$\begin{aligned} \arg\min_{\mathsf{x}} \quad & \mathsf{c}^\top \mathsf{x} \\ \text{subject to} \quad & \mathsf{A}\mathsf{x} \leq \mathsf{b}, \\ & \mathsf{l} \leq \mathsf{x} \leq \mathsf{u}, \end{aligned}$$

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

Mixed-Integer Linear Program (MILP)

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

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

Mixed-Integer Linear Program (MILP)

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

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

NP-hard problem.

Applications

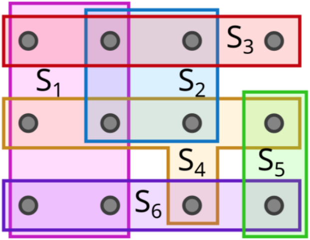
Combinatorial Auctions

Facility location-Allocation

Maximum Indendent Set

Set Covering

and many more ...



Mixed-Integer Linear Program (MILP)

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

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

NP-hard problem.

Mixed-Integer Linear Program (MILP)

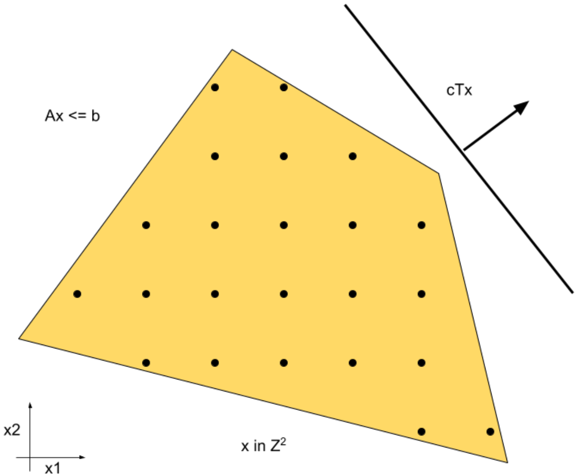


Image credit: Maxime Gasse

Linear Program (LP)

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

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

Linear Program (LP)

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

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

◦ Polynomially solvable

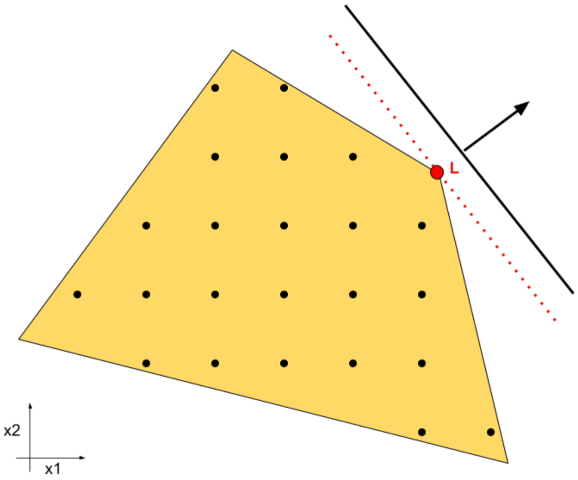
Linear Program (LP)

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

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

- Polynomially solvable
- Yields lower bounds to the original MILP

LP Relaxation of a MILP

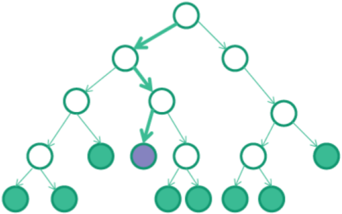


Outline

Branch-and-Bound (B&B)

B&B (Land et al., 1960) is the widely used framework to solve MILPs.

It consists of two steps



Each node in branch-and-bound is a new MIP

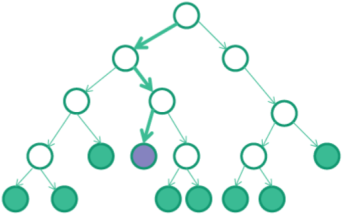
Image source: https://www.gurobi.com/resource/mip-basics/

Branch-and-Bound (B&B)

B&B (Land et al., 1960) is the widely used framework to solve MILPs.

