Learning To Cut By Looking Ahead: Cutting Plane Selection via Imitation Learning

Max B. Paulus, Giulia Zarpellon,
Andreas Krause, Laurent Charlin, Chris J. Maddison

Integer Programming

Integer programs are linear programs (LPs) with integral variables:

$$z^{OPT} := \min_x w^\intercal x \quad \text{subject to} \quad Ax \leq b,\ x \in \mathbb{Z} \tag{1}$$

Integer Programming

Integer programs are linear programs (LPs) with integral variables:

$$z^{OPT} := \min_{x} w^{\intercal} x \quad \text{subject to} \quad Ax \leq b,\ x \in \mathbb{Z} \tag{1}$$

Solving integer programs (IPs) is hard, but ubiquitous...

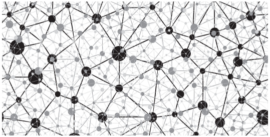
Crew Scheduling

Power Production

Vehicle Routing

Server Load Balancing

Portfolio Optimization



Neural Network Verification

Integer Programming

Integer programs are linear programs (LPs) with integral variables:

$$z^{OPT} := \min_x w^\intercal x \quad \text{subject to} \quad Ax \leq b,\ x \in \mathbb{Z} \tag{1}$$

Solving integer programs (IPs) is hard, but ubiquitous...

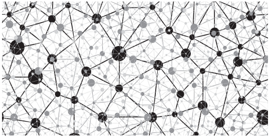
Crew Scheduling

Power Production

Vehicle Routing

Server Load Balancing

Portfolio Optimization



Neural Network Verification

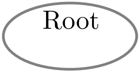
Why ML? $\rightarrow$ IPs are *many* and *similar* in most applications.

Branch and Bound Algorithm

IPs can be solved with branch and bound (B&B):

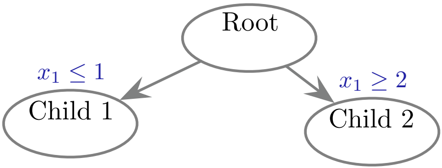
Branch and Bound Algorithm

IPs can be solved with branch and bound (B&B):



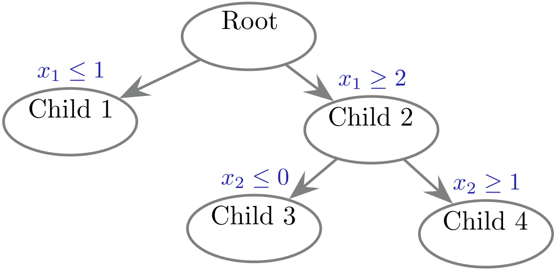
Branch and Bound Algorithm

IPs can be solved with branch and bound (B&B):



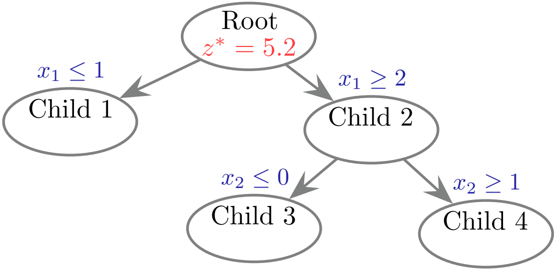
Branch and Bound Algorithm

IPs can be solved with branch and bound (B&B):



Branch and Bound Algorithm

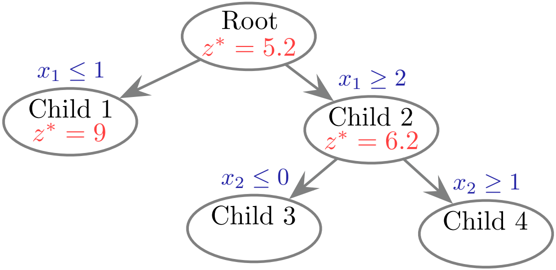
IPs can be solved with branch and bound (B&B):



global lower bound = 5.2

Branch and Bound Algorithm

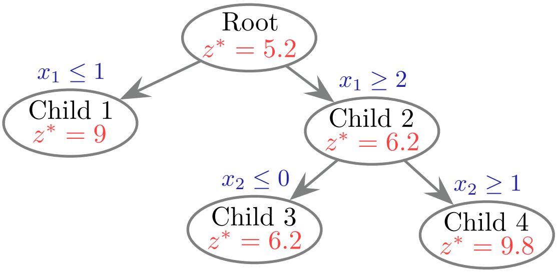
IPs can be solved with branch and bound (B&B):



