DeepMind

Solving Mixed Integer Programs Using Neural Networks

https://arxiv.org/pdf/2012.13349.pdf

Vinod Nair
vinair@google.com
DeepMind

24th February 2021

Collaborators

Sergey Bartunov*[1], Felix Gimeno*[1], Ingrid von Glehn*[1], Pawel Lichocki*[2], Ivan Lobov*[1], Brendan O'Donoghue*[1], Nicolas Sonnerat*[1], Christian Tjandraatmadja*[2], Pengming Wang*[1],

Ravichandra Addanki[1] , Tharindi Hapuarachchi[1], Thomas Keck[1], James Keeling[1] , Pushmeet Kohli[1] , Ira Ktena[1] , Yujia Li[1] , Oriol Vinyals[1] , Yori Zwols[1]

*Equal contributors
[1]DeepMind
[2]Google Research

Mixed Integer Programming (MIP)

- Integer programming: One of Karp’s 21 NP-complete problems

$$\min_{x} c^T x$$

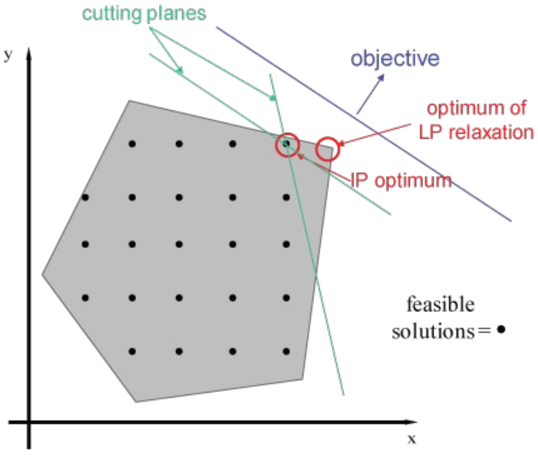
Objective function

$$s.t. \quad Ax \leq b$$

Linear constraints

$$x_i \in \mathbb{Z} \quad \forall i$$

Integrality constraint

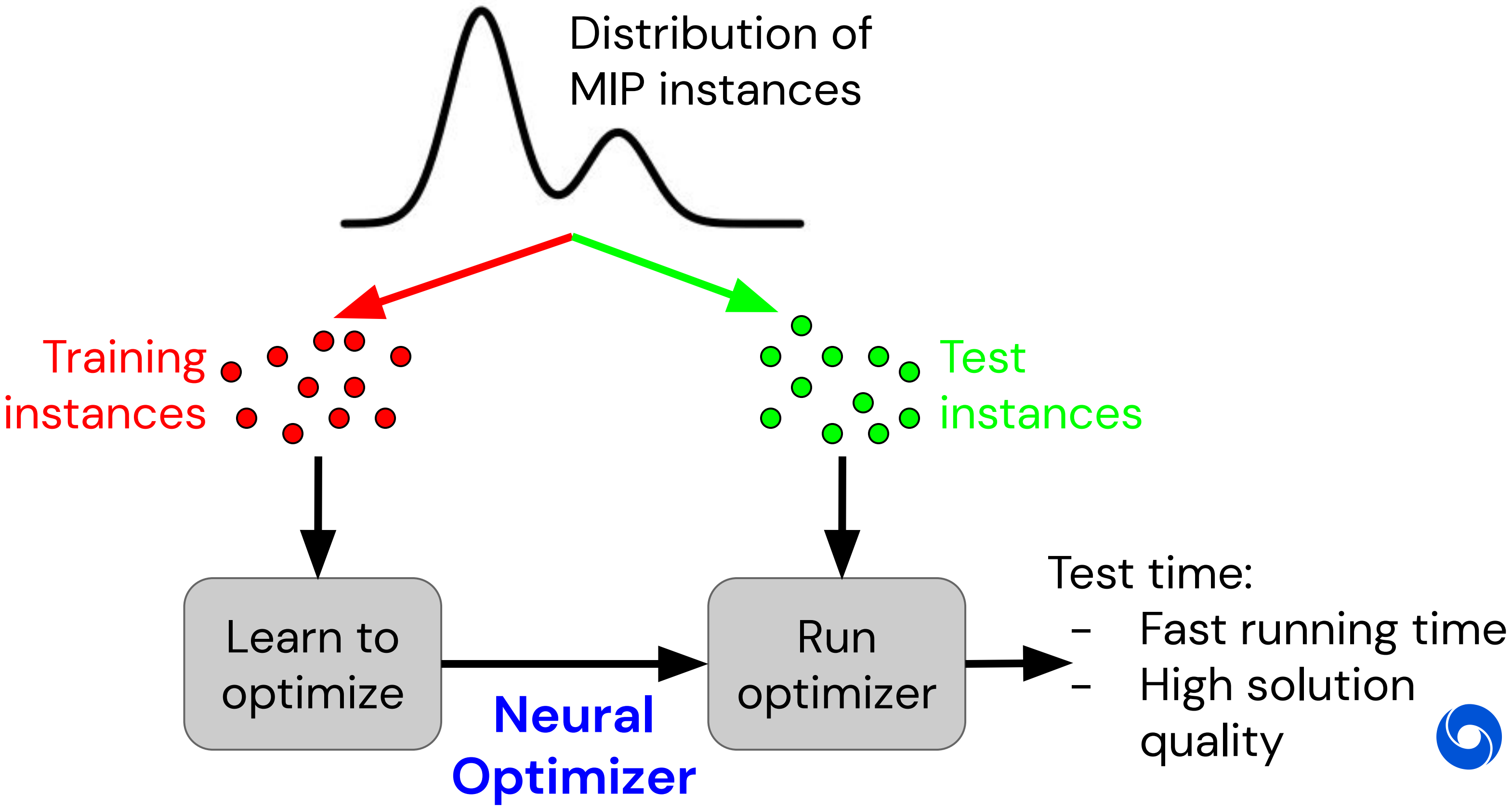


Cutting Planes

- “Mixed” → x can also contain continuous variables
- Many real-world applications!