It consists of two steps

- **Branching** - Select variable to split the problem into two



Each node in branch-and-bound is a new MIP

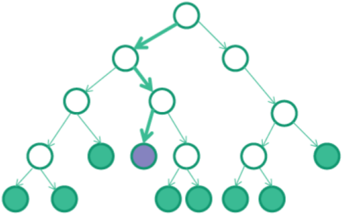
Image source: https://www.gurobi.com/resource/mip-basics/

Branch-and-Bound (B&B)

B&B (Land et al., 1960) is the widely used framework to solve MILPs.

It consists of two steps

- **Branching** - Select variable to split the problem into two
- **Bounding** - Solve the LP relaxation of resulting problem to obtain optimization guarantees on the solution



Each node in branch-and-bound is a new MIP

Image source: https://www.gurobi.com/resource/mip-basics/

Branch-and-Bound

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

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

Branch-and-Bound

Branch: 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.

Branch-and-Bound

Branch: 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: 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:

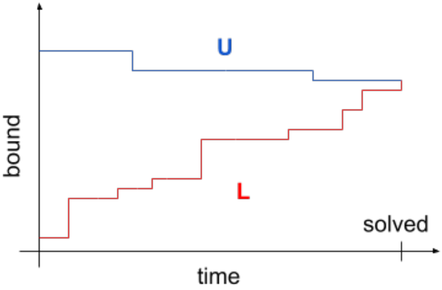
- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (early stopping)

Note: A time limit is used to ensure termination.

Branch-and-bound: a sequential process

Sequential decisions:

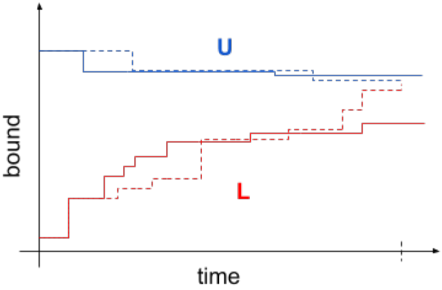
- variable selection (branching)
- node selection
- *cutting plane selection*
- *primal heuristic selection*
- *simplex initialization*
- ...



Branch-and-bound: a sequential process

Sequential decisions:

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



Outline

Branching Policy

It is also called as **variable selection policy**.

Policy Objective: Given a B&B node i.e. MILP, select a variable $i \leq p \mid x_i^* \notin \mathbb{Z}$ so that the final size of the tree is minimum (a proxy for running time).

A gold standard: Strong Branching (impractical)

Strong branching[1]: one-step forward looking (greedy)

- solve both LPs **for each candidate variable**
- select the variable resulting in tightest relaxation

\+ small trees

− computationally expensive

[1] D. Applegate et al. (1995). Finding cuts in the TSP. Tech. rep. DIMACS; J. Linderoth et al. (May 1999). A Computational Study of Search Strategies for Mixed Integer Programming.

A gold standard: Strong Branching (impractical)

Strong branching score for a variable i at a node n

A gold standard: Strong Branching (impractical)

Strong branching score for a variable i at a node n

- Let L be the value of LP relaxation of the MILP

A gold standard: Strong Branching (impractical)

Strong branching score for a variable i at a node n

- Let L be the value of LP relaxation of the MILP
- Denote L_i^+ as the value of LP relaxation of the MILP *after adding $x_i \geq \lceil x_i^* \rceil$ constraint*

A gold standard: Strong Branching (impractical)

Strong branching score for a variable i at a node n

- Let L be the value of LP relaxation of the MILP
- Denote L_i^+ as the value of LP relaxation of the MILP *after adding* $x_i \geq \lceil x_i^* \rceil$ *constraint*
- Similarly, denote L_i^- for the other half

A gold standard: Strong Branching (impractical)

Strong branching score for a variable i at a node n

- Let L be the value of LP relaxation of the MILP
- Denote L_i^+ as the value of LP relaxation of the MILP *after adding* $x_i \geq \lceil x_i^* \rceil$ *constraint*
- Similarly, denote L_i^- for the other half

Strong branching score

$$\text{score}_{SB,i} = \max(L - L_i^+, \epsilon) \times \max(L - L_i^-, \epsilon)$$

A gold standard: Strong Branching (impractical)