global lower bound = 6.2

Branch and Bound Algorithm

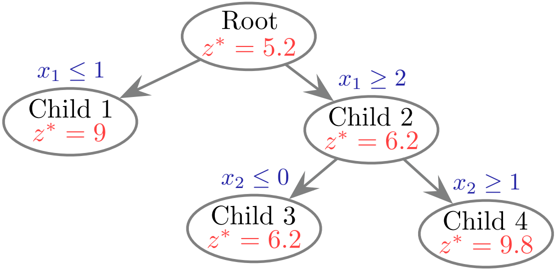
IPs can be solved with branch and bound (B&B):



global lower bound $= 6.2$

Branch and Bound Algorithm

IPs can be solved with branch and bound (B&B):

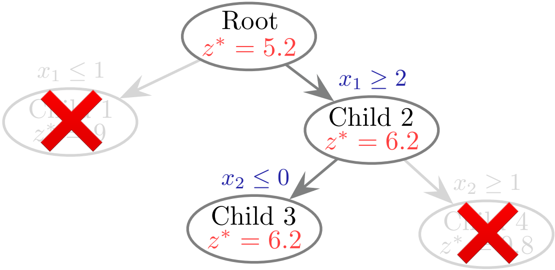


heur. upper bound $= 8$

global lower bound $= 6.2$

Branch and Bound Algorithm

IPs can be solved with branch and bound (B&B):



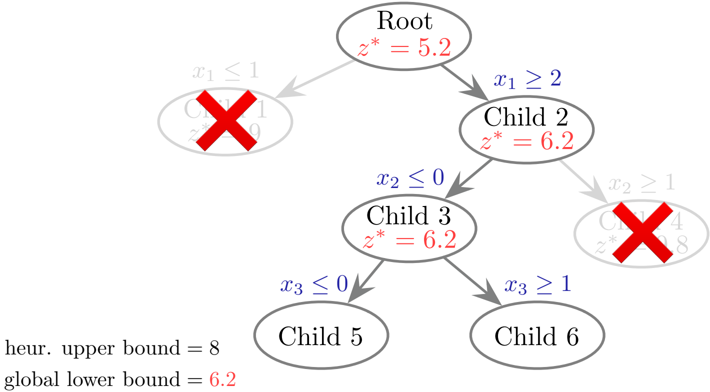
heur. upper bound $= 8$

global lower bound $= 6.2$

B&B uses the LP relax. $z^* = \min_x w^\intercal x$ s.t. $Ax \leq b$ (incl. branch cons.) to locally lower bound the objective and prune search space.

Branch and Bound Algorithm

IPs can be solved with branch and bound (B&B):



B&B uses the LP relax. $z^* = \min_x w^\intercal x$ s.t. $Ax \leq b$ (incl. branch cons.) to locally lower bound the objective and prune search space.

Branch and Cut Algorithm (B&C)

B&C tightens the lower bound by adding a cut $\boldsymbol{C = (c, c_0)}$,

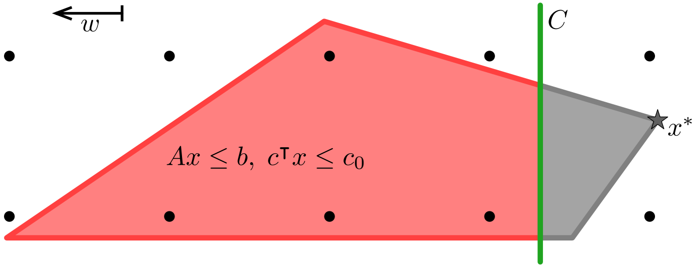
$$z_C \coloneqq \min_x w^\intercal x \quad \text{subject to} \quad Ax \le b,\ \boldsymbol{c^\intercal x \le c_0}. \tag{2}$$

Branch and Cut Algorithm (B&C)

B&C tightens the lower bound by adding a cut $\boldsymbol{C = (c, c_0)}$,

$$z_C := \min_x w^\intercal x \quad \text{subject to} \quad Ax \leq b,\ \boldsymbol{c^\intercal x \leq c_0}. \tag{2}$$

