Hybrid Models for Learning to Branch

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

Huawei London Research Center, Nov. 26th 2021

Resources

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

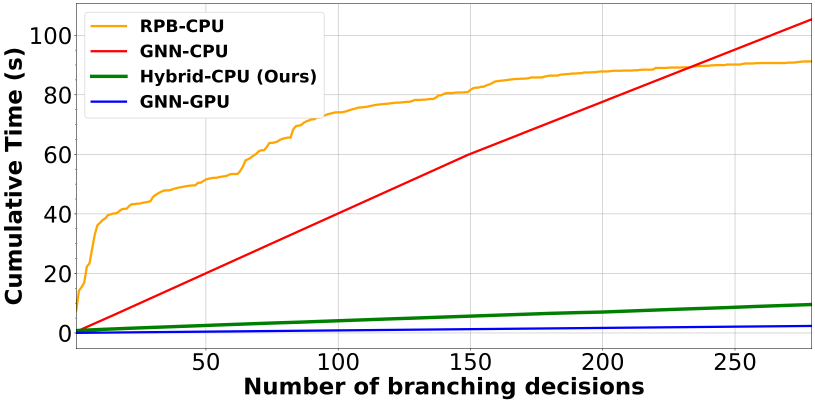
Code: `https://github.com/pg2455/Hybrid-learn2branch`

Slides: `www.pgupta.info/talks`

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

To enable **CPU-based discrete optimization solvers** to **use deep learning models** without sacrificing runtime performance.

To enable **CPU-based discrete optimization solvers** to **use deep learning models** without sacrificing runtime performance.



Outline

Problem formulation

Our approach: Hybrid Models

Conclusion

Outline

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)

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

- $c \in \mathbb{R}^n$ 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

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

Mixed-Integer Linear Program (MILP)

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

- $\mathrm{c} \in \mathbb{R}^n$ 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

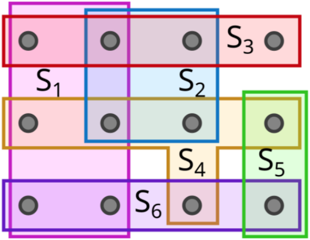
Combinatorial Auctions

Facility location-Allocation

Maximum Indendent Set

Set Covering

and many more ...



Mixed-Integer Linear Program (MILP)

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

- $c \in \mathbb{R}^n$ 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

NP-hard problem.

Mixed-Integer Linear Program (MILP)

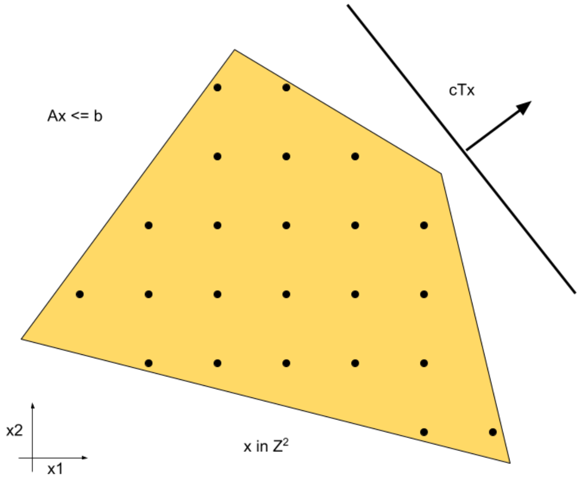


Image credit: Maxime Gasse

Linear Program (LP)

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

- $c \in \mathbb{R}^n$ 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

Linear Program (LP)

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

- $\mathsf{c} \in \mathbb{R}^n$ 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

Linear Program (LP)

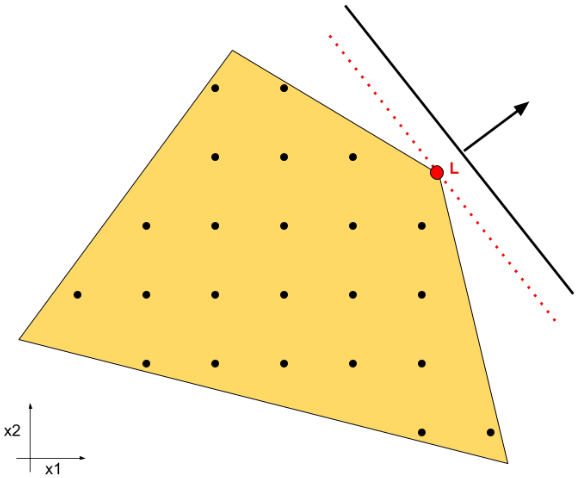
$$\begin{aligned} \arg\min_{x} \quad & c^\top x \\ \text{subject to} \quad & Ax \le b, \\ & l \le x \le u, \\ & \boxed{x \in \mathbb{R}^n}. \end{aligned}$$