Strong branching score for a variable i at a node n

- Let L be the value of LP relaxation of the MILP
- Denote L_i^+ as the value of LP relaxation of the MILP *after adding* $x_i \geq \lceil x_i^* \rceil$ *constraint*
- Similarly, denote L_i^- for the other half

Strong branching score

$$\text{score}_{SB,i} = \max(L - L_i^+, \epsilon) \times \max(L - L_i^-, \epsilon)$$

Strong branching decision

$$i_{SB}^{\star} = \arg\max_i \quad \text{score}_{SB,i}$$

Outline

Learning to branch

Objective:
Given a distribution of problem sets, find a branching policy that yields a shortest tree on an average.
Exploits statistical correlation across problem sets.

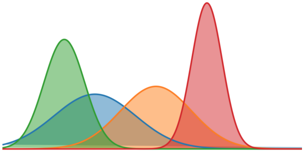


Figure: Application specific distribution

Learning to branch

Objective: Given a dataset of MILPs

- learn an inexpensive function f
- that imitates strong branching decisions (computationally expensive)

Learning to branch

Objective: Given a dataset of MILPs

- learn an inexpensive function f
- that imitates strong branching decisions (computationally expensive)

$$i^\star_{SB} = \arg\max_{i \in \mathcal{C}} \text{score}_{SB,i} \qquad i^\star_{f} = \arg\max_{i \in \mathcal{C}} \text{score}_{f_\theta,i},$$

where $s^i_{f_\theta}$ is the score for $i \leq p$ variable as estimated by f_θ.

Learning to branch

Objective: Given a dataset of MILPs

- learn an inexpensive function f
- that imitates strong branching decisions (computationally expensive)

$$i^{\star}_{SB} = \arg\max_{i \in \mathcal{C}} \text{score}_{SB,i} \qquad i^{\star}_{f} = \arg\max_{i \in \mathcal{C}} \text{score}_{f_\theta,i},$$

where $s^{i}_{f_\theta}$ is the score for $i \leq p$ variable as estimated by f_θ.

$$\theta^{*} = \arg\min_{\theta} \mathcal{L}(f_\theta(MILP), i^{\star}_{SB})$$

Learning to branch

Objective: Given a dataset of MILPs

- learn an inexpensive function f
- that imitates strong branching decisions (computationally expensive)

$$i^{\star}_{SB} = \arg\max_{i \in \mathcal{C}} \text{score}_{SB,i} \qquad i^{\star}_{f} = \arg\max_{i \in \mathcal{C}} \text{score}_{f_\theta,i},$$

where $s^{i}_{f_\theta}$ is the score for $i \leq p$ variable as estimated by f_θ.

$$\theta^{*} = \arg\min_{\theta} \mathcal{L}(f_\theta(MILP), i^{\star}_{SB})$$

Well studied problem (not an exhaustive list)

- Gasse et al., 2019 $\implies$ offline imitation learning using GCNN
- Nair et al., 2020 $\implies$ uses GCNNs to design other heuristics
- Chen et al., 2022 $\implies$ studies the limitations of existing GNNs to represent MILPs

Learning to branch: GNNs

Gasse et al., 2019 uses Graph Neural Networks to imitate the strong branching policy *through classification framework*

+ superior representation power
+ best overall accuracy

Learning to branch: GNNs

Gasse et al., 2019 uses Graph Neural Networks to imitate the strong branching policy *through classification framework*

- \+ superior representation power
- \+ best overall accuracy

Model inputs

Inputs to the GNN is a **bipartite-representation of MILP**: G

GNNs: Bipartite Representation of MILPs

Natural representation : variable / constraint bipartite graph

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

GNNs: Bipartite Representation of MILPs

Natural representation : variable / constraint bipartite graph

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

x_0

x_1

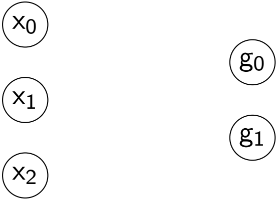
x_2

- x_i: variable features (type, coef., bounds, LP solution...)