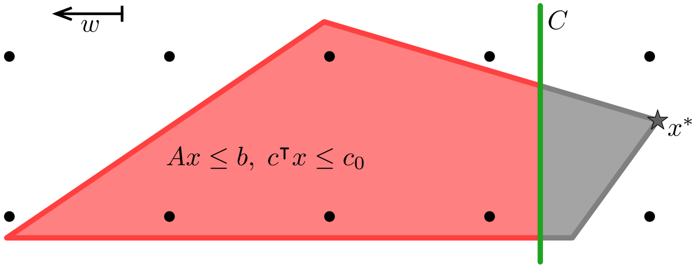
A **cut** $\boldsymbol{C}$ is an extra constraint satisfied by all feasible x of the IP...



..but not by $x^* := \arg\min_x w^\intercal x$ s.t. $\boldsymbol{Ax \leq b}$ (LP relaxation).

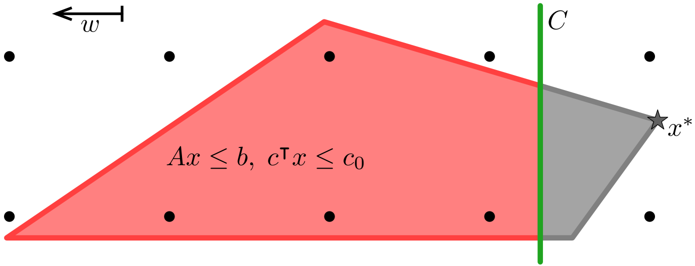
Cutting Planes

In practice, many cuts $C \in \mathcal{C}$ can be generated...



Cutting Planes

In practice, many cuts $C \in \mathcal{C}$ can be generated...



..but only *effective* cuts should be added, because every cut increases the complexity of resolving the linear program (2).

Our Contribution: "Learning To Cut By Looking Ahead"

To date, modern B&C solvers use *manual heuristics* to select cuts.

Our Contribution: "Learning To Cut By Looking Ahead"

To date, modern B&C solvers use *manual heuristics* to select cuts.

Instead, we leverage *machine learning* for cutting plane selection:

Our Contribution: "Learning To Cut By Looking Ahead"

To date, modern B&C solvers use *manual heuristics* to select cuts.

Instead, we leverage *machine learning* for cutting plane selection:

- **Looking Ahead:** Propose expensive but effective *lookahead* criterion for cut selection, used as label for supervised training.

Our Contribution: "Learning To Cut By Looking Ahead"

To date, modern B&C solvers use *manual heuristics* to select cuts.

Instead, we leverage *machine learning* for cutting plane selection:

- **Looking Ahead:** Propose expensive but effective *lookahead* criterion for cut selection, used as label for supervised training.
- **Learning To Cut:** Propose *NeuralCut*, a graph neural network (GNN) to *imitate* lookahead for cut selection.

Looking Ahead: Lookahead Criterion is strong but expensive

For a cut $C \in \mathcal{C}$, we define the lookahead score s_{LA} as...

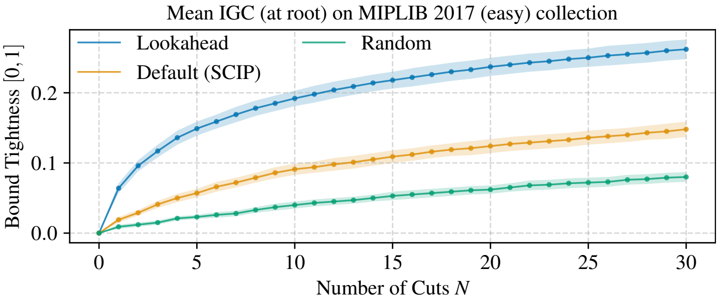
$$s_{LA}(C) := z_C - z^* \geq 0 \tag{3}$$

Looking Ahead: Lookahead Criterion is strong but expensive

For a cut $C \in \mathcal{C}$, we define the lookahead score s_{LA} as...

$$s_{LA}(C) := z_C - z^* \geq 0 \tag{3}$$

Selecting $C^*_{LA} = \arg\max_{C \in \mathcal{C}} s_{LA}(C)$ is effective on a benchmark...