- $c \in \mathbb{R}^n$ 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

◦ Yields lower bounds to the original MILP

LP Relaxation of a MILP

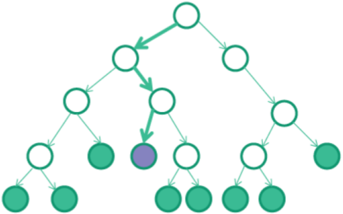


Outline

Branch-and-Bound (B&B)

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

It consists of two steps



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

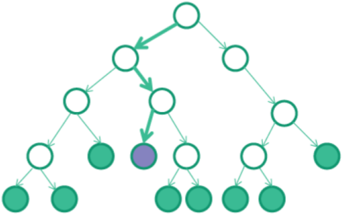
Image source: https://www.gurobi.com/resource/mip-basics/

Branch-and-Bound (B&B)

B&B (Land et al., 1960) is the widely used 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

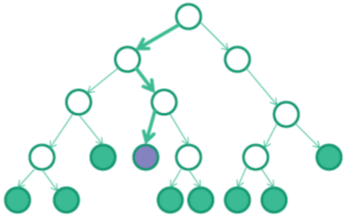
Image source: https://www.gurobi.com/resource/mip-basics/

Branch-and-Bound (B&B)

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

It consists of two steps

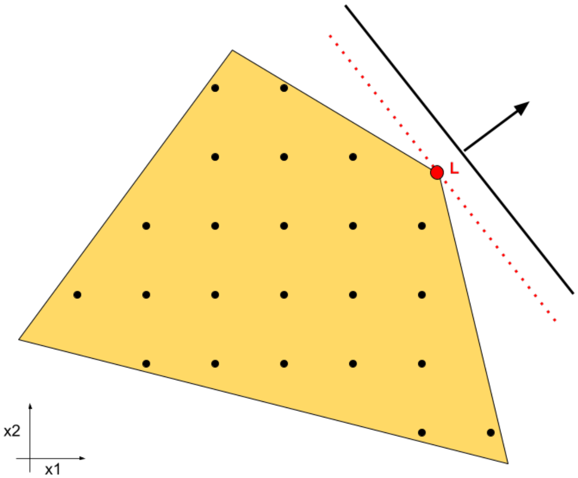
- **Branching** - Select variable to split the problem into two
- **Bounding** - Solve the LP relaxation of resulting problem to obtain optimization guarantees on the solution



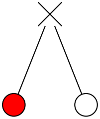
Each node in branch-and-bound is a new MIP

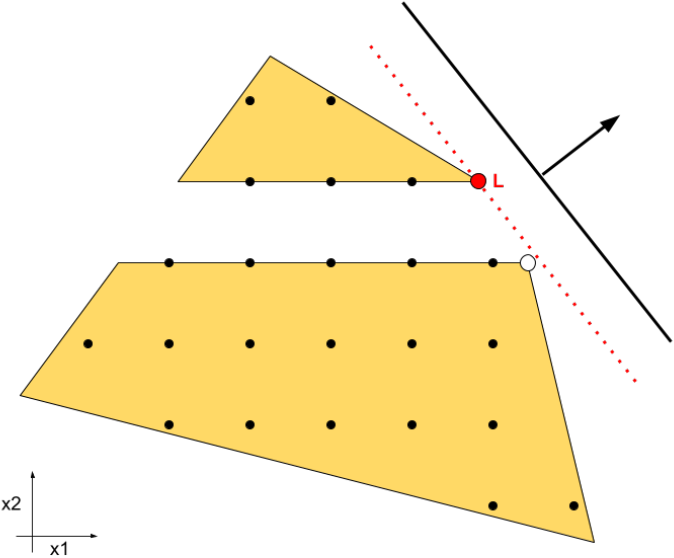
Image source: https://www.gurobi.com/resource/mip-basics/

