Exact Combinatorial Optimization with Graph Convolutional Neural Networks

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, Andrea Lodi

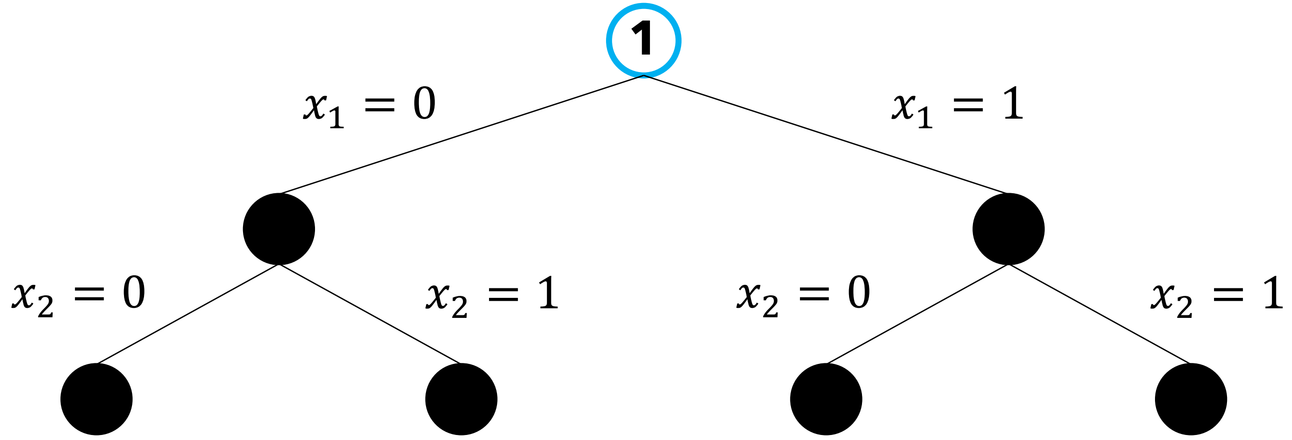
NeurIPS'19

Outline

1. **Branch-and-bound refresher**
2. Learning to imitate strong branching
3. Benchmarks
4. Experiments

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$



Incumbent: $\boldsymbol{x}^* = (0,0,0,0)$
$z^* = 0$

Problem(1):

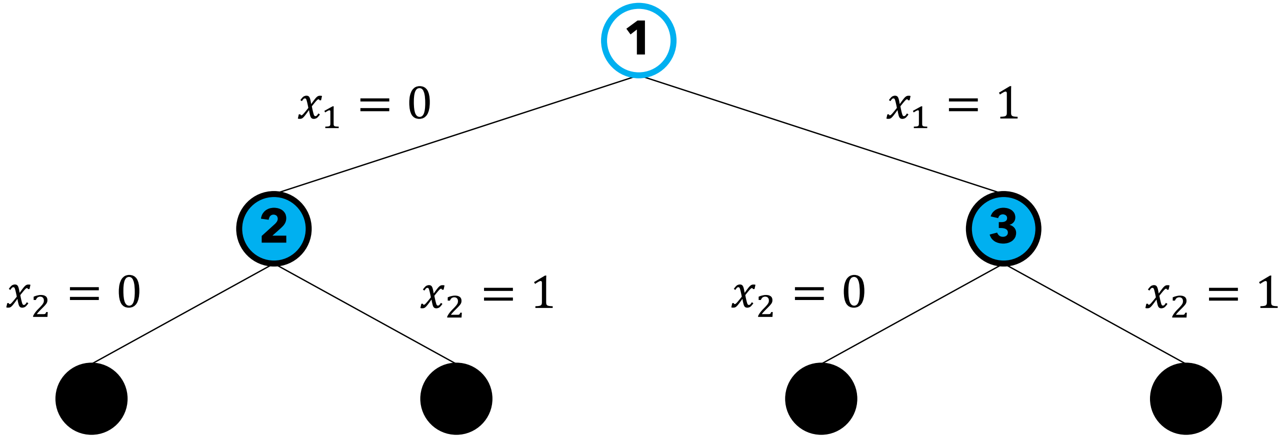
max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in [0,1]$

$\boldsymbol{x}(1) = \left(\frac{5}{8}, 1,0,0\right)$
$z(1) = 21.38$

Case 1: If $z^* < z(j)$ and $\boldsymbol{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$



Incumbent: $\boldsymbol{x}^* = (0,0,0,0)$
$z^* = 0$

Problem(1):

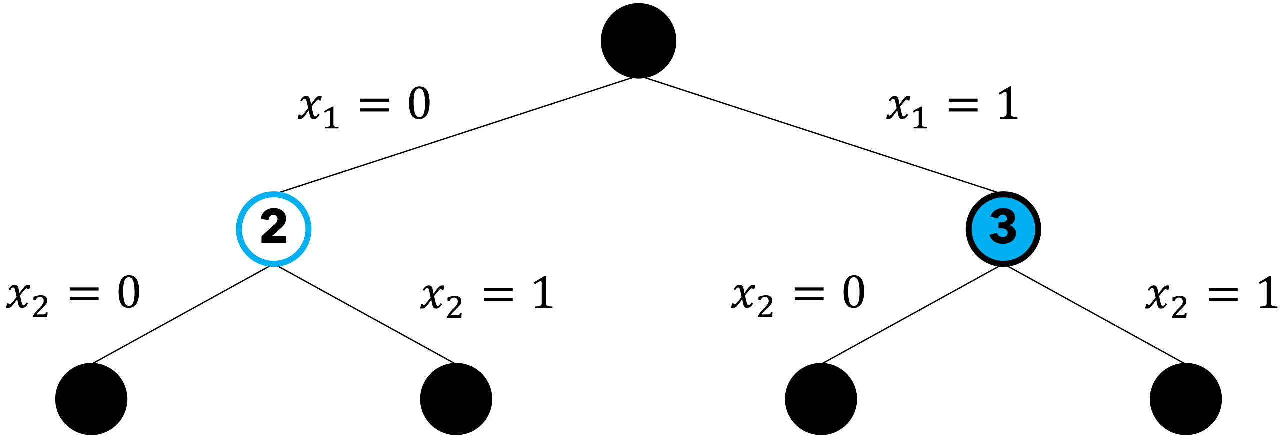
max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in [0,1]$