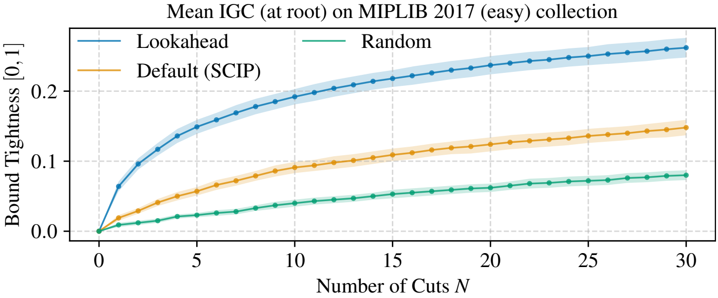


Looking Ahead: Lookahead Criterion is strong but expensive

For a cut $C \in \mathcal{C}$, we define the lookahead score s_{LA} as...

$$s_{LA}(C) := z_C - z^* \geq 0 \tag{3}$$

Selecting $C^*_{LA} = \arg\max_{C \in \mathcal{C}} s_{LA}(C)$ is effective on a benchmark...



..but computing $s_{LA}(C)$ for all $C \in \mathcal{C}$ is too expensive in practice.

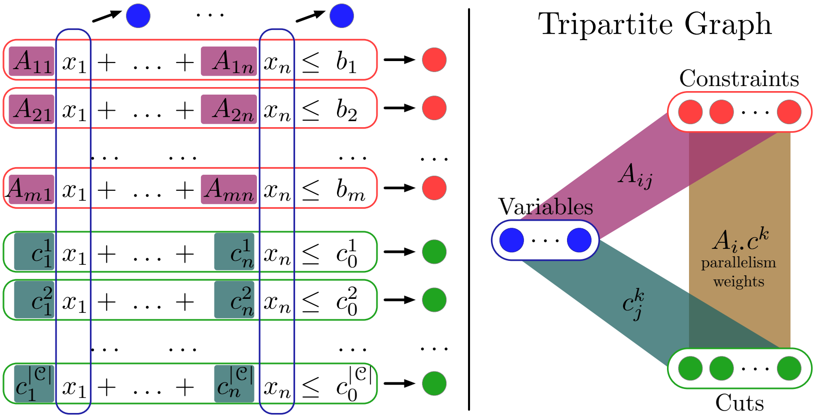
Learning To Cut: NeuralCut to learn lookahead selection

Hence, we **learn to imitate** lookahead selection with *NeuralCut*.

Learning To Cut: NeuralCut to learn lookahead selection

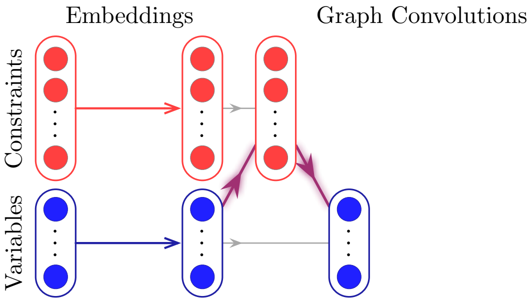
Hence, we **learn to imitate** lookahead selection with *NeuralCut*.

To learn from IPs and cutpools, Neuralcut uses a tripartite graph...



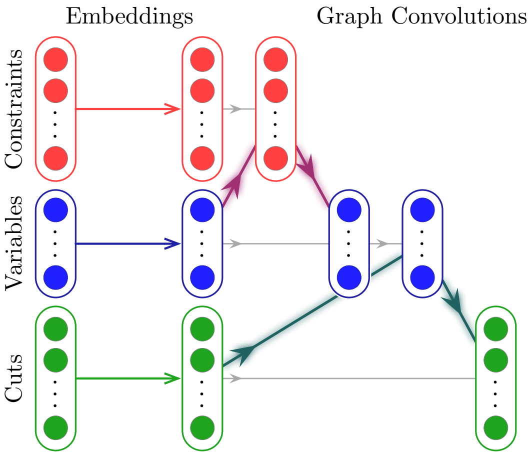
Learning to Cut: NeuralCut is a graph neural network

NeuralCut is a GNN based on Gasse et al. (2019)...



