Parameterizing B&B search trees to learn branching policies

Giulia Zarpellon, Jason Jo, Andrea Lodi and Yoshua Bengio

SCIP Online Workshop · June 4, 2020

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

CANADA EXCELLENCE RESEARCH CHAIR
DATA SCIENCE FOR REAL-TIME DECISION-MAKING

IVADO

Mila

B&B variable selection

Branch and Bound (B&B) is the backbone of Mixed-Integer Linear Programming (MILP)

$$min_x \{c^T x : Ax \geq b, x \geq 0, x_i \in \mathbb{Z} \; \forall \; i \; \in \mathcal{I}\}$$

an *exact* **tree-search** method
iteratively solve LP relaxations and
partition the solution space by **branching on variables**

VARIABLE SELECTION: at branching step t, split current node Q_t into subproblems by

- selecting a candidate variable $j \in \mathcal{C}_t := \{i \in \mathcal{J} : x_i^{Q_t} \notin \mathbb{Z}\}$,
- creating new nodes according to the disjunction $x_j \leq \lfloor x_j^{Q_t} \rfloor \vee x_j \geq \lceil x_j^{Q_t} \rceil$

Learning to branch

Complex and flexible MILP solver environment,
several components interact and many crucial decisions are *heuristic*

VARIABLE SELECTION is key for search success

⟶ Perfect ground for **machine learning** (ML) experiments

"Learning to branch" (l2b) established theme, works mostly focus on

imitation of **strong branching** and
specialization of policies to **combinatorial classes**

[Lodi and Z., 2017] survey
[Khalil at al., 2016]
[Balcan et al., 2018]
[Gasse et al., 2019]

⟶ Seek broader generalization, across generic MILPs, no restrictions on structure/size

Branching on heterogeneous MILPs

Hypothesis

The space of B&B search trees can represent the complexity and dynamism of branching, in a way that is shared across heterogeneous MILPs.

Intuitions

- Algorithmic decisions could depend on the state of the search

  MILP phases and algorithmic pattern in B&B process,
  abundant yet (mostly) unexploited data from the search

[Berthold, Hendel and Koch., 2017]
[Fischetti, Lodi and Z., 2019]
[Hendel et al., 2020]

- Search evolution and variable selection are deeply linked

  select a variable based on its role in the search components

Branching on heterogeneous MILPs

- State-of-the-art MILP branching rules are mechanisms to score variables based on their effectiveness in the search

  `relpscost` combines multiple scores from different components in weighted sum,
  `conflict`, `inference`, `cutoff` and `pseudo-cost`

- Importance of different functionalities should **change dynamically** during exploration

  `dynamicfactor` adjusts weights and variables' scores

Idea

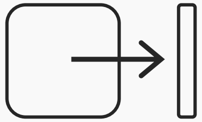
Consider variables' roles and the tree exploration itself to perform a more flexible variable selection, adapted to search stages.

Novel l2b framework

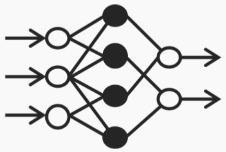
To explore the idea of learning branching policies

from *parameterizations of B&B search trees*

that are *shared among general MILPs*,



Represent branching in the space of B&B trees via **input features**



Combine data via **ML model** for branching predictions

Input features

$C_t \in \mathbb{R}^{25 \times |\mathcal{C}_t|}$

- Represent **set of candidate variables** $\mathcal{C}_t$

    capture multiple roles of a variable throughout the search:
    LP bound and solution, statistics on search participation and past branchings

    scores in `relpscost` formula are included

$Tree_t \in \mathbb{R}^{61}$

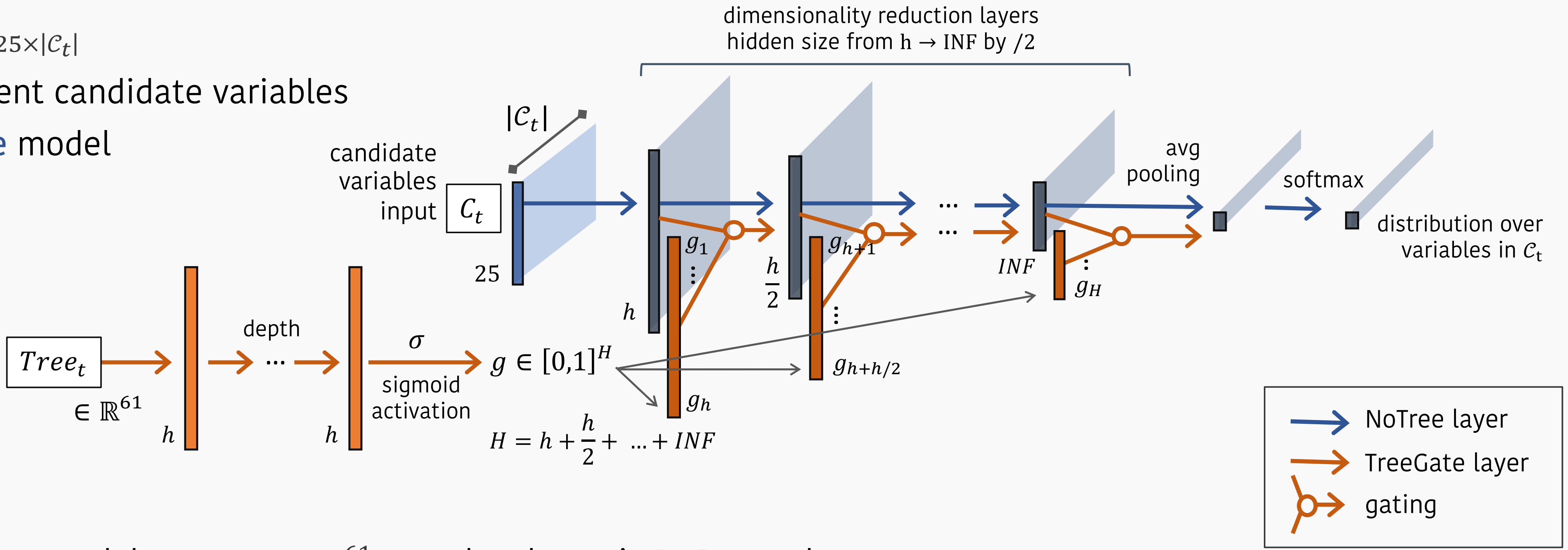
- Encode dynamic **state of B&B search**