LP Relaxation of a MILP

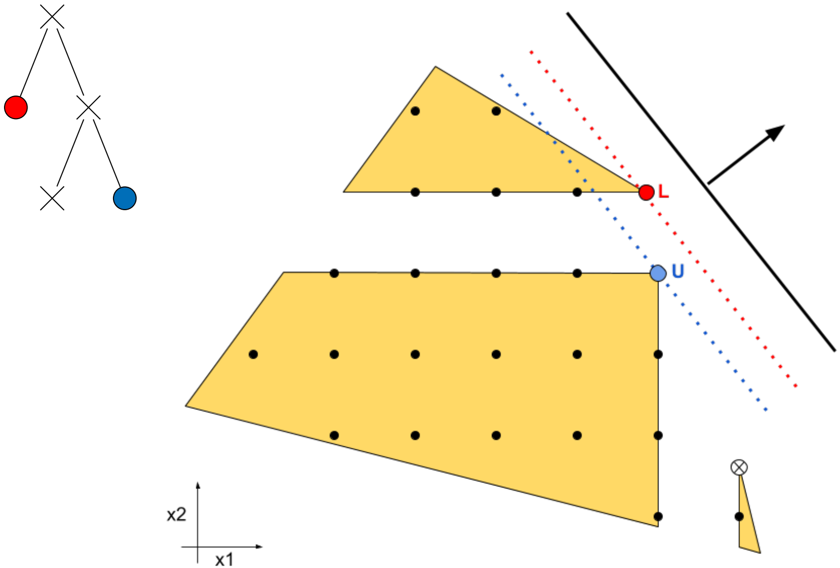


Branch-and-Bound

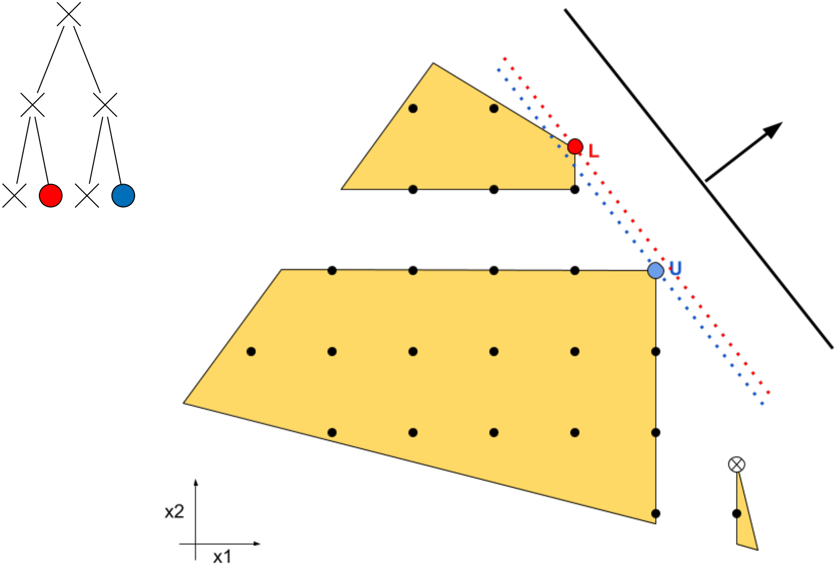




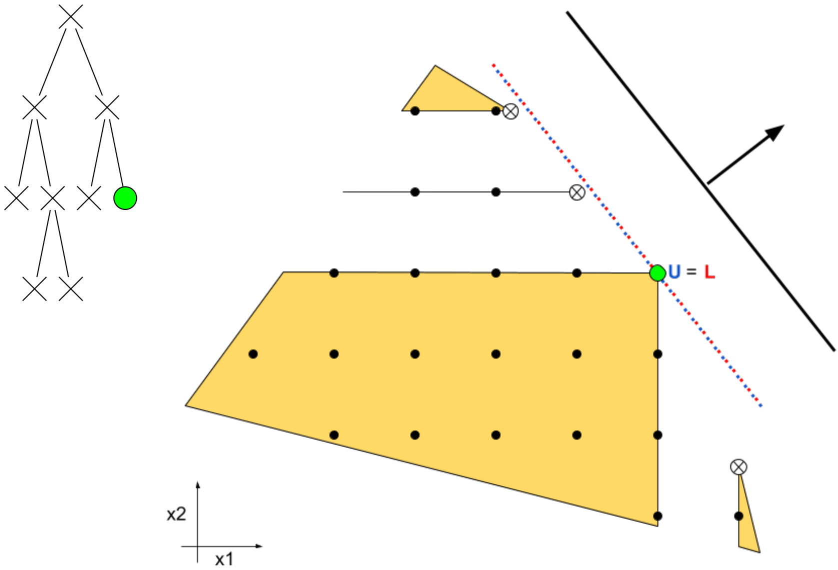
Branch-and-Bound



Branch-and-Bound



Branch-and-Bound



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:

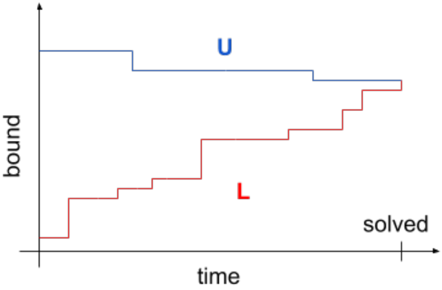
- L = U (optimality certificate)
- L = ∞ (infeasibility certificate)
- L - U < threshold (early stopping)

Note: A time limit is used to ensure termination.

Branch-and-bound: a sequential process

Sequential decisions:

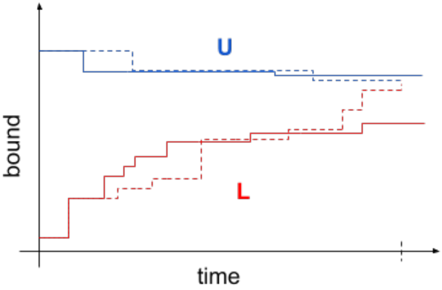
- variable selection (branching)
- node selection
- *cutting plane selection*
- *primal heuristic selection*
- *simplex initialization*
- ...



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

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

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 branching (PB): backward looking

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

Reliability pseudo-cost branching (RPB): 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)

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 branching (PB): backward looking

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

Reliability pseudo-cost branching (RPB): 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)

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 branching (PB): backward looking

- keep track of tightenings in past branchings
- pick the most promising variable

+ very fast, almost no computations

− cold start

Reliability pseudo-cost branching (RPB): 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)

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.

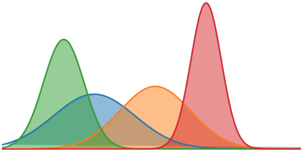


Figure: Application specific distribution

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)

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

where $s^i_{f_\theta}$ is the score for $i \le p$ 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)

- Khalil et al., 2016 $\Longrightarrow$ "online" imitation learning
- Balcan et al., 2018 $\Longrightarrow$ theoretical results
- Gasse et al., 2019 $\Longrightarrow$ offline imitation learning using GCNN

Learning to branch: SVMs

Khalil et al., 2016 uses Support Vector Machines (SVM) to imitate the strong branching policy *through learning-to-rank framework*

+ adapts to the problem instance instead of the distribution
+ computationally inexpensive once the SVM weights are learned
− less representational power as compared to GNNs

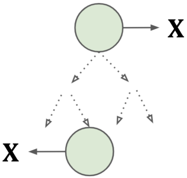
Learning to branch: SVMs

Khalil et al., 2016 uses Support Vector Machines (SVM) to imitate the strong branching policy *through learning-to-rank framework*

+ adapts to the problem instance instead of the distribution
+ computationally inexpensive once the SVM weights are learned
− less representational power as compared to GNNs

Model inputs

Inputs to the SVM model are **hand-designed features**: X



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

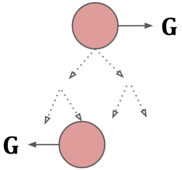
Learning to branch: GNNs

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

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

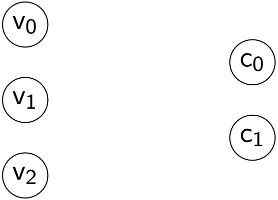
$$
\begin{aligned}
\arg\min_{x} \quad & c^\top x \\
\text{subject to} \quad & Ax \leq b, \\
& l \leq x \leq u, \\
& x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}.
\end{aligned}
$$

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

GNNs: Bipartite Representation of MILPs

Natural representation : variable / constraint bipartite graph

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

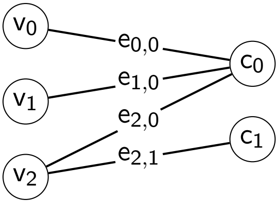


- v_i: variable features (type, coef., bounds, LP solution. . .)
- c_j: constraint features (right-hand-side, LP slack. . .)