$\boldsymbol{x}(1) = \left(\frac{5}{8}, 1,0,0\right)$
$z(1) = 21.38$

Case 1: If $z^* < z(j)$ and $\boldsymbol{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$



Incumbent: $\boldsymbol{x}^* = (0,0,0,0)$
$z^* = 0$

Problem(2):

max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1 = 0$
	$x_2, x_3, x_4 \in [0,1]$

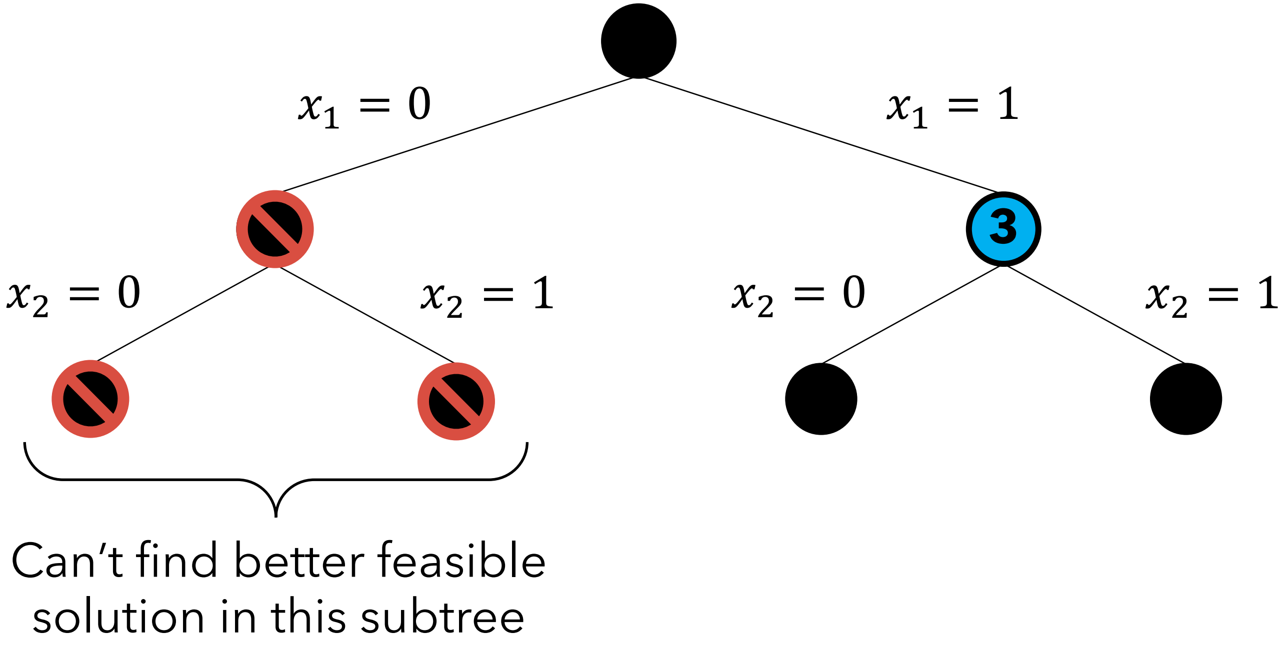
$\boldsymbol{x}(2) = (0,1,1,1)$
$z(2) = 18$

Case 2: If $z^* < z(j)$ and $\boldsymbol{x}(j)$ is feasible for IP then
Replace the incumbent by $\boldsymbol{x}(j)$ and prune node j

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$

Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$



Problem(2):

max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1 = 0$
	$x_2, x_3, x_4 \in [0,1]$

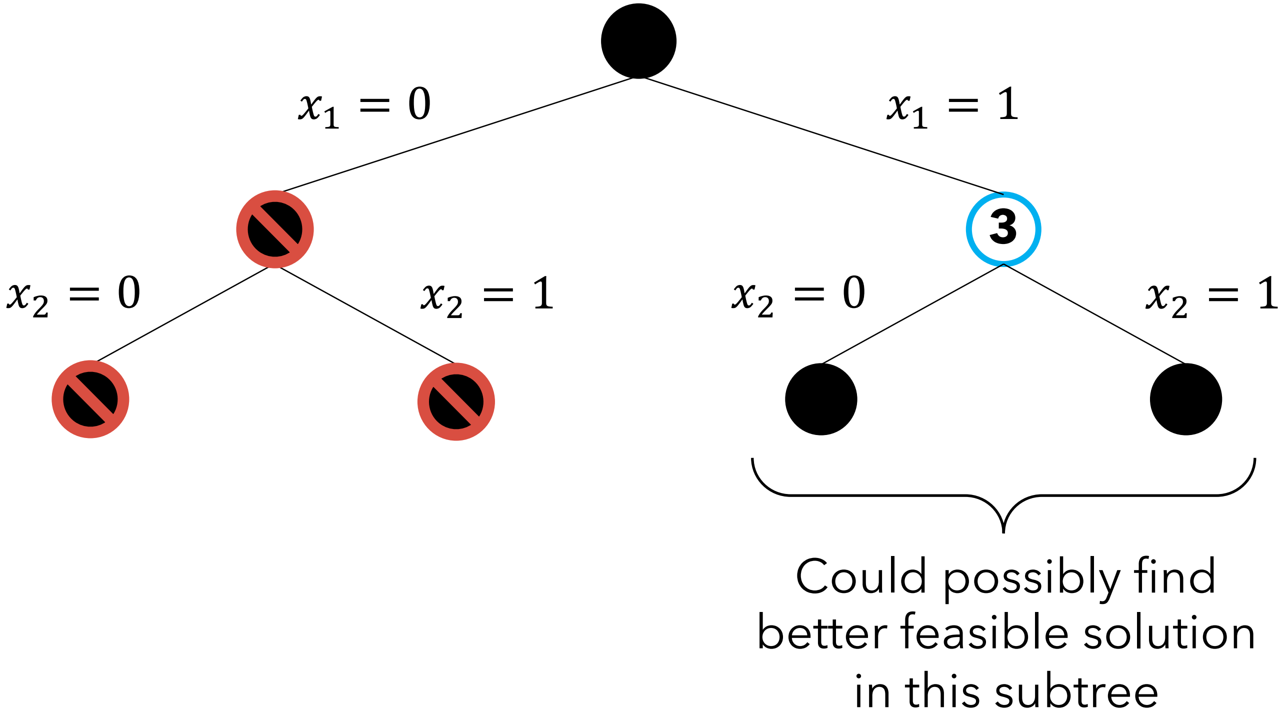
$\boldsymbol{x}(2) = (0,1,1,1)$
$z(2) = 18$

Case 2: If $z^* < z(j)$ and $\boldsymbol{x}(j)$ is feasible for IP then
Replace the incumbent by $\boldsymbol{x}(j)$ and prune node j

Branch-and-bound

maximize $15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to $8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
$x_1, x_2, x_3, x_4 \in \{0,1\}$

Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$



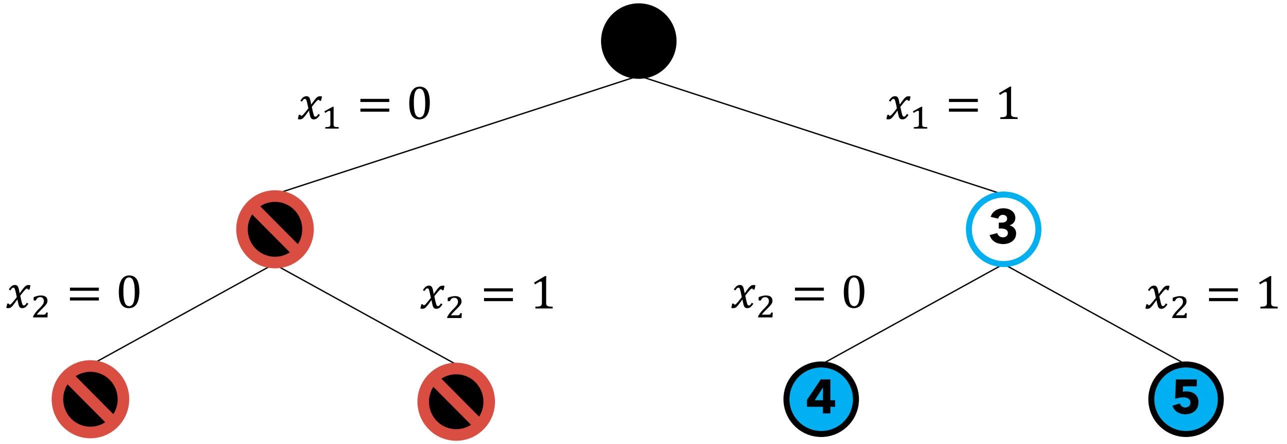
Problem(3):
max $15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t $8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
$x_1 = 1$
$x_2, x_3, x_4 \in [0,1]$

$\boldsymbol{x}(3) = \left(\frac{5}{8}, 1,0,0\right)$
$z(3) = 21.38$

Case 1: If $z^* < z(j)$ and $\boldsymbol{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$



Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$

Problem(3):

max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1 = 1$
	$x_2, x_3, x_4 \in [0,1]$

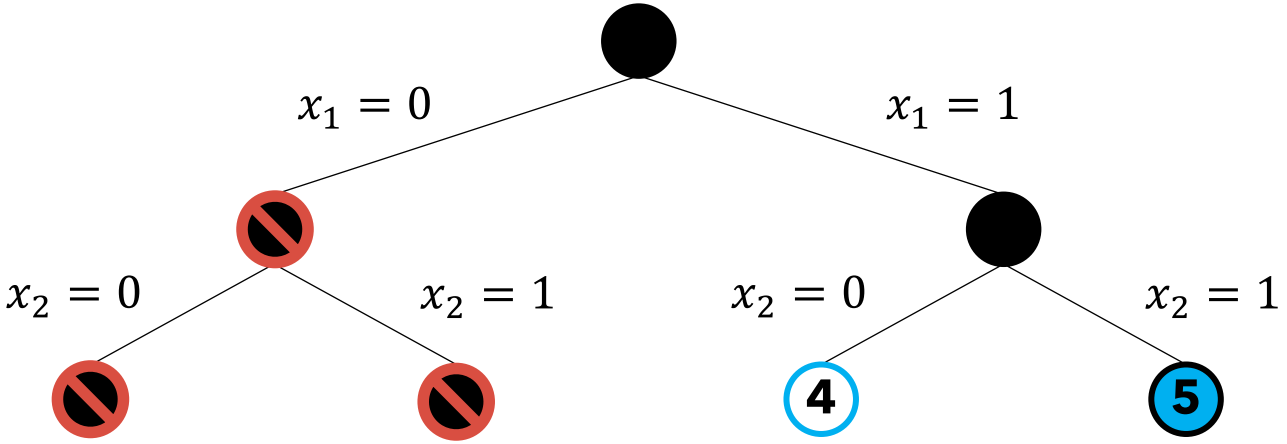
$\boldsymbol{x}(3) = \left(\frac{5}{8}, 1,0,0\right)$
$z(3) = 21.38$

Case 1: If $z^* < z(j)$ and $\boldsymbol{x}(j)$ isn't feasible for IP then
Mark the direct descendants of node j as active

Branch-and-bound

maximize $15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to $8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
$x_1, x_2, x_3, x_4 \in \{0,1\}$

Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$



Problem(4):
max $15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t $8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
$x_1 = 1$
$x_2 = 0$
$x_3, x_4 \in [0,1]$

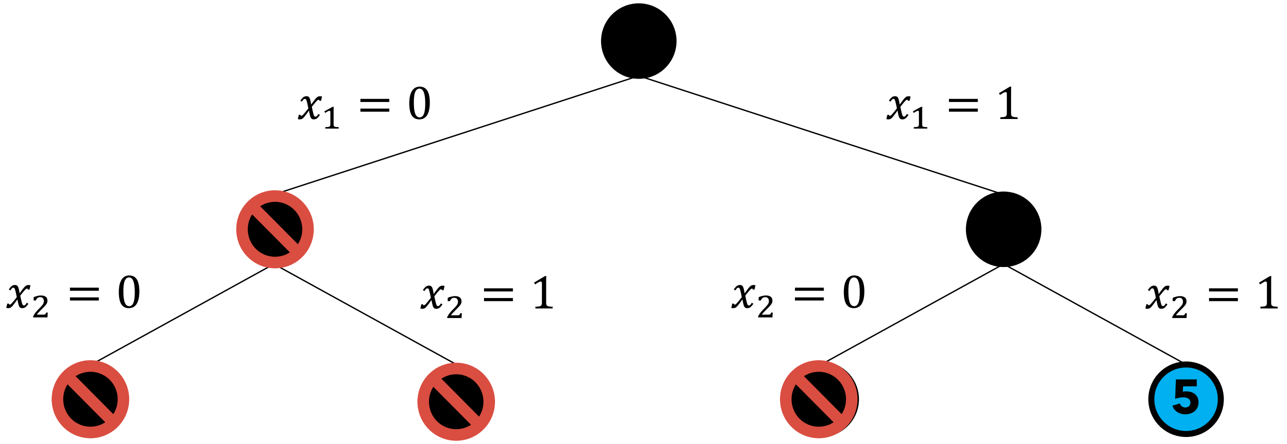
$\boldsymbol{x}(4) = \left(1,0,\frac{2}{3},0\right)$
$z(4) = 17.66$

Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$

Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$



Problem(4):

max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1 = 1$
	$x_2 = 0$
	$x_3, x_4 \in [0,1]$

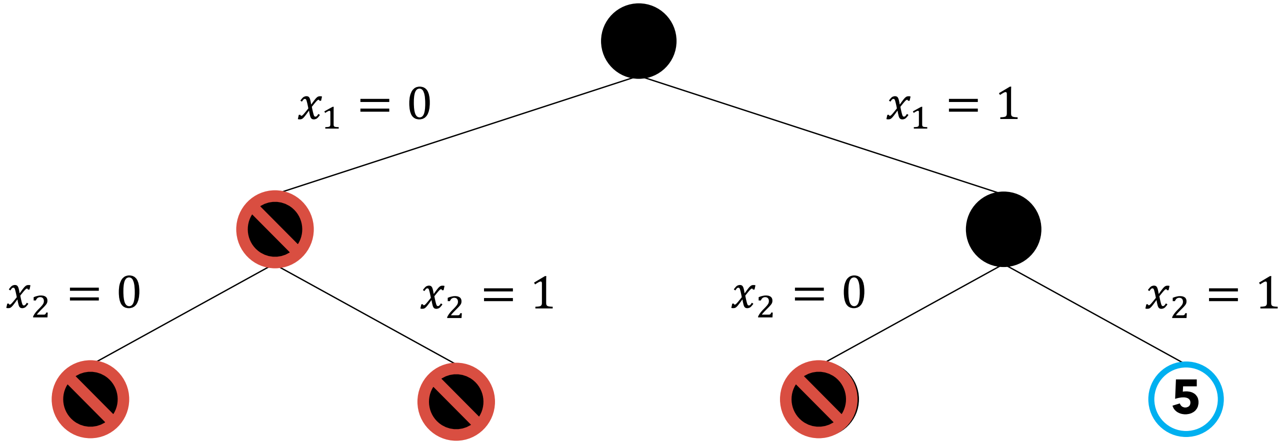
$\boldsymbol{x}(4) = \left(1,0,\frac{2}{3},0\right)$
$z(4) = 17.66$

Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$

Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$



Problem(5):

max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1 = 1$
	$x_2 = 1$
	$x_3, x_4 \in [0,1]$

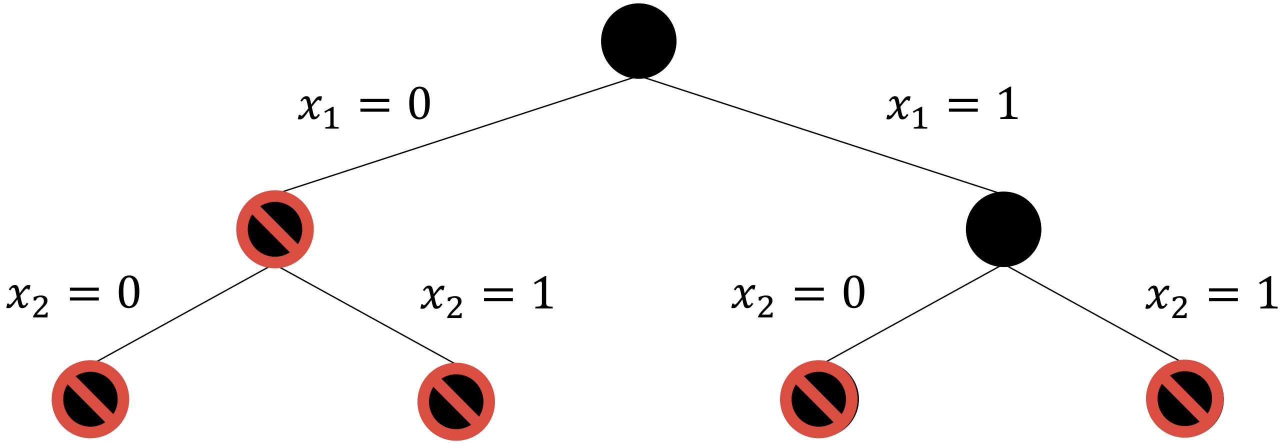
$\boldsymbol{x}(5) =$ infeasible
$z(5) =$ infeasible

Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$

Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$



Problem(5):

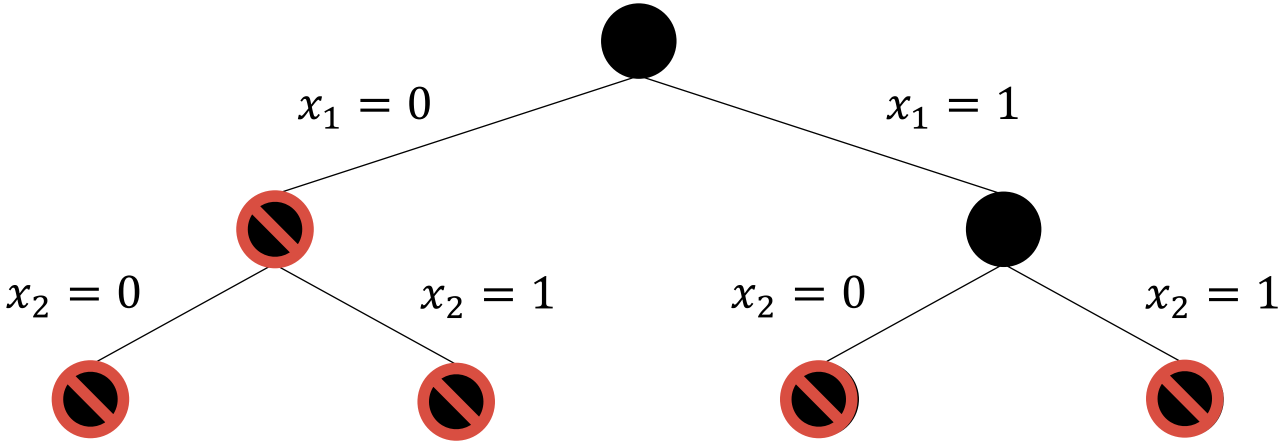
max	$15x_1 + 12x_2 + 4x_3 + 2x_4$
s.t	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1 = 1$
	$x_2 = 1$
	$x_3, x_4 \in [0,1]$

$\boldsymbol{x}(5)$ = infeasible
$z(5)$ = infeasible

Case 3: If LP is infeasible or $z^* \geq z(j)$ then prune node j

Branch-and-bound

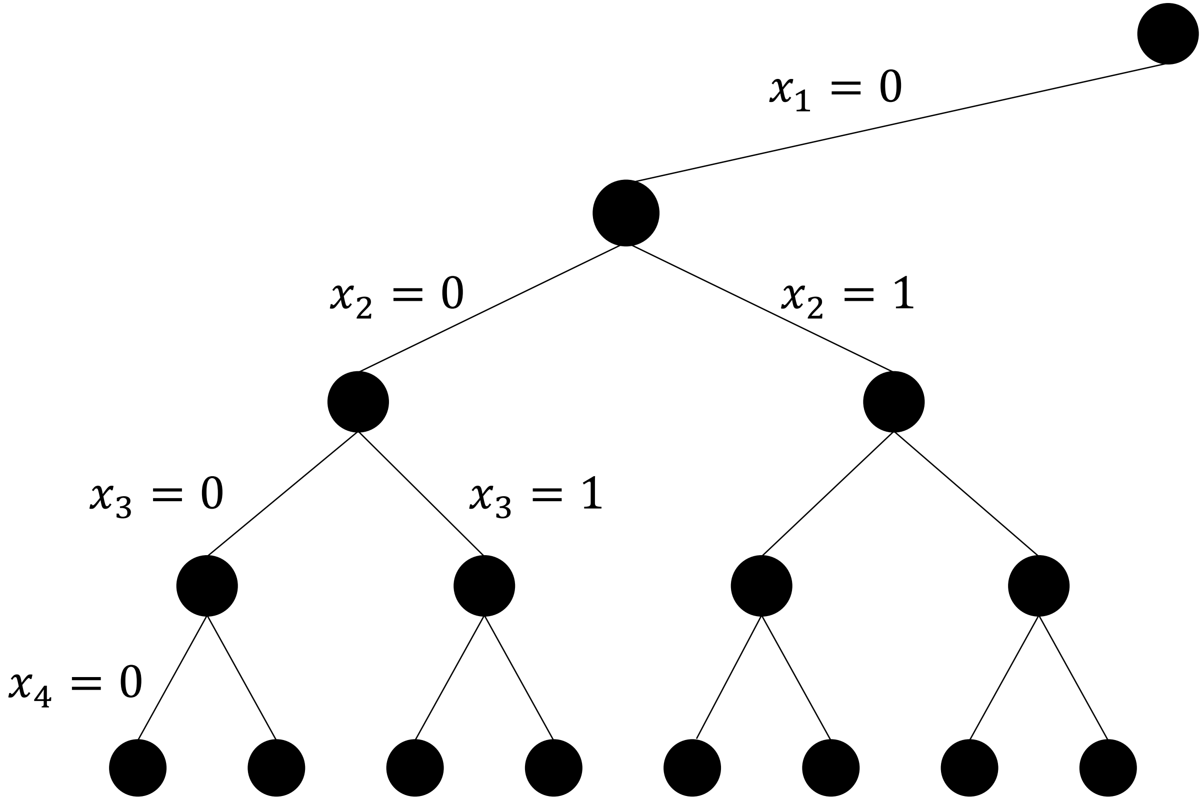
maximize	$15x_1 + 12x_2 + 4x_3 + 2x_4$
subject to	$8x_1 + 5x_2 + 3x_3 + 2x_4 \leq 10$
	$x_1, x_2, x_3, x_4 \in \{0,1\}$



Incumbent: $\boldsymbol{x}^* = (0,1,1,1)$
$z^* = 18$

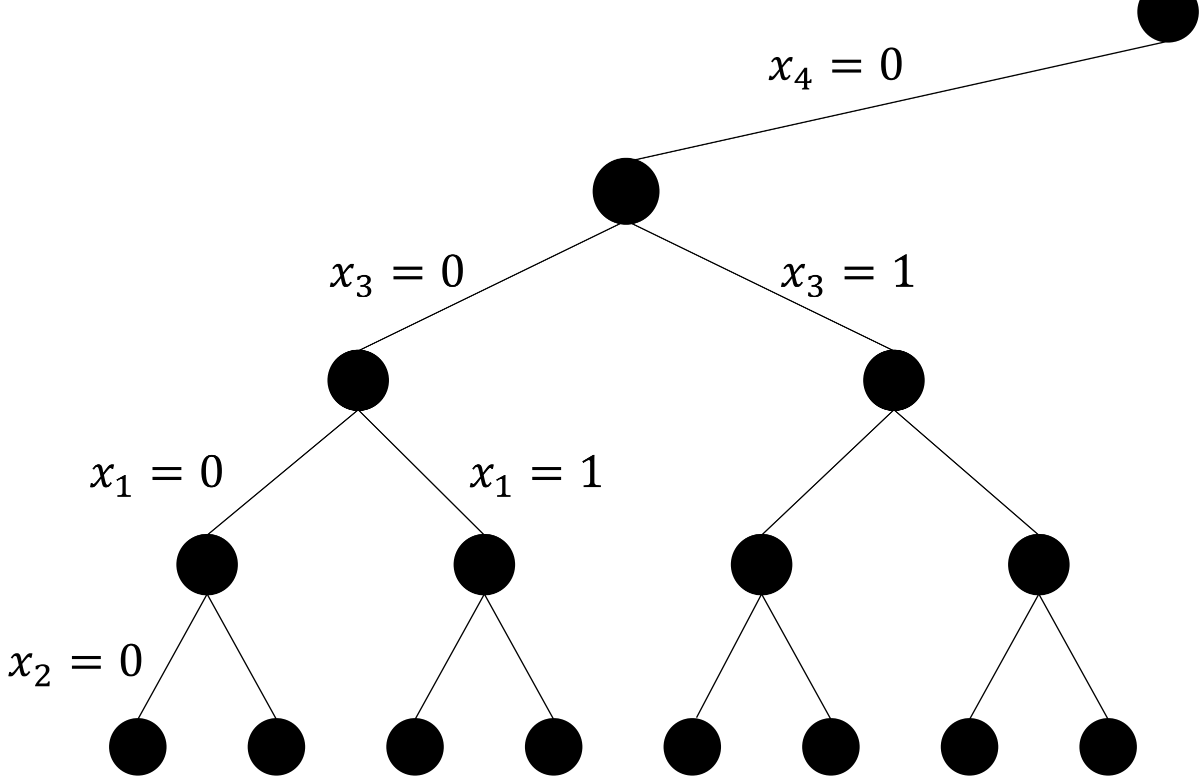
Optimal solution

Variable selection policy (VSP)



Better branching order than x_1, x_2, x_3, x_4?

Variable selection policy (VSP)

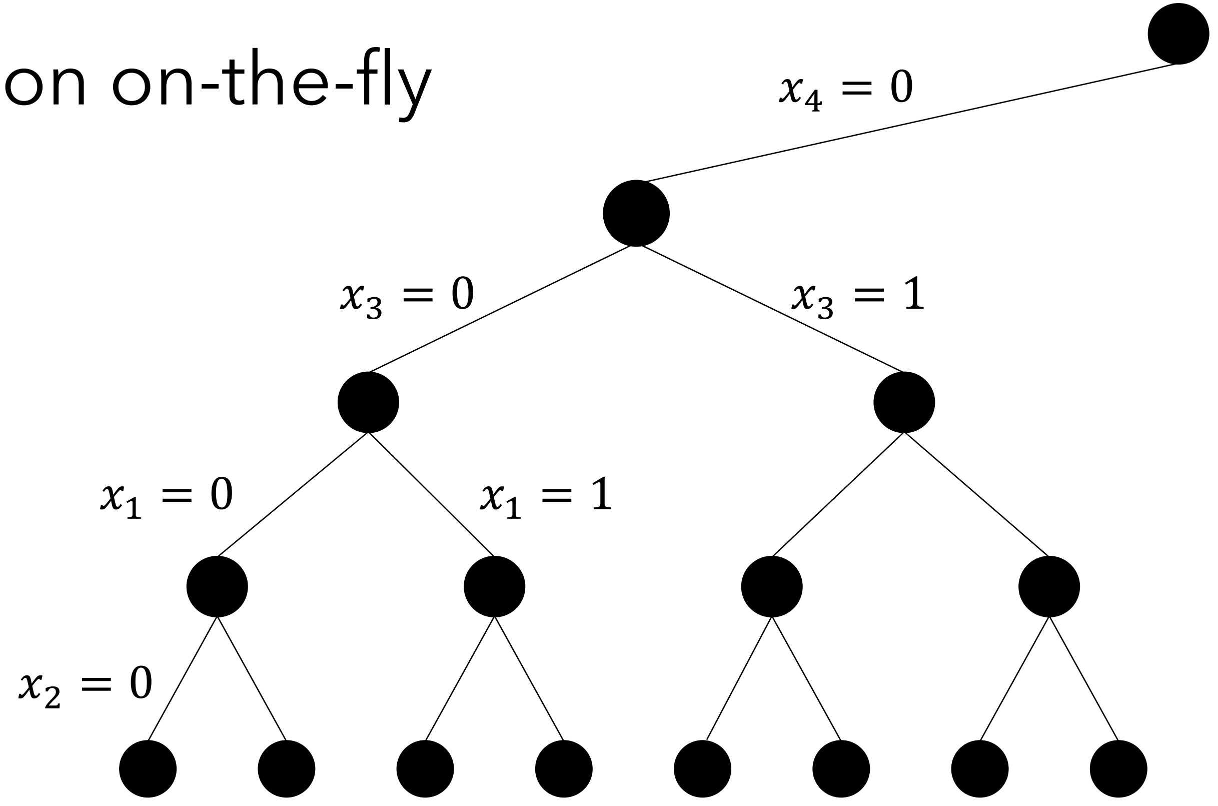


Better branching order than x_1, x_2, x_3, x_4? E.g., x_4, x_3, x_1, x_2

Variable selection policy (VSP)

Chooses variables to branch on on-the-fly

Rather than pre-defined order



Variable selection policy (VSP)

On Problem(j) with LP objective value $z(j)$:

- Let $z_i^+(j)$ be the LP objective value after setting $x_i = 1$
- Let $z_i^-(j)$ be the LP objective value after setting $x_i = 0$

VSP example:

Branch on the variable x_i that maximizes

$$\max\{z(j) - z_i^+(j), 10^{-6}\} \cdot \max\{z(j) - z_i^-(j), 10^{-6}\}$$

If score was $\big(z(j) - z_i^+(j)\big)\big(z(j) - z_i^-(j)\big)$ and $z(j) - z_i^+(j) = 0$:

would lose information stored in $z(j) - z_i^-(j)$

Outline

1. Branch-and-bound refresher
2. **Learning to imitate strong branching**
3. Benchmarks
4. Experiments

Strong branching

Challenge: Computing $z_i^-(j), z_i^+(j)$ requires solving a lot of LPs

- Computing all LP relaxations referred to as ***strong-branching***
- Very **time intensive**

Pro: Strong branching leads to small search trees

Idea: Train an ML model to imitate strong-branching

Khalil et al. [AAAI'16], Alvarez et al. [INFORMS JoC'17], Hansknecht et al. [arXiv'18]

This paper: using a GNN

Problem formulation

Goal: learn a policy $\pi(a_t \mid s_t)$

Probability of branching on variable a_t when solver is in state s_t

Approach (imitation learning)**:**

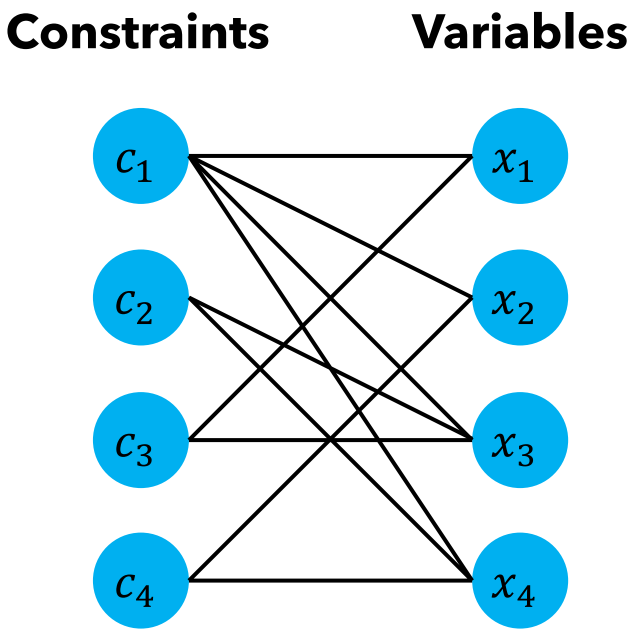
- Run strong branching on training set of instances
- Collect dataset of (state, variable) pairs $S = \{(s_i, a_i^*)\}_{i=1}^N$
- Learn policy minimizing cross-entropy loss

$$L(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N} \log \pi_{\boldsymbol{\theta}}(a_i^* \mid s_i)$$

State encoding

State s_t of B&B encoded as a **bipartite graph**
with **node** and **edge features**

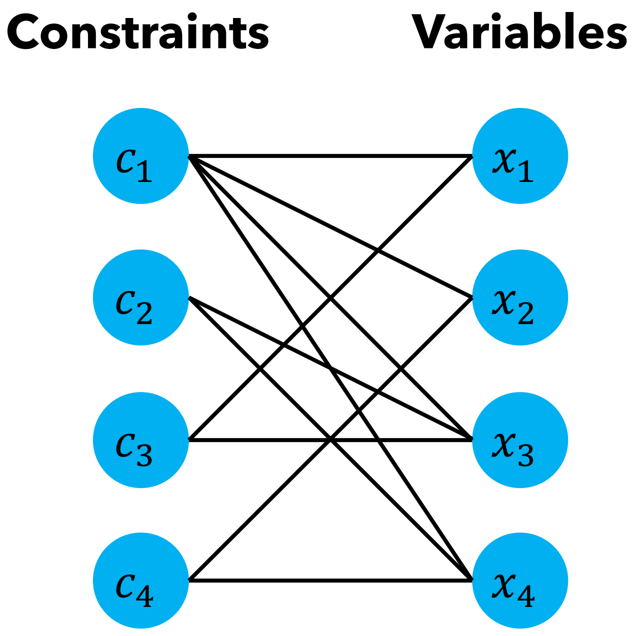
$$\begin{aligned} \max \quad & 9x_1 + 5x_2 + 6x_3 + 4x_4 \\ \text{s.t.} \quad & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 && (c_1) \\ & x_3 + x_4 \leq 10 && (c_2) \\ & -x_1 + x_3 \leq 0 && (c_3) \\ & -x_2 + x_4 \leq 0 && (c_4) \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$



State encoding

State s_t of B&B encoded as a **bipartite graph** with **node** and **edge features**

- **Edge feature:** constraint coefficient
- **Example node features:**
  - Constraints:
    - Cosine similarity with objective
    - Tight in LP solution?
  - Variables:
    - Objective coefficient
    - Solution value equals upper/lower bound?

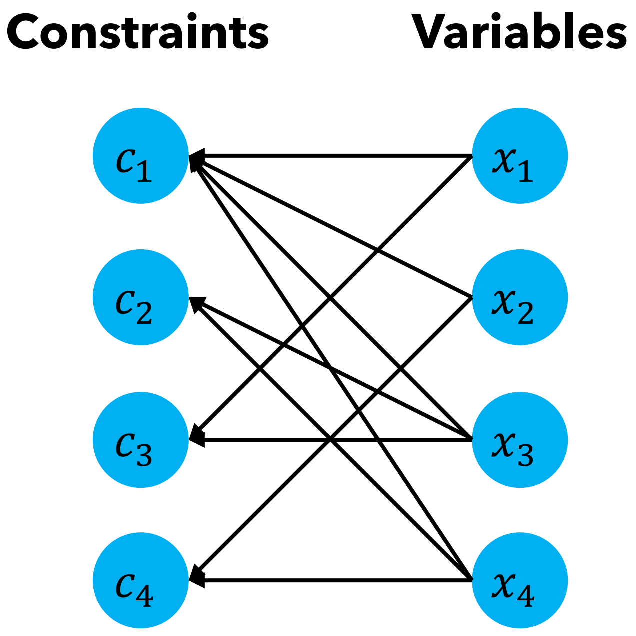


GNN structure

1. Pass from variables → constraints

$$\boldsymbol{c}_i \leftarrow f_C\left(\boldsymbol{c}_i, \sum_{j:(i,j)\in E} g_C(\boldsymbol{c}_i, \boldsymbol{v}_j, \boldsymbol{e}_{ij})\right)$$

- $\boldsymbol{c}_i$: Constraint features
- f_C, g_C: 2-layer MLP with relu activations
- $\boldsymbol{v}_j$: Variable features
- $\boldsymbol{e}_{ij}$: Edge features



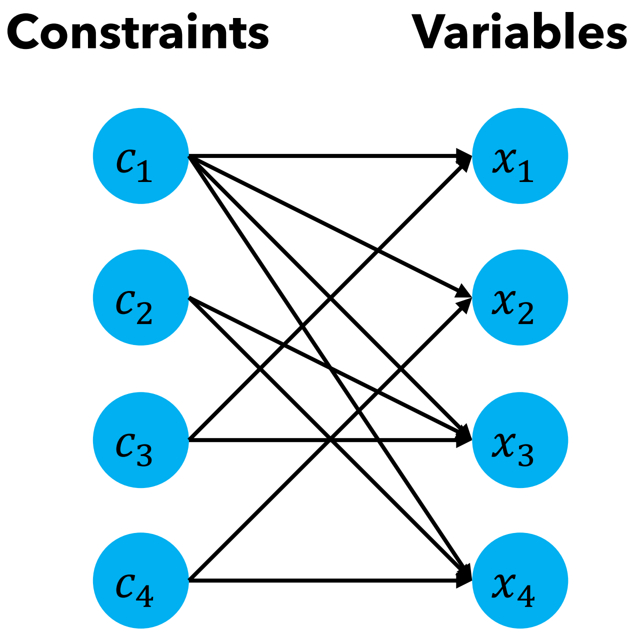
GNN structure

1. Pass from variables → constraints

$$\boldsymbol{c}_i \leftarrow f_C\left(\boldsymbol{c}_i, \sum_{j:(i,j)\in E} g_C(\boldsymbol{c}_i, \boldsymbol{v}_j, \boldsymbol{e}_{ij})\right)$$

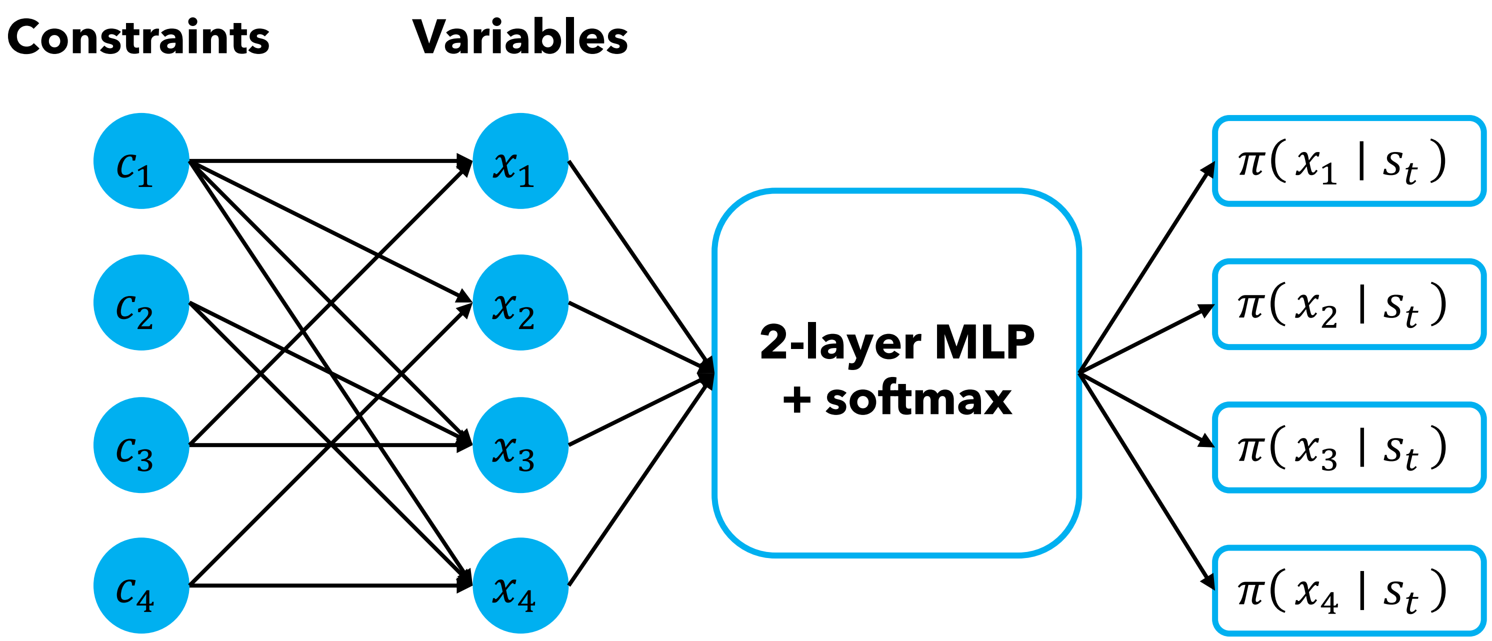
2. Pass from constraints