    current node (depth, bound), tree composition (nodes explored, open, leaves),
    global bounds evolution, aggregated scores, statistics on open nodes

Architectures

$C_t \in \mathbb{R}^{25 \times |\mathcal{C}_t|}$
represent candidate variables

NoTree model



TreeGate model $\quad Tree_t \in \mathbb{R}^{61}$ encodes dynamic B&B search

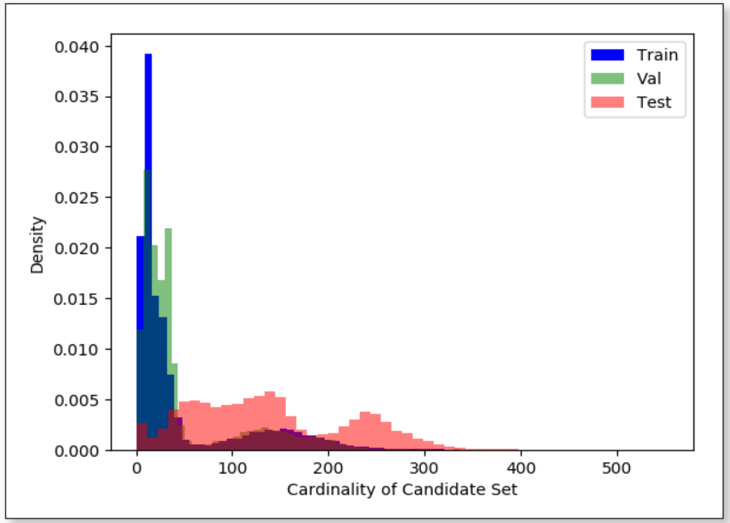
Modulation of variables' representations

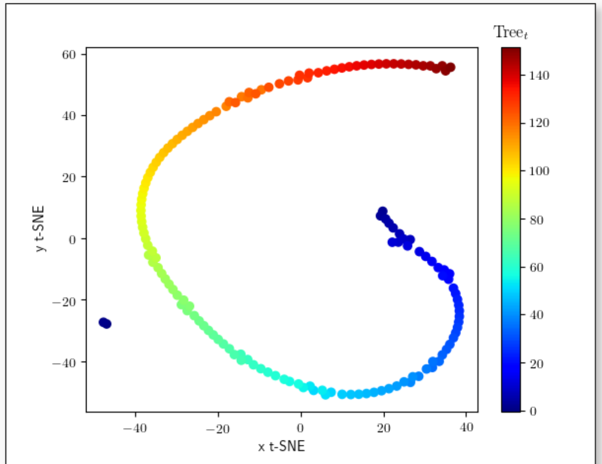
- provides context over branching via learned tree-based signal
- adapts variable selection to tree evolution

Remarks

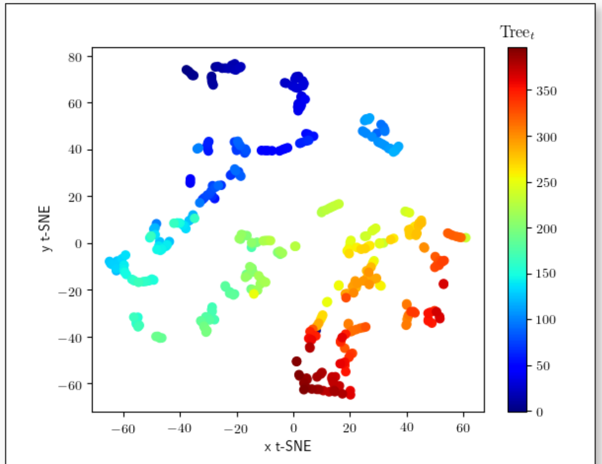
- $\{\mathcal{C}_t, Tree_t\}$ gathered via customized `PySCIPOpt`, do not depend on parameters $(c, A, b, \mathcal{J})$
- $|\mathcal{C}_t|$ **varies wildly** across MILPs and B&B search (vs. fixed structure/size)

  and so does C_t dimensionality: treat $|\mathcal{C}_t|$ as “batch dimension” in our networks
- $Tree_t$ is **not static**, but evolves with the search





eil33-2



seymour1

Experimental setup

- **Curation of MILP dataset**

  27 heterogeneous instances from MIPLIB 3, 2010, 2017 and MILPLib
  To better explore generalization abilities, focus on manageable trees

- **Data collection** (offline) for **imitation learning** (IL)

  Datapoints given by $\mathrm{x}_t = \{C_t, Tree_t\}$ and expert labels $\mathrm{y}_t =$ `relpscost` decisions

  Data **augmentation schemes**:
  different seeds and initial randomization (train only)

⟶ Data heterogeneity is challenging but important to assess the framework

Experimental setup

- **Solver setting**

  SCIP 6 in **“sandbox”** setting to fairly compare branching rules, disable heuristics and provide optimal cutoff, 1h TLim

  [Gamrath and Schubert, 2018]

- **Training and validation** (hyperparameter search)

- **IL Test + SCIP evaluations**

  Never seen MILP instances and larger branching sets

  Metrics: imitation accuracy and **(fair)** nnodes

  Comparisons: GCNN and SCIP `random`, `pscost`, `relpscost`

Learned policies

- **TreeGate** better than **NoTree** in all aggregated metrics

  less clear-cut results instance-wise, but bigger reductions from TreeGate

- No time-limit (*vs.* GCNN), both policies are better than `pscost`, overall comparable to `relpscost` when accounting for SB side-effects

➤ IL accuracy

Policy	Test acc@1 (@5)	Val acc@1 (@5)
NoTree	64.02 (88.51)	77.69 (95.88)
TreeGate	**83.70** (95.83)	**84.33** (96.60)
GCNN	15.28 (44.16)	19.28 (38.44)

➤ B&B nodes
x5 SCIP runs

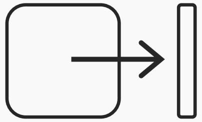
Set	NoTree	TreeGate	% diff	GCNN	`random`	`pscost`	`relpscost` (fair)
All	1241.79	1056.79	**-14.90**	*3660.32	*6580.79	*1471.61	286.15 (719.20)
Train	834.40	759.94	**-8.92**	*1391.41	*2516.04	884.37	182.27 (558.34)
Test	3068.96	2239.47	**-27.03**	*33713.63	*61828.29	*4674.34	712.77 (1276.76)

Wrap-up

To explore the idea of learning branching policies

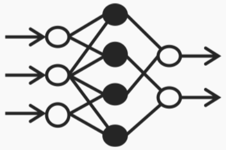
from *parameterizations of B&B search trees*

that are *shared among general MILPs*,



Represent branching in the space of B&B trees via **input features**

$$x_t = \{C_t, Tree_t\}$$



Combine data via **ML model** for branching predictions

DNN architectures, using modulation by tree-signal

We parameterize **B&B search trees** to learn **branching** policies that generalize across **heterogenous MILPs**

Incorporating **tree-related context**

- allows to approach heterogeneous MILPs w/o need of training analogs (*vs.* GCNN),
- useful for future (reinforcement) learning approaches

⟶ could also be leveraged in **MILP algorithmic design!**

Thank you! Questions?

`https://arxiv.org/abs/2002.05120` | `giulia.zarpellon@polymtl.ca`

Minimal references

[Lodi and Z., 2017] A. Lodi and G. Zarpellon. On learning and branching: a survey. TOP, 2017

[Khalil at al., 2016] E.B. Khalil, P.L. Bodic, L. Song, G. Nemhauser, and B. Dilkina. Learning to branch in mixed integer programming. In Thirtieth AAAI Conference on Artificial Intelligence, 2016

[Balcan et al., 2018] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik. Learning to branch. In 35th International Conference on Machine Learning, 2018

[Gasse et al., 2019] M. Gasse, D. Chételat, N. Ferroni, L. Charlin and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In 33rd Conference on Neural Information Processing Systems , 2019

[Berthold, Hendel and Koch., 2017] T. Berthold, G. Hendel, and T. Koch, From feasibility to improvement to proof: three phases of solving mixed-integer programs. Optimization Methods and Software, 2017.

[Fischetti, Lodi and Z., 2019] M. Fischetti, A. Lodi and G. Zarpellon. Learning MILP resolution outcomes before reaching time-limit. In International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, 2019

[Hendel at al., 2020] G. Hendel, D. Anderson, P. Le Bodic, and M.E. Pfetsch. Estimating the size of branch-and-bound trees. Optimization Online, 2020

[Gamrath and Schubert, 2018] G. Gamrath and C. Schubert. Measuring the impact of branching rules for Mixed-Integer Programming. In Operations Research Proceedings, 2017