GNNs: Bipartite Representation of MILPs

Natural representation : variable / constraint bipartite graph

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



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

Outline

MILP Solvers

MILP solvers do not use GPUs.
Use of GNNs can get infeasible in the following scenarios

- No GPUs: It will be infeasible to incorporate GNNs as a branching policy in any of the available solvers
- Parallel MILP solving: As a single GPU can only fit a limited number of GNNs, when several 100s of MILPs need to be solved in parallel, GNNs can get infeasible

Outline

Data Extraction

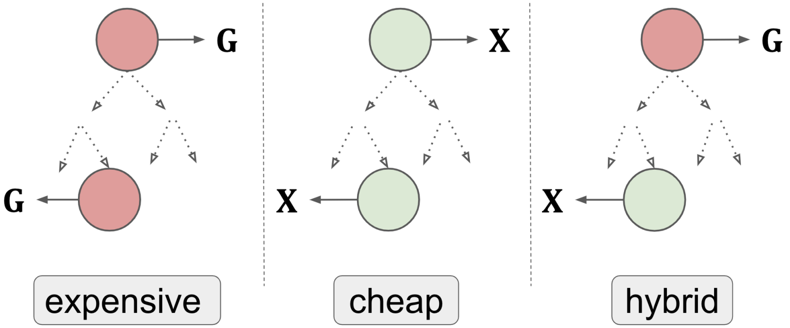
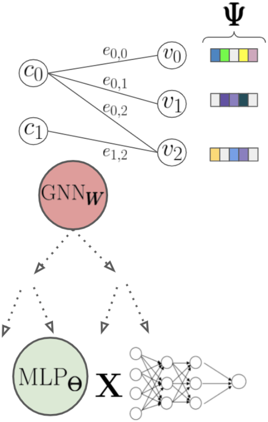


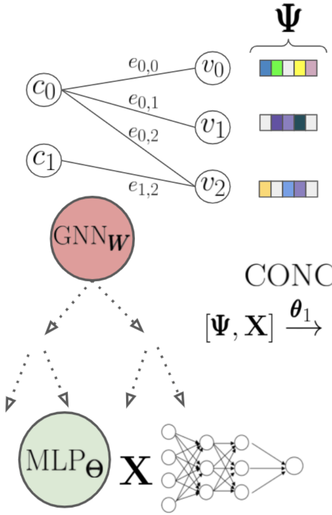
Figure: Data extraction strategies: bipartite graph representation G at every node (expensive); candidate variable features X at every node (cheap); bipartite graph at the root node and variable features at tree node (hybrid).

Outline

Model Architecture



Model Architecture: CONCAT

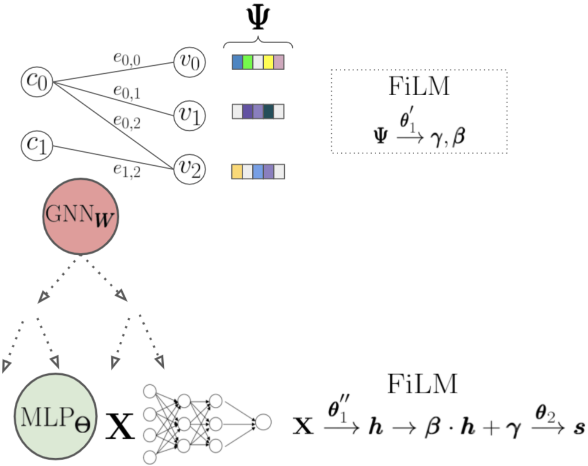


CONCAT

$$[\boldsymbol{\Psi}, \mathbf{X}] \xrightarrow{\boldsymbol{\theta}_1} \mathbf{h} \xrightarrow{\boldsymbol{\theta}_2} \boldsymbol{s}$$

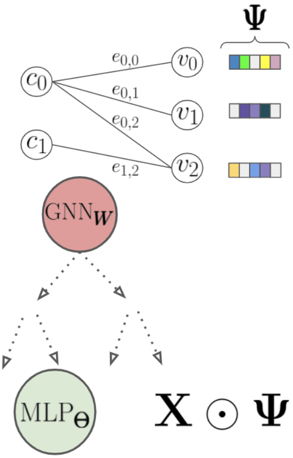
Perez et al., 2018 first proposed FiLM for visual question answering task