Why Learning?

- Exploit distribution-specific structure to construct better optimizers

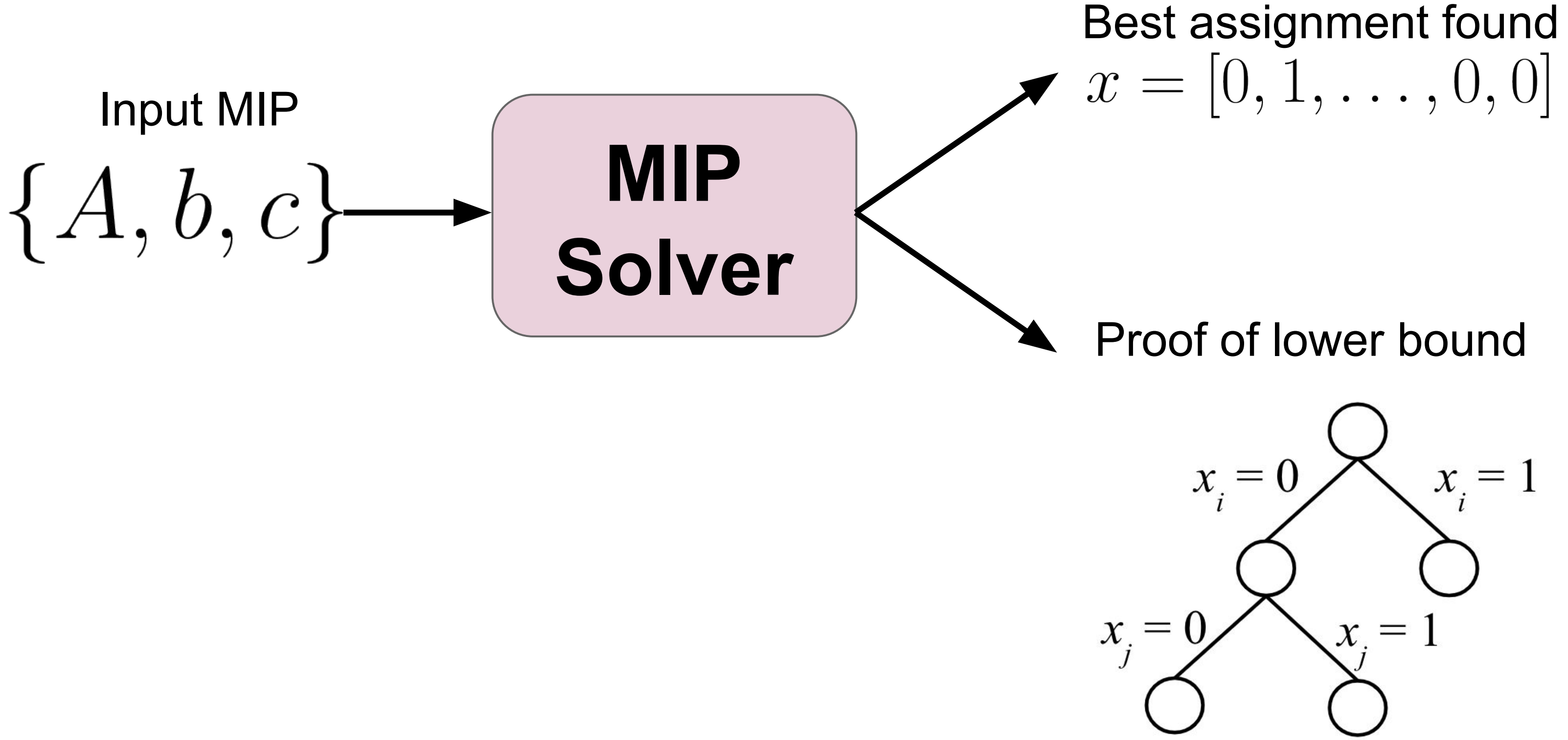


Related Work

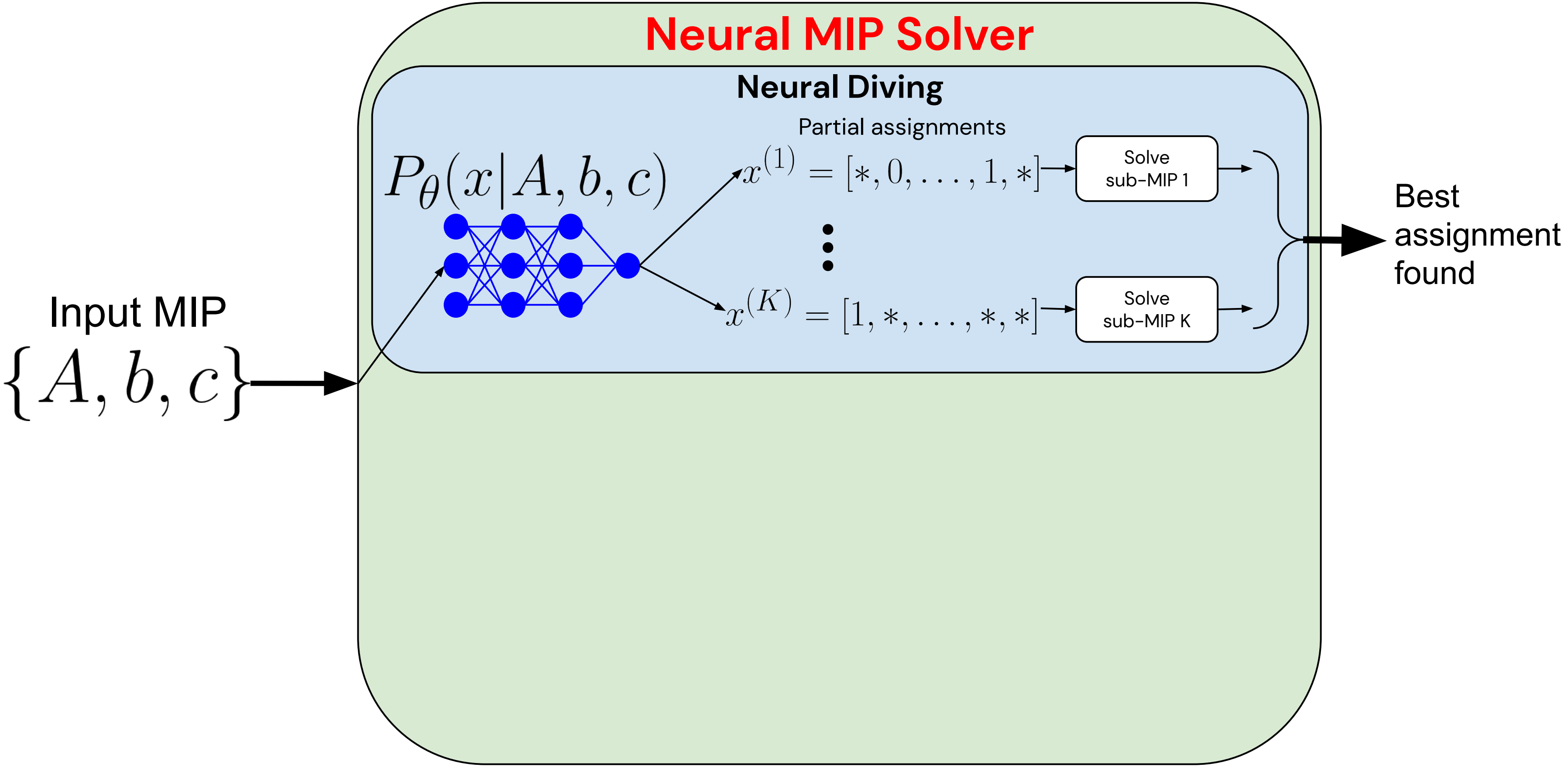
- Lots of work on learning for MIPs!

- **Learning primal heuristics:** Khalil et al., 2017, Hendel, 2018, Ding et al., 2020, Xavier et al., 2020, Addanki et al., 2020, Hottung and Tierney, 2020, Song et al., 2020, ...
- **Learning branching policies:** Khalil et al., 2016, Alvarez et al., 2017, Gasse et al., 2019, Zarpellon et al., 2020, Gupta et al., 2020, ...
- **Learning to cut:** Tang et al., 2020.
- **Learning to configure MIP solvers:** Hutter et al., 2011, Hutter et al., 2014, ...