GNNs: Bipartite Representation of MILPs

Natural representation : variable / constraint bipartite graph

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

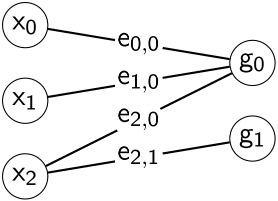


- x_i: variable features (type, coef., bounds, LP solution...)
- g_j: constraint features (right-hand-side, LP slack...)

GNNs: Bipartite Representation of MILPs

Natural representation : variable / constraint bipartite graph

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



- x_i: variable features (type, coef., bounds, LP solution...)
- g_j: constraint features (right-hand-side, LP slack...)
- $e_{i,j}$: non-zero coefficients in A

Learning to branch: GNNs

Gasse et al., 2019 uses Graph Neural Networks to imitate the strong branching policy *through classification framework*

+ superior representation power
+ best overall accuracy

Learning to branch: GNNs

Gasse et al., 2019 uses Graph Neural Networks to imitate the strong branching policy *through classification framework*

+ superior representation power
+ best overall accuracy
− requires GPUs for best running times (Gupta et al., 2020 addresses this drawback)

Learning to branch: GNNs

Gasse et al., 2019 uses Graph Neural Networks to imitate the strong branching policy *through classification framework*

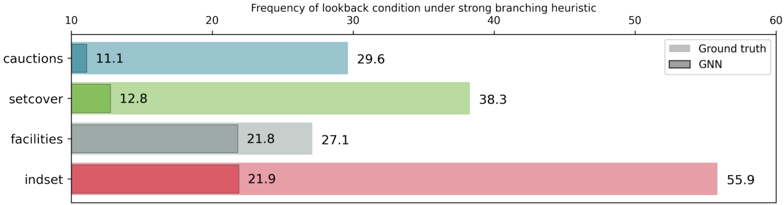
+ superior representation power
+ best overall accuracy
− requires GPUs for best running times (Gupta et al., 2020 addresses this drawback)
? Can we further improve the performance?

Outline

Lookback condition in strong branching

Strong branching heuristic exhibits the following condition:
Parent's second best choice is *often* the **child's best choice**.

Frequency of Lookback condition



Frequency of Lookback condition

Instances	Description	number of parent-child pairs collected	number of parent-child pairs exhibiting the lookback property	Frequency of the lookback property
CORLAT	Corridor planning in wildlife management	5082	1765	34.73%
RCW	Red-cockaded woodpecker diffusion conservation	5115	1952	38.16%

Frequency of the lookback property in the real-world instances is as prevalent as in the synthetic instances considered in the main paper. These instances are made available by Dilkina et al., 2017.

Outline

Outline

Loss targets

We consider two types of targets
($\mathcal{Z}$ is the set of all the second best branching variables)

Original one-hot encoded target,

$$\mathrm{y}$$

$$\mathrm{y}_i = \begin{cases} 1, & i = i^*_{SB} \\ 0, & \text{otherwise} \end{cases}$$

Loss targets

We consider two types of targets
($\mathcal{Z}$ is the set of all the second best branching variables)

Original one-hot encoded target,

y

$$\mathrm{y}_i = \begin{cases} 1, & i = i^*_{SB} \\ 0, & \text{otherwise} \end{cases}$$

$$\theta^\star_y = \arg\min_\theta \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{y}_k)$$

Loss targets

We consider two types of targets
($\mathcal{Z}$ is the set of all the second best branching variables)

Original one-hot encoded target, y

Second-best ϵ-smoothed target, z

$$\mathrm{y}_i = \begin{cases} 1, & i = i^*_{SB} \\ 0, & \text{otherwise} \end{cases}$$

$$\mathrm{z}_i = \begin{cases} 1-\epsilon, & i = i^*_{SB} \\ \frac{\epsilon}{|\mathcal{Z}|}, & i \in \mathcal{Z} \\ 0, & \text{otherwise} \end{cases}$$

$$\theta^\star_y = \arg\min_\theta \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{y}_k)$$

Loss targets

We consider two types of targets
($\mathcal{Z}$ is the set of all the second best branching variables)