Model Architecture: FiLM



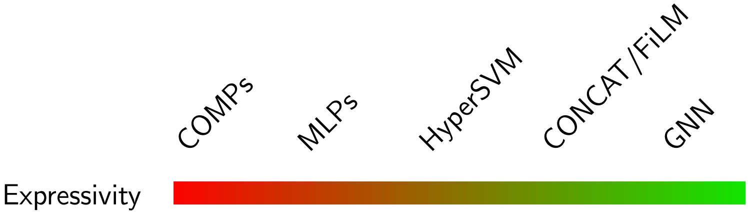
Perez et al., 2018 first proposed FiLM for visual question answering task

Model Architecture: HyperSVM

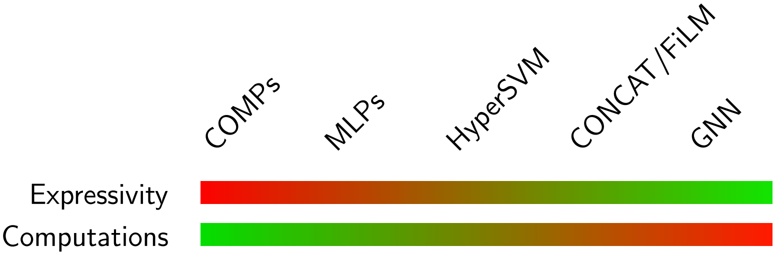


Perez et al., 2018 first proposed FiLM for visual question answering task

Model Architecture



Model Architecture



Model Architecture: Performance

End-to-end training

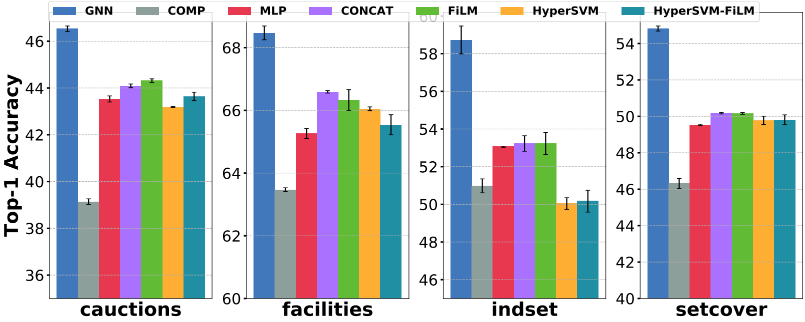


Figure: Test accuracy of the different models, with a simple e2e training protocol.

Outline

- Problem formulation
- Our approach: Hybrid Models
  - Model Architecture
  - Training Protocols
- Conclusion

Training Protocols

To enhance the generalization power of the learned models on the bigger instances

Training Protocols: Loss weights

A good decision closer to the root node is more important than the ones far away from it.

Table: Effect of different sample weighting schemes on combinatorial auctions (big) instances, with a simple MLP model. $z \in [0, 1]$ is the ratio of the depth of the node and the maximum depth observed in a tree.

Type	Weighting scheme	Nodes	Wins
Constant	1	9678	10/60
Exponential decay	$e^{-0.5z}$	9793	10/60
Linear	$(e^{-0.5} - 1) * z + 1$	9789	12/60
Quadratic decay	$(e^{-0.5} - 1) * z^2 + 1$	9561	14/60
Sigmoidal	$(1 + e^{-0.5})/(1 + e^{z-0.5})$	**9534**	14/60

Training Protocols: Knowledge Distillation

Knowledge distillation (KD):[2] Use the output of an expert GNN from Gasse et al., 2019 as a target for the model.

KD reweights the samples so that the student doesn't attempt to sharply classify samples that even the teacher didn't succeed with (i.e. the logits have higher entropy for the more difficult samples) Phuong et al., 2019.

Table: Test accuracy of FiLM, using different training protocols.

	cauctions	facilities	indset	setcover
Pretrained GNN	44.12 $\pm$ 0.09	65.78 $\pm$ 0.06	53.16 $\pm$ 0.51	50.00 $\pm$ 0.09
e2e	44.31 $\pm$ 0.08	66.33 $\pm$ 0.33	53.23 $\pm$ 0.58	50.16 $\pm$ 0.05
e2e & KD	44.10 $\pm$ 0.09	66.60 $\pm$ 0.21	53.08 $\pm$ 0.3	50.31 $\pm$ 0.19

[2] G. Hinton et al. (2015). Distilling the knowledge in a neural network.

Training Protocols: Auxiliary Task

Auxiliary Task (AT): No additional data requied. We force the variable representations to be far apart from each other.

- ED : Maximum distance between these representations in euclidean space
- MHE[3] : Uniform distribution of these representations over a unit hypersphere

Table: Test accuracy of FiLM, using different training protocols.

	cauctions	facilities	indset	setcover
Pretrained GNN	44.12 $\pm$ 0.09	65.78 $\pm$ 0.06	53.16 $\pm$ 0.51	50.00 $\pm$ 0.09
e2e	44.31 $\pm$ 0.08	66.33 $\pm$ 0.33	53.23 $\pm$ 0.58	50.16 $\pm$ 0.05
e2e & KD	44.10 $\pm$ 0.09	66.60 $\pm$ 0.21	53.08 $\pm$ 0.3	50.31 $\pm$ 0.19
e2e & KD & AT	**44.56 $\pm$ 0.13**	**66.85 $\pm$ 0.28**	**53.68 $\pm$ 0.23**	**50.37 $\pm$ 0.03**

[3] W. Liu et al. (2018). Learning towards minimum hyperspherical energy.

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

Model	Easy			Medium			Hard		
	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
$\boxed{\text{GNN}}$	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4]A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[4].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[4] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Finally, the learned models are used as a branching policy in SCIP solver[5].

Hybrid models have a better runtime performance on average than other baselines as evaluated on CPU only machines.

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	42.5	1 / 60	13	313.3	0 / 59	75	997.2	0 / 51	50
PB	31.4	4 / 60	139	177.7	4 / 60	384	712.6	3 / 56	309
RPB	36.9	1 / 60	**23**	214.0	1 / 60	**152**	794.8	2 / 54	**99**
COMP	30.4	3 / 60	120	172.5	4 / 60	347	633.4	6 / 57	294
GNN	39.2	0 / 60	112	209.8	0 / 60	314	748.8	0 / 54	286
FiLM (ours)	**24.7**	**51** / 60	109	**136.4**	**51** / 60	325	**531.7**	**46** / 57	295
GNN	28.9	– / 60	112	150.1	– / 60	314	628.1	– / 56	286

Capacitated Facility Location

[5] A. Gleixner et al. (July 2018). The SCIP Optimization Suite 60. Technical Report. Optimization Online

B&B Performance (Runtime)

Model	Small Time	Small Wins	Small Nodes	Medium Time	Medium Wins	Medium Nodes	Big Time	Big Wins	Big Nodes
FSB	42.53	1 / 60	13	313.33	0 / 59	75	997.23	0 / 51	50
PB	31.35	4 / 60	139	177.69	4 / 60	384	712.45	3 / 56	309
RPB	36.86	1 / 60	**23**	213.99	1 / 60	**152**	794.80	2 / 54	**99**
COMP	30.37	3 / 60	120	172.51	4 / 60	347	633.42	6 / 57	294
GNN	39.18	0 / 60	112	209.84	0 / 60	314	748.85	0 / 54	286
FiLM (ours)	**24.67**	**51** / 60	109	**136.42**	**51** / 60	325	**531.70**	**46** / 57	295
GNN-GPU	28.91	– / 60	112	150.11	– / 60	314	628.12	– / 56	286
				Capacitated Facility Location					
FSB	27.16	0 / 60	17	582.18	0 / 45	116	2700.00	0 / 0	n/a
PB	10.19	0 / 60	286	94.12	0 / 60	2451	2208.57	0 / 23	82 624
RPB	14.05	0 / 60	**54**	94.65	0 / 60	1129	1887.70	7 / 27	48 395
COMP	9.83	3 / 60	178	89.24	0 / 60	1474	2166.44	0 / 21	52 326
GNN	17.61	0 / 60	136	242.15	0 / 60	**1013**	2700.17	0 / 0	n/a
FiLM (ours)	**8.73**	**57** / 60	147	**63.75**	**60** / 60	1131	**1843.24**	**20** / 26	**37 777**
GNN-GPU	8.26	– / 60	136	53.56	– / 60	1013	1535.80	– / 36	31 662
				Set Covering					
FSB	6.12	0 / 60	6	132.38	0 / 60	71	2127.35	0 / 28	318
PB	2.76	1 / 60	234	25.83	0 / 60	2765	393.60	0 / 59	13 719
RPB	4.01	0 / 60	**11**	26.36	0 / 60	714	**210.95**	**29** / 60	4701
COMP	2.76	0 / 60	82	29.76	0 / 60	930	494.59	0 / 54	5613
GNN	2.73	1 / 60	71	22.26	0 / 60	688	257.99	6 / 60	**3755**
FiLM (ours)	**2.13**	**58** / 60	73	**15.71**	**60** / 60	**686**	217.02	25 / 60	4315
GNN-GPU	1.96	– / 60	71	11.70	– / 60	688	121.18	– / 60	3755
				Combinatorial Auctions					
FSB	673.43	0 / 53	47	1689.75	0 / 20	10	2700.00	0 / 0	n/a
PB	172.03	2 / 57	5728	753.95	0 / 45	1570	2685.23	0 / 1	38 215
RPB	59.87	5 / 60	603	173.17	11 / 60	**205**	1946.51	9 / 21	2461
COMP	82.22	1 / 58	847	383.97	1 / 52	267	2393.75	0 / 6	5589
GNN*	**44.07**	15 / 60	**331**	625.23	1 / 50	599	2330.95	0 / 10	**687**
FiLM* (ours)	52.96	**37** / 55	376	**131.45**	**47** / 54	264	**1823.29**	**12** / 15	1201
GNN-GPU*	31.71	– / 60	331	63.96	– / 60	599	1158.59	– / 27	685
				Maximum Independent Set					