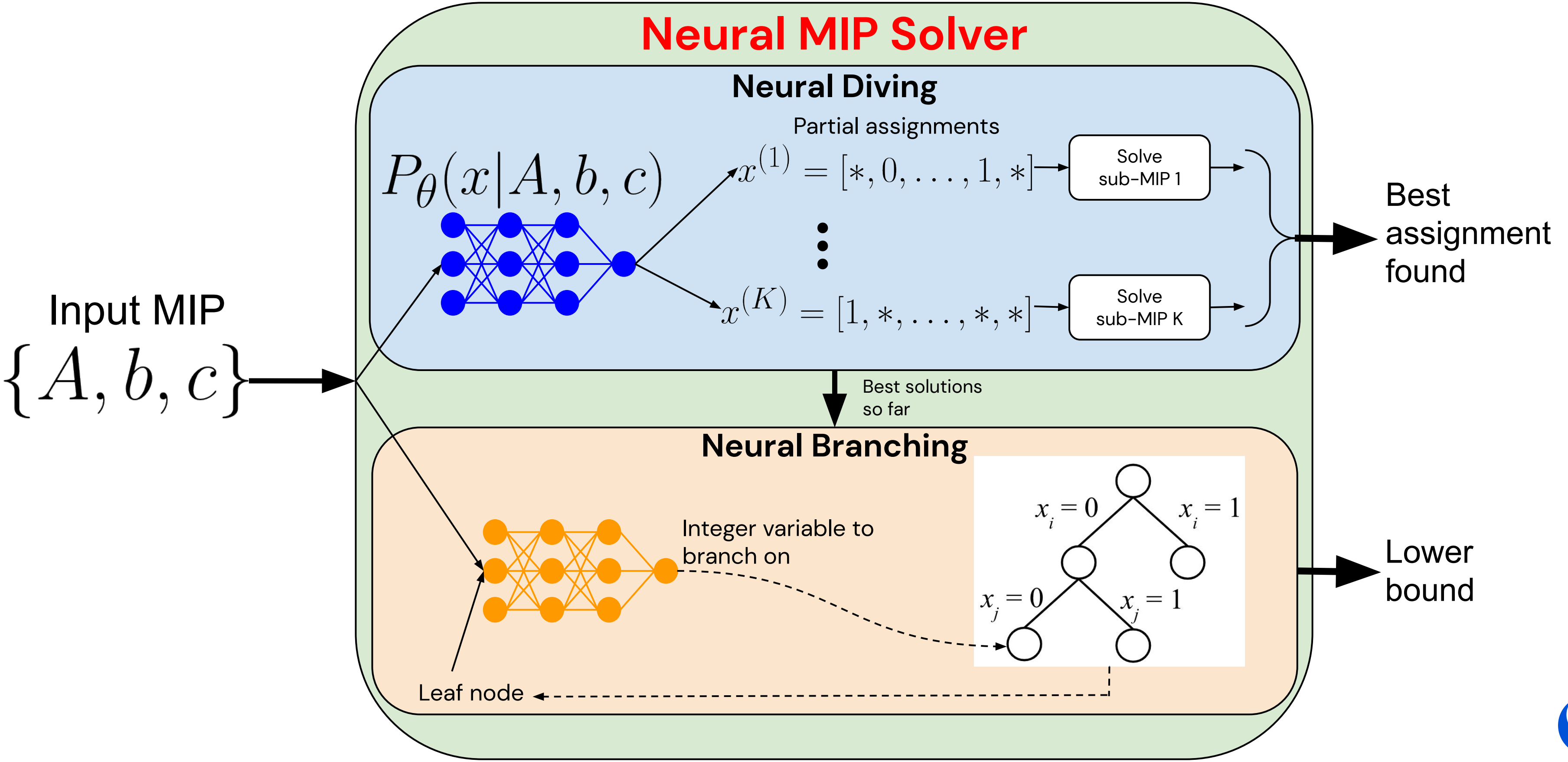
Solving a MIP



Our Approach



Our Approach



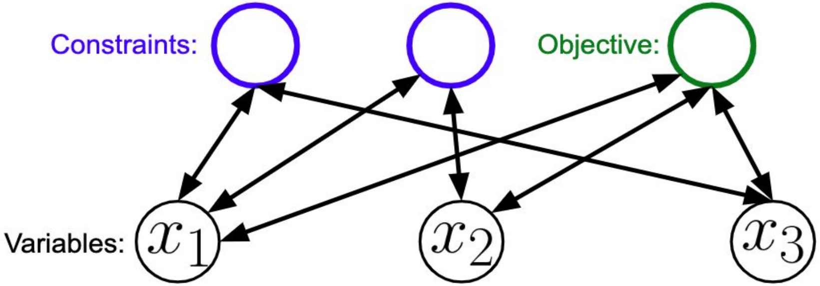
Graph Representation of a MIP

- Convert a mixed integer program into a bipartite graph
- Use GraphNets for learning
  - Handles permutation invariance and variable-sized instances

Mixed Integer Program

$$
\begin{aligned}
& \min_{x} d_1x_1 + d_2x_2 + d_3x_3 \\
s.t. \quad & A_{11}x_1 + A_{13}x_{13} \leq b_1 \\
& A_{21}x_1 + A_{22}x_2 \leq b_2 \\
& x \in \mathbb{Z}
\end{aligned}
$$

Graph representation



Neural Diving

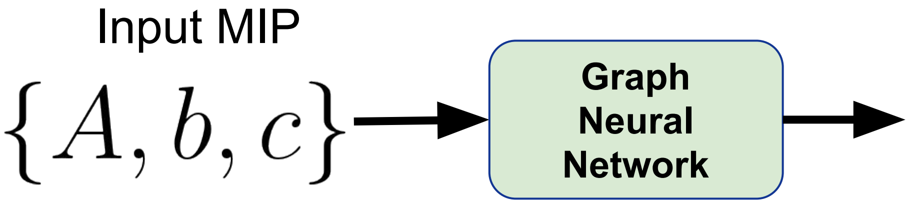
- **Key idea:** Learn a generative model of feasible assignments of discrete variables x given a MIP $G = \{A, b, c\}$

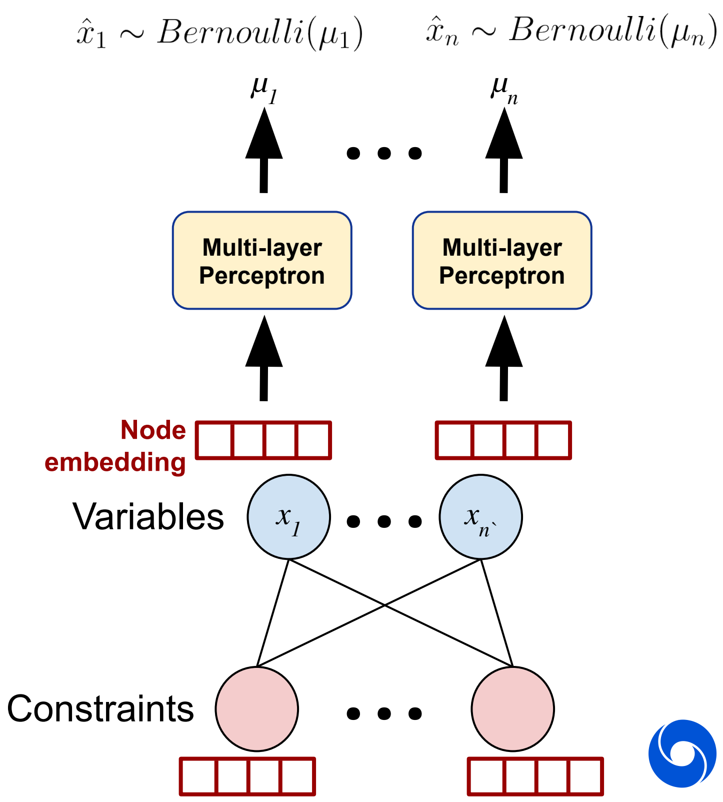
$$P(x|G) = \begin{cases} \dfrac{\exp(-c^T x)}{Z(G)} & \text{if } x \text{ is feasible} \\ 0 & \text{otherwise} \end{cases}$$

- Use samples from the generative model to define partial assignments
    - Naturally lends itself to parallelization