Original one-hot encoded target, y

Second-best ϵ-smoothed target, z

$$y_i = \begin{cases} 1, & i = i^*_{SB} \\ 0, & \text{otherwise} \end{cases}$$

$$z_i = \begin{cases} 1-\epsilon, & i = i^*_{SB} \\ \frac{\epsilon}{|\mathcal{Z}|}, & i \in \mathcal{Z} \\ 0, & \text{otherwise} \end{cases}$$

$$\theta^\star_y = \arg\min_\theta \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), y_k)$$

$$\theta^\star_z = \arg\min_\theta \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), z_k)$$

Outline

Parent-As-Target (PAT) regularizer

We consider a regularizer to encourage the lookback proprety in GNNs

Parent-As-Target (PAT) regularizer

We consider a regularizer to encourage the lookback proprety in GNNs

$$\text{loss}_{PAT} = 1\{Lookback_i\}\cdot$$

Parent-As-Target (PAT) regularizer

We consider a regularizer to encourage the lookback proprety in GNNs

$$\text{loss}_{PAT} = 1\{Lookback_i\} \cdot CE(f_\theta(\mathcal{G}_i), ??),$$

Parent-As-Target (PAT) regularizer

We consider a regularizer to encourage the lookback proprety in GNNs

$$\text{loss}_{PAT} = 1\{Lookback_i\} \cdot CE(f_\theta(\mathcal{G}_i), f_\theta(\mathcal{G}_i^{parent})[\mathcal{C}_i]),$$

Outline

Performance evaluation

We will consider three different set of parameters

- Choice of the target:
  - One-hot encoded, y
  - Second-best ϵ-smoothed, z
- Strength of the PAT regularizer, $\lambda_{PAT} \in \{0, 0.01, 0.1, 0.2, 0.3\}$
- Strength of the $l2$-regularizer, $\lambda_{l2} \in \{0.0, 0.01, 0.1, 1.0\}$

Performance evaluation

$$\theta_y = \underset{\theta, \lambda_{l2}}{\arg\min} \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{y}_k) + \lambda_{l2} \cdot ||\theta||_2$$

Performance evaluation

$$\theta_y = \underset{\theta, \lambda_{l2}}{\arg\min} \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{y}_k) + \lambda_{l2} \cdot ||\theta||_2$$

$$\theta_z = \underset{\theta, \lambda_{l2}}{\arg\min} \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{z}_k) + \lambda_{l2} \cdot ||\theta||_2$$

Performance evaluation

$$\theta_y = \underset{\theta, \lambda_{l2}}{\arg\min} \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{y}_k) + \lambda_{l2} \cdot ||\theta||_2$$

$$\theta_z = \underset{\theta, \lambda_{l2}}{\arg\min} \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{z}_k) + \lambda_{l2} \cdot ||\theta||_2$$

$$\theta_{PAT} = \underset{\theta, \mathrm{v}, \lambda_{l2}, \lambda_{PAT}}{\arg\min} \frac{1}{N} \sum_{k=1}^{N} CE(f_\theta(\mathcal{G}_k), \mathrm{v}) + \lambda_{l2} \cdot ||\theta||_2 + \lambda_{PAT} \cdot \mathrm{loss}_{PAT}$$

Performance evaluation: Instances

- **Small** instances are used to *collect training data* of parent-child nodes by solving these instances using the strong branching heuristic as the variable selection policy in the solver

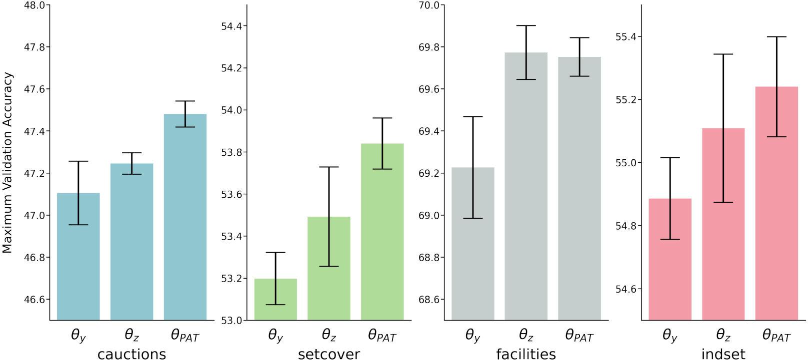
Performance evaluation: Instances

- **Small** instances are used to *collect training data* of parent-child nodes by solving these instances using the strong branching heuristic as the variable selection policy in the solver
- **Medium** instances are used for *hyperparameter selection* incorporating harder-to-formulate criterion in the objective function