B&B Performance (Optimality Gap)

Hybrid models also have the least optimality gap at the end of the time limit as compared to other baselines as evaluated on CPU only machines.

Table: Mean optimality gap (lower the better) of commonly unsolved "big" instances (number of such instances in brackets).

	setcover (33)	indset (39)
FSB	0.1709	0.0755
PB	0.0713	0.0298
RPB	0.0628	0.0252
COMP	0.0740	0.0252
GNN	0.1039	0.0341
FiLM	**0.0597**	**0.0187**

Runtime performance

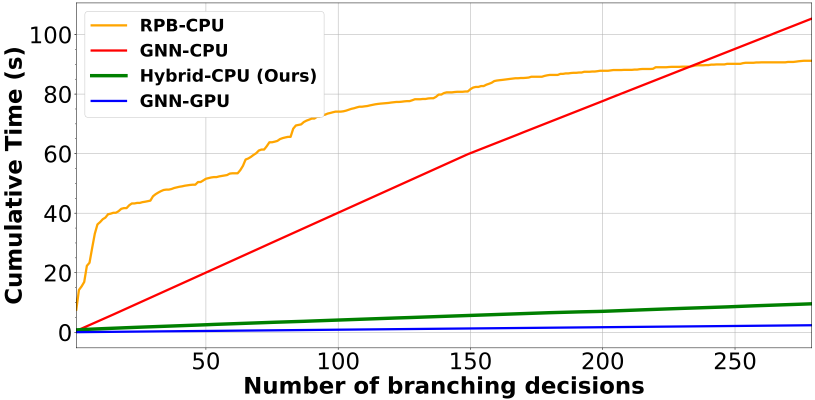


Figure: Cumulative time cost of different branching policies: (i) the default internal rule RPB of the SCIP solver; (ii) a GNN model (using a GPU or a CPU); and (iii) our hybrid model. Clearly the GNN model requires a GPU for being competitive, while our hybrid model does not. (Measured on a capacitated facility location problem, medium size).

Outline

Problem formulation

Our approach: Hybrid Models

Conclusion

Conclusion

- We **propose hybrid models** as branching policies that can be easily integrated with any discrete optimization solvers without needing any access to GPU machines

Conclusion

- We **propose hybrid models** as branching policies that can be easily integrated with any discrete optimization solvers without needing any access to GPU machines
- We **explored various training protocols** to enhance both the Top-1 accuracy as well as the runtime B&B performance of these hybrid models

Conclusion

- We **propose hybrid models** as branching policies that can be easily integrated with any discrete optimization solvers without needing any access to GPU machines
- We **explored various training protocols** to enhance both the Top-1 accuracy as well as the runtime B&B performance of these hybrid models
- We **recommend** training protocol as FiLM (e2e & KD & AT) only when these models have significantly better accuracy than FiLM (e2e & KD).

Open questions

- scaling to the real-world problems
- reinforcement learning: still a lot of challenges
- interpretation: which variables are chosen? Why ?
- learning in collaboration with other heuristics, e.g, cut selection, node selection, etc.
- meta-learning to transfer to unseen instances

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

Code: `https://github.com/pg2455/Hybrid-learn2branch`

Slides: `www.pgupta.info/talks`

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

Hybrid Models for Learning to Branch

Thank you!

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