Generative Model

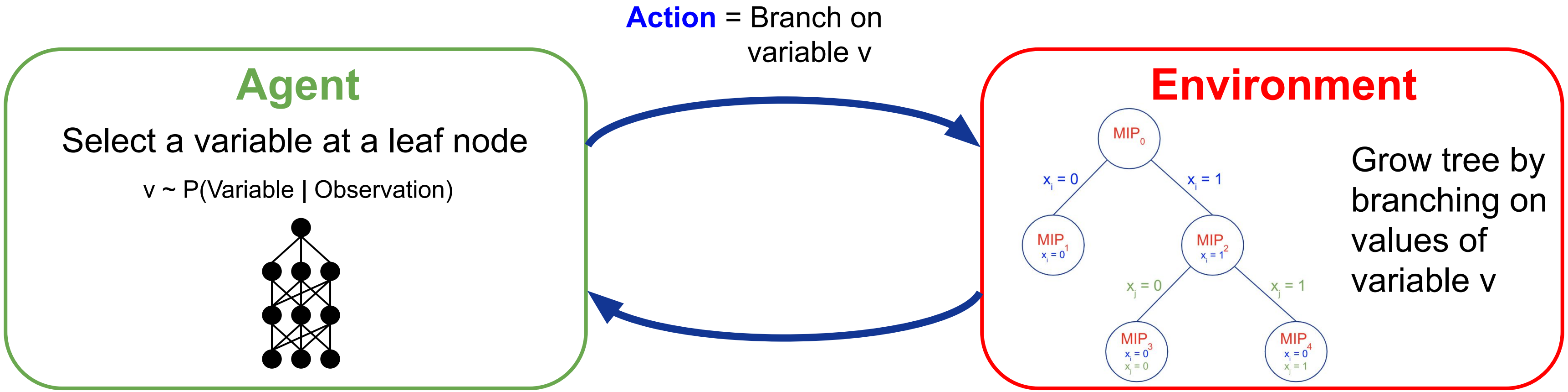
- Train the model on (MIP, assignment) pairs generated using an existing solver on a training set of MIPs.





Neural Branching

- Branching as a sequential decision problem



Observation = Summary of node to branch on

Reward = Progress towards target optimality gap

- Environment is built on SCIP, the SOTA non-commercial solver

Imitation Learning of Branching Policy

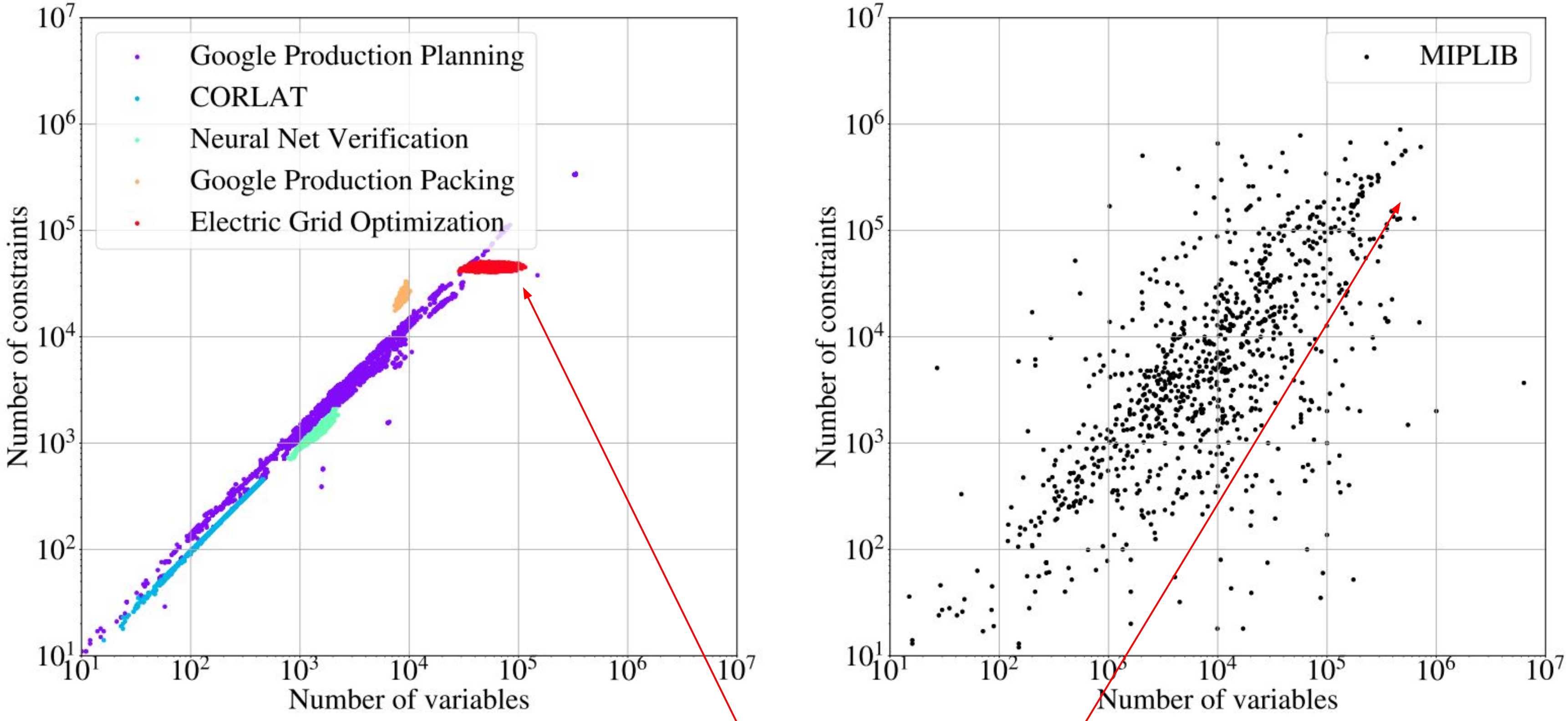
- Learn to imitate an expert policy
  - *Fullstrong branching* is a classical expert from optimization literature
  - We propose a scalable version of Fullstrong branching that uses GPUs
    - Based on Alternating Direction Method of Multipliers (ADMM)

- Imitation learning algorithms
  - Behavioral Cloning
  - Dataset Aggregation (Dagger), Ross et al., AISTATS, 2011.