Performance evaluation: Instances

- **Small** instances are used to *collect training data* of parent-child nodes by solving these instances using the strong branching heuristic as the variable selection policy in the solver
- **Medium** instances are used for *hyperparameter selection* incorporating harder-to-formulate criterion in the objective function
- **Big** instances are used to *report performance* evaluation

Model selection criterion: Validation accuracy



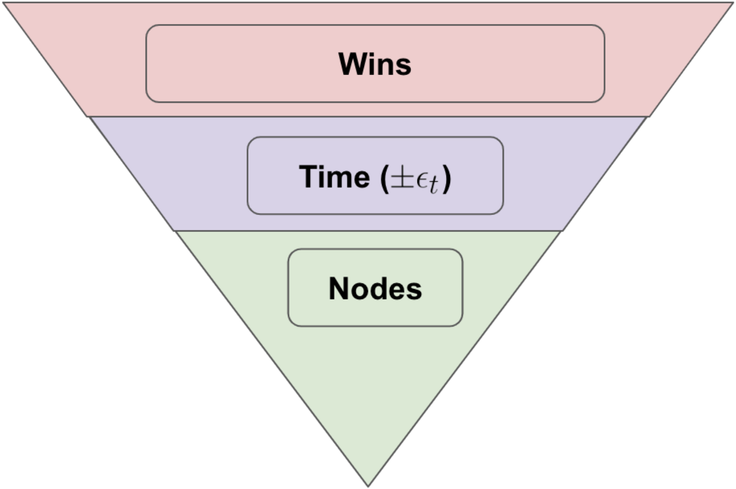
Top-1 accuracy (1-standard deviation) on validation dataset.

Model selection criterion: Out-of-distribution performance

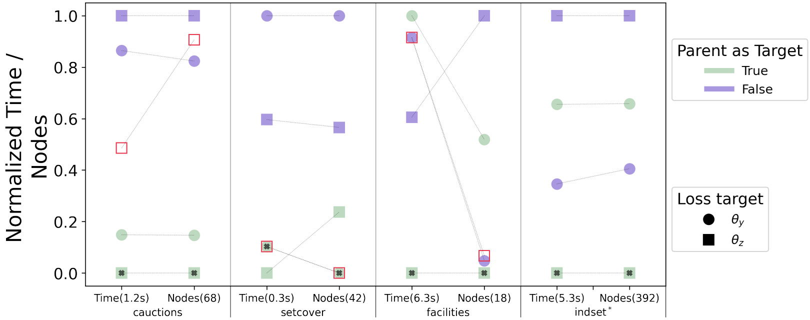
We solve 100 **medium** instances and collect the following metrics

- Wins: Number of times a model solved the instance fastest
- Time: 1-shifted geometric mean of time taken to solve each instance
- Nodes: 1-shifted geometric mean of nodes taken in the B&B tree of the *commonly solved instances*

Model selection criterion: Out-of-distribution performance



Model selection criterion: Out-of-distribution performance



We plot the range-normalized (range is specified in parenthesis) Time and Node performance of the selected models. The centered "X" black mark shows the final models that were selected to be used for evaluating the performance on Big instances. The points with a red outline show the performance of the models selected according to the best validation accuracy (Note that we omit such models for indset as it distorts the scale of the plot.)

Final performance

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	1	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**69**	**84**	**16 388**

Combinatorial Auction (Bigger)

Figure: Evaluation metrics on **Big** instances with a time budget of 30 minutes per instance

Final performance

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	1	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**69**	**84**	**16 388**
		Combinatorial Auction (Bigger)			

Figure: Evaluation metrics on Big instances with a time budget of 30 minutes per instance

Final performance

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	1	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**69**	**84**	**16 388**
		Combinatorial Auction (Bigger)			

Figure: Evaluation metrics on **Big** instances with a time budget of 30 minutes per instance

Final performance

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	1	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**69**	**84**	**16 388**

Combinatorial Auction (Bigger)