Learning to Cut: NeuralCut is a graph neural network

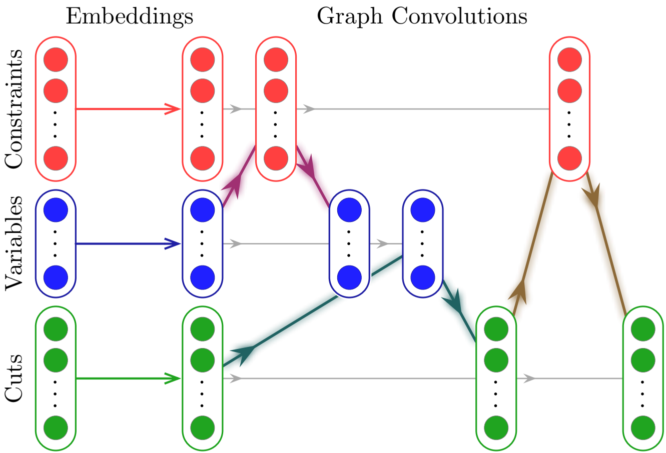
NeuralCut is a GNN based on Gasse et al. (2019)...



..but adds more convolutions,

Learning to Cut: NeuralCut is a graph neural network

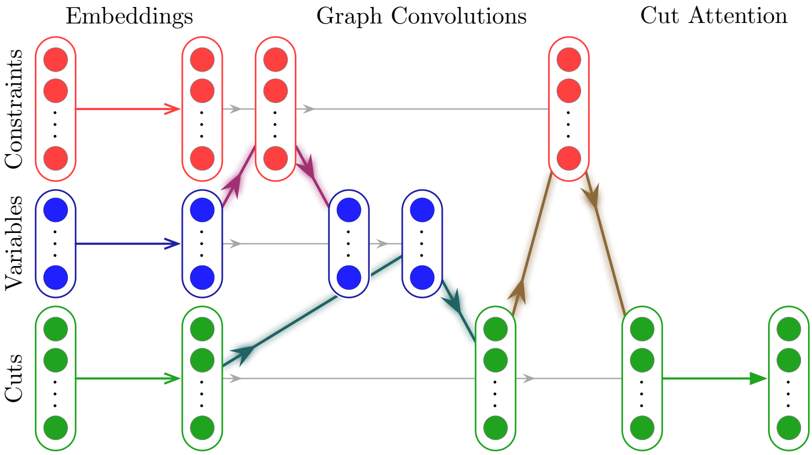
NeuralCut is a GNN based on Gasse et al. (2019)...



..but adds more convolutions, shortcuts

Learning to Cut: NeuralCut is a graph neural network

NeuralCut is a GNN based on Gasse et al. (2019)...



..but adds more convolutions, shortcuts and self-attention on cuts.

Learning To Cut: NeuralCut to learn lookahead selection

NeuralCut jointly predicts a score $\hat{\boldsymbol{s}}_{\boldsymbol{C}}$ for each cut $C \in \mathcal{C}$..

Learning To Cut: NeuralCut to learn lookahead selection

NeuralCut jointly predicts a score $\hat{s}_C$ for each cut $C \in \mathcal{C}$..

- to minimize a training log-loss **over all cuts**...

$$L(\hat{s}) \coloneqq -\frac{1}{|\mathfrak{C}|} \sum_{C \in \mathfrak{C}} q_C \log \hat{s}_C + (1 - q_C) \log(1 - \hat{s}_C) \quad (4)$$

..with soft targets $q_C = \frac{s_{LA}(C)}{s_{LA}(C^*_{LA})}$.

Learning To Cut: NeuralCut to learn lookahead selection

NeuralCut jointly predicts a score $\hat{s}_C$ for each cut $C \in \mathcal{C}$..

- to minimize a training log-loss **over all cuts**...

$$L(\hat{s}) \coloneqq -\frac{1}{|\mathfrak{C}|} \sum_{C \in \mathfrak{C}} q_C \log \hat{s}_C + (1 - q_C) \log(1 - \hat{s}_C) \quad (4)$$

..with soft targets $q_C = \frac{s_{LA}(C)}{s_{LA}(C^*_{LA})}$.

- and selects the cut $C^*_{NC} = \arg\max_{C \in \mathfrak{C}} \hat{s}_C$ at test time.

Experimental Overview

We evaluate NeuralCut on a diverse range of IP families...

Experimental Overview

We evaluate NeuralCut on a diverse range of IP families...

- ..on **pure cut selection**..

  ..to select cuts that achieve *tight* bounds *quickly*

  ($\rightarrow$ small area above bound tightness curve).

Experimental Overview

We evaluate NeuralCut on a diverse range of IP families...