Dataset	Domain
Neural Network Verification	Verifying a convnet on MNIST images
Electric Grid Optimization	Planning daily operations of a US East Coast regional grid
Google Production Packing	Production packing problem for data centers
Google Production Planning	Production planning problem for data centers
MIPLIB	Public benchmark containing many different applications

Application-specific datasets

Distribution of MIP Sizes (After Presolve)



Large-scale instances!

Evaluation

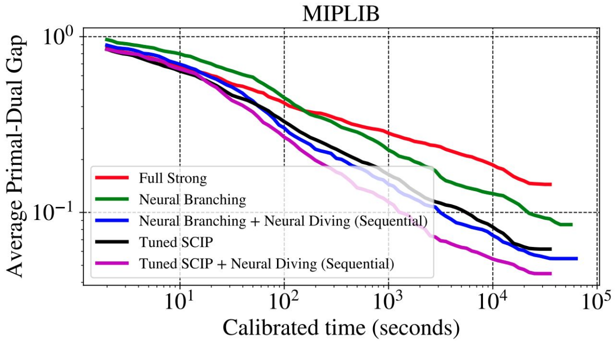
- Use learned policy to solve *unseen* MIP instances

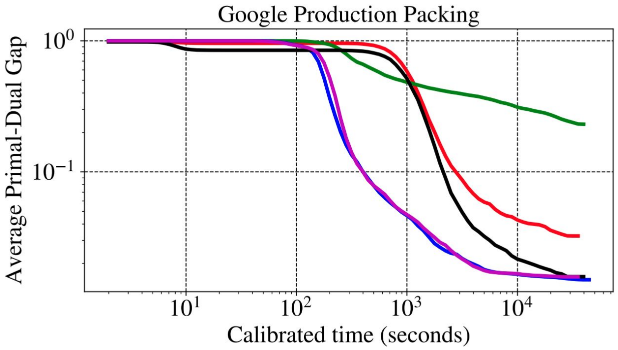
- **Metric:** Average primal–dual gap vs. time

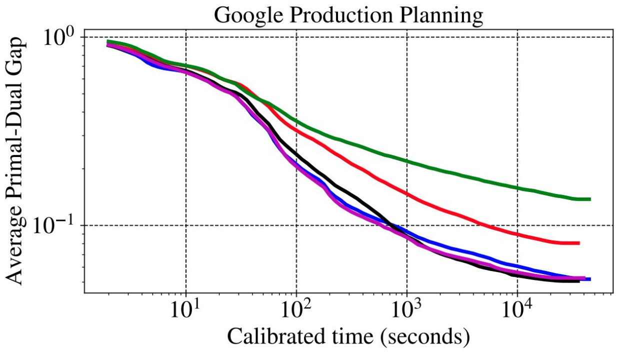
- **Baseline: Tuned SCIP**
  - Tune SCIP's hyperparameters on each dataset
  - Run two SCIP instances with different seeds, use the best primal–dual bound pair to compute the gap

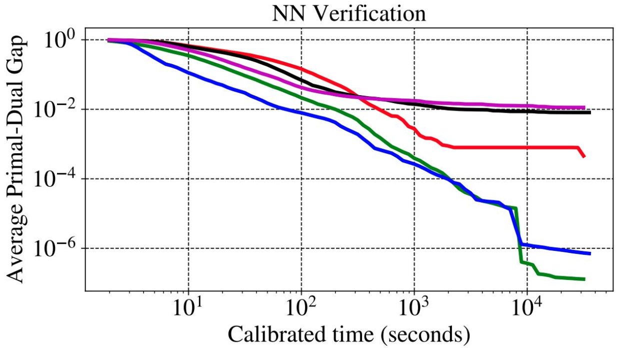
- **Neural solvers**
  - Neural Diving + Neural Branching
  - Neural Branching or Neural Diving alone

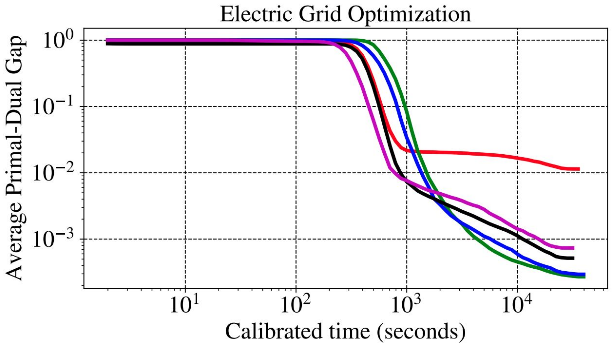
Results


MIPLIB
Average Primal-Dual Gap
Calibrated time (seconds)
Full Strong
Neural Branching
Neural Branching + Neural Diving (Sequential)
Tuned SCIP
Tuned SCIP + Neural Diving (Sequential)


Google Production Packing
Average Primal-Dual Gap
Calibrated time (seconds)

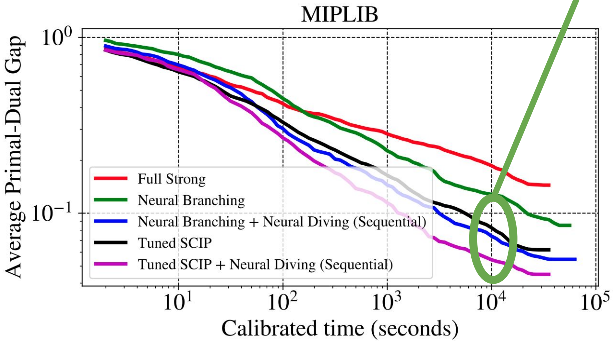

Google Production Planning
Average Primal-Dual Gap
Calibrated time (seconds)