Figure: Evaluation metrics on Big instances with a time budget of 30 minutes per instance

Final performance

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	1	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**69**	**84**	**16 388**

Combinatorial Auction (Bigger)

Figure: Evaluation metrics on **Big** instances with a time budget of 30 minutes per instance

Final performance

Model	Time	Time (c)	Wins	Solved	Nodes (c)
FSB*	n/a	n/a	n/a	n/a	n/a
RPB	626.81	434.92	1	80	17 979
TUNEDRPB	644.20	450.06	0	80	18 104
GNN	507.06	333.59	14	80	17 145
GNN-PAT (ours)	**477.26**	**310.22**	**69**	**84**	**16 388**

Combinatorial Auction (Bigger)

Figure: Evaluation metrics on **Big** instances with a time budget of 30 minutes per instance

Optimality gap on commonly unsolved instances

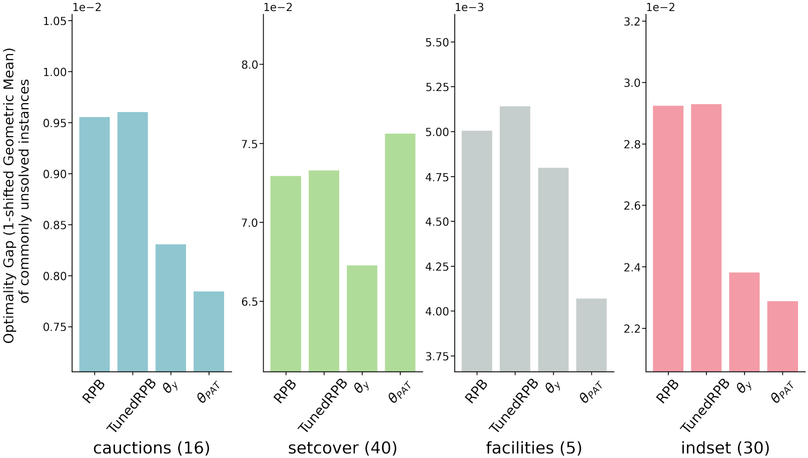


Figure: **Mean optimality gap of the *commonly unsolved instances***

Outline

Conclusion

- We **discover** *lookback* phenomenon in the gold-standard (by tree size) variable-selection heuristic

Conclusion

- We **discover** *lookback* phenomenon in the gold-standard (by tree size) variable-selection heuristic
- We **proposed** second-best ϵ-smoothed target and a PAT regularizer term to incorporate lookback phenomenon in deep learning models

Conclusion

- We **discover** *lookback* phenomenon in the gold-standard (by tree size) variable-selection heuristic
- We **proposed** second-best ϵ-smoothed target and a PAT regularizer term to incorporate lookback phenomenon in deep learning models
- We **proposed** a model selection scheme to incorporate final utility of these models in the objective function

Conclusion

- We **discover** *lookback* phenomenon in the gold-standard (by tree size) variable-selection heuristic
- We **proposed** second-best ϵ-smoothed target and a PAT regularizer term to incorporate lookback phenomenon in deep learning models
- We **proposed** a model selection scheme to incorporate final utility of these models in the objective function
- Our proposed models outperform the SOTA results

Open questions

- Discovery of more inductive biases

QR Codes generated via `https://www.qr-code-generator.com/`

Open questions

- Discovery of more inductive biases
- Designing better ways to incorporate lookback property

QR Codes generated via `https://www.qr-code-generator.com/`

Open questions

- Discovery of more inductive biases
- Designing better ways to incorporate lookback property
- Improve reinforcement learning solutions using the lookback property

QR Codes generated via `https://www.qr-code-generator.com/`

Open questions

- Discovery of more inductive biases
- Designing better ways to incorporate lookback property
- Improve reinforcement learning solutions using the lookback property

Paper: `https://arxiv.org/abs/2006.15212`

QR Codes generated via `https://www.qr-code-generator.com/`

Lookback for Learning to Branch

Thank you!

Prateek Gupta*, Elias B. Khalil, Didier Chételat, Maxime Gasse, M. Pawan Kumar, Andrea Lodi, Yoshua Bengio