- ..on **pure cut selection**..

  ..to select cuts that achieve *tight* bounds *quickly*

  ($\rightarrow$ small area above bound tightness curve).

- ..as a **plug-in of the SCIP[1] B&C solver**...

  ..to select *effective* cuts at root that speed up B&C search

  ($\rightarrow$ small *residual* solving time after processing root).

[1] Gamrath et al., 2020

Experiments: NeuralCut for *pure* cut selection

NeuralCut achieves **tight bounds quickly** on various IP families...

Area above bound tightness curve (↓) on test instances, mean (ste), $N = 30$

		Max. Cut	Packing	Bin. Packing	Planning
	Lookahead	*15.05 (0.09)*	*26.42 (0.07)*	*9.85 (0.33)*	*10.33 (0.04)*
ML	**NeuralCut**	**15.55 (0.09)**	***26.30 (0.08)***	**10.96 (0.33)**	**10.42 (0.04)**

[2] Tang et al., ICML 2020

Experiments: NeuralCut for *pure* cut selection

NeuralCut achieves **tight bounds quickly** on various IP families...

Area above bound tightness curve (↓) on test instances, mean (ste), $N = 30$

		Max. Cut	Packing	Bin. Packing	Planning
	Lookahead	*15.05 (0.09)*	*26.42 (0.07)*	*9.85 (0.33)*	*10.33 (0.04)*
ML	**NeuralCut**	**15.55 (0.09)**	***26.30 (0.08)***	**10.96 (0.33)**	**10.42 (0.04)**
ML	RL[2]	19.00 (0.09)	27.59 (0.06)	16.06 (0.38)	14.94 (0.08)
Heuristics	Default	16.72 (0.09)	***26.29 (0.08)***	15.42 (0.28)	14.01 (0.06)
Heuristics	Efficacy	17.01 (0.09)	***26.28 (0.08)***	15.19 (0.29)	14.52 (0.06)
Heuristics	Obj. Parallelism	24.01 (0.08)	28.28 (0.05)	22.23 (0.26)	27.71 (0.07)
Heuristics	IntSupport	21.87 (0.07)	28.78 (0.03)	21.14 (0.28)	23.31 (0.08)
Heuristics	Random	21.99 (0.07)	28.73 (0.04)	21.23 (0.28)	23.26 (0.08)

..and **outperforms** heuristic scores and a competing RL approach.

[2] Tang et al., ICML 2020

Experiments: NeuralCut inside a B&C solver

Paired with an ϵ-threshold rule to run inside the SCIP B&C solver on (mixed) integer programs from neural network verification...

Experiments: NeuralCut inside a B&C solver

Paired with an ϵ-threshold rule to run inside the SCIP B&C solver on (mixed) integer programs from neural network verification...

Median performance on test instances

	# cuts	Rel. bound improv.	Residual Time (s)
Default (SCIP B&C)	279	**1.00**	23.65
NeuralCut ($\epsilon = 10^{-5}$)	105	**1.00**	22.35
NeuralCut ($\epsilon = 10^{-4}$)	81	0.99	**20.89**
NeuralCut ($\epsilon = 10^{-3}$)	48	0.98	22.73

...NeuralCut achieves **strong lower bounds at root** with fewer cuts and **speeds up** the remaining B&C search.

Conclusion

We introduced **NeuralCut**, our approach

Conclusion

We introduced **NeuralCut**, our approach

- proposes the expensive **lookahead criterion** and shows it selects strong cuts,

Conclusion

We introduced **NeuralCut**, our approach

- proposes the expensive **lookahead criterion** and shows it selects strong cuts,
- demonstrates the **effectiveness of imitation learning** for cut selection on various IP benchmarks,

Conclusion

We introduced **NeuralCut**, our approach

- proposes the expensive **lookahead criterion** and shows it selects strong cuts,
- demonstrates the **effectiveness of imitation learning** for cut selection on various IP benchmarks,
- is **deeply integrated** into the SCIP solver (cut selection, separation, plug-in).

Conclusion

We introduced **NeuralCut**, our approach

- proposes the expensive **lookahead criterion** and shows it selects strong cuts,
- demonstrates the **effectiveness of imitation learning** for cut selection on various IP benchmarks,
- is **deeply integrated** into the SCIP solver (cut selection, separation, plug-in).

Code: We plan to make code available at `github.com/mbp28`.