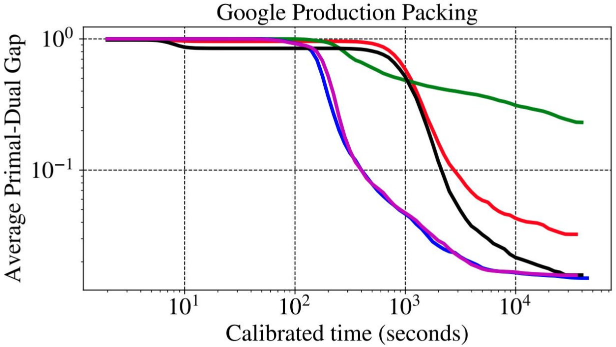

NN Verification
Average Primal-Dual Gap
Calibrated time (seconds)

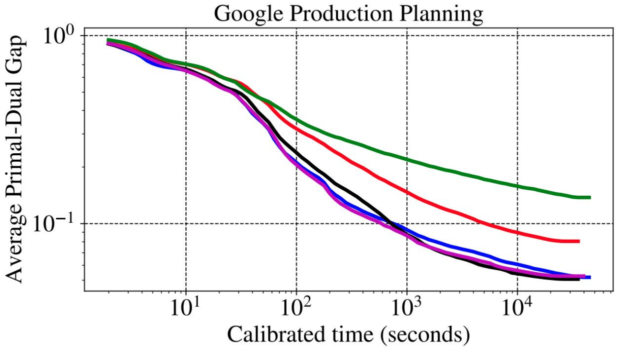

Electric Grid Optimization
Average Primal-Dual Gap
Calibrated time (seconds)

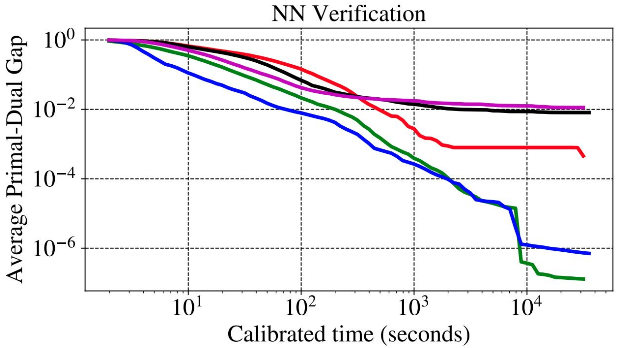
Results

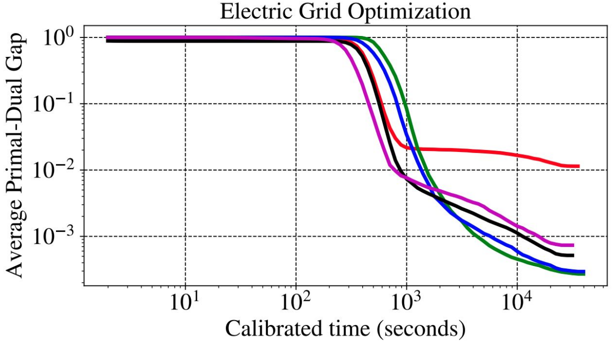
1.5x better gap than Tuned SCIP







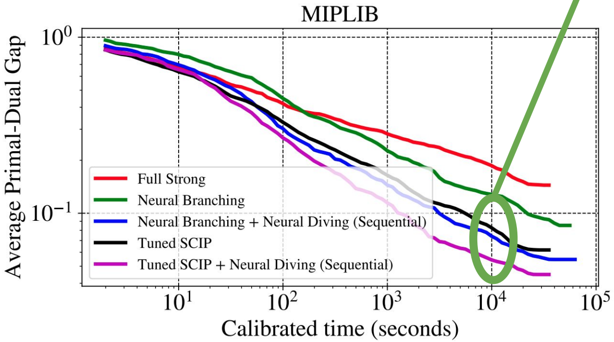


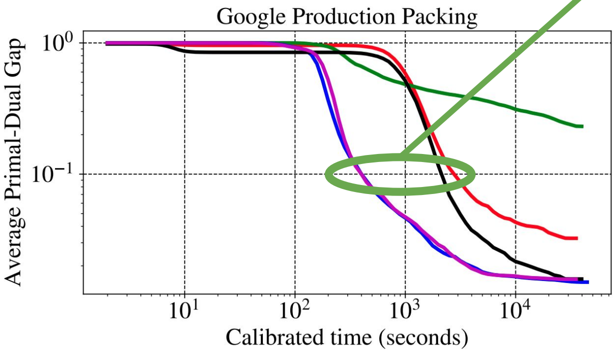


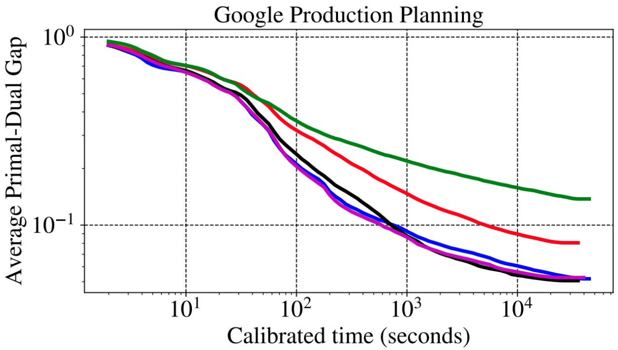
Results

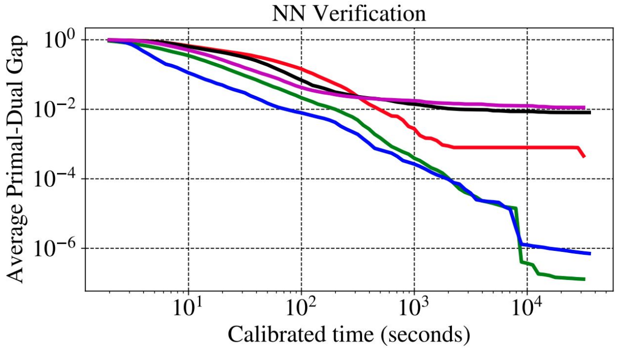
1.5x better gap than Tuned SCIP

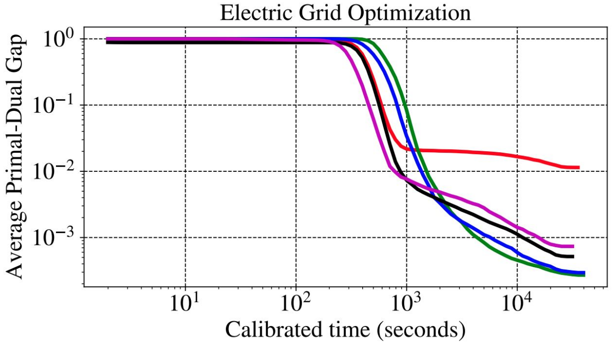
Reaches 10% gap 5x faster than Tuned SCIP







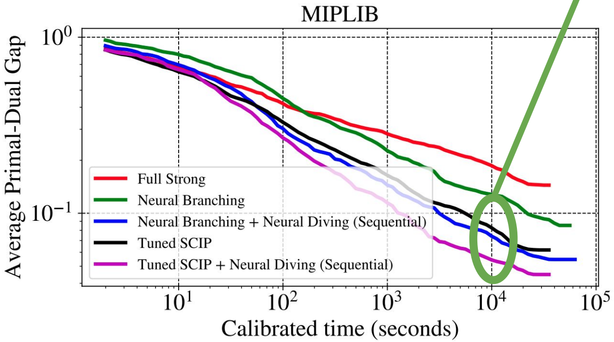


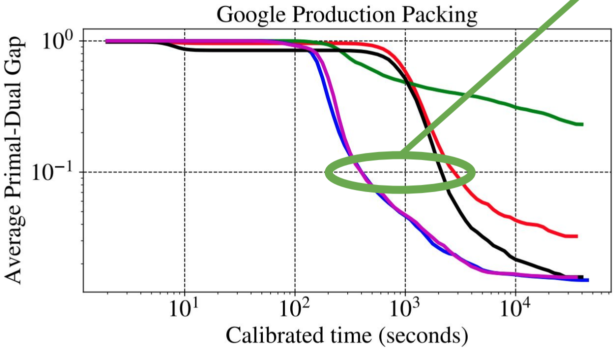


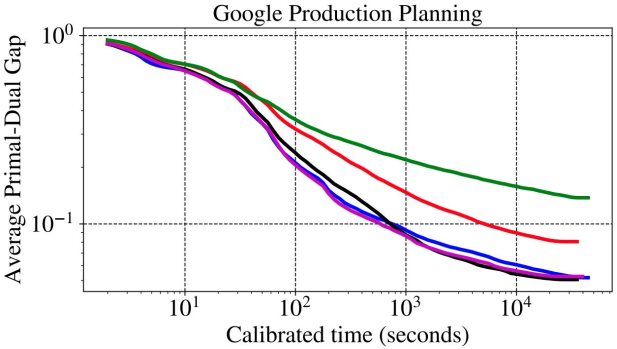
Results

1.5x better gap than Tuned SCIP

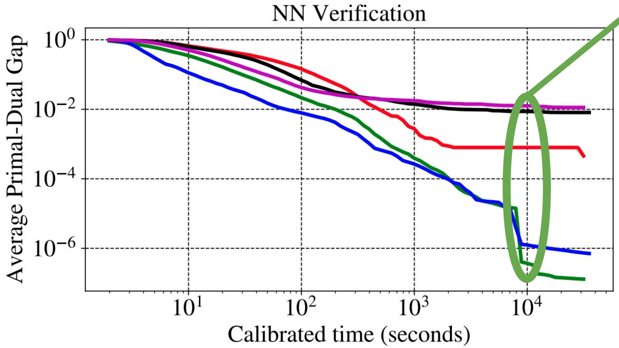
Reaches 10% gap 5x faster than Tuned SCIP

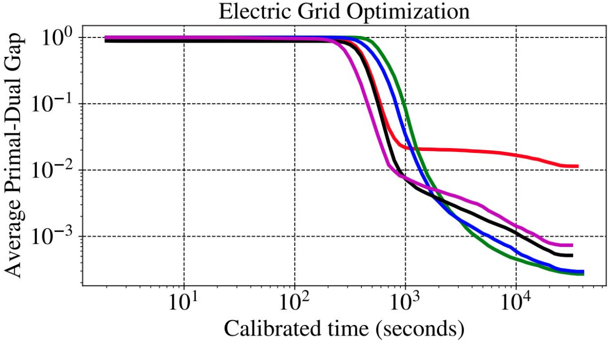






> 10^4x better gap than Tuned SCIP





Surprising Result

- Applied Neural Diving to singleton MIPs in MIPLIB
  - Create smaller MIPs from singleton MIP → Train → Apply model to singleton MIP.

- Achieves best ever objective value on three of the open MIPs!

MIP Name	New < previous best objective value (lower = better)	Previous best solver
milo-v12-6-r1-75-1	**1153756.398** < 1153880	CPLEX, Dec 2019
neos-1420790	**3121.29** < 3121.42	CPLEX, Dec 2019
xmas10-2	**-497** < -495	Gurobi 9.0, Feb 2020

Conclusions and Next Steps

- **First demonstration of learning beating SCIP on large-scale, real-world datasets!**
  - Learning is effective even on MIPLIB!

- Next: Better learned primal heuristics
  - Combine with classical techniques, e.g., domain propagation, iterative LP solving, ...
  - Neural Large Neighborhood Search (Addanki et al., LMCA Workshop at NeurIPS 2020): Iteratively improve initial assignment produced by Neural Diving.

DeepMind

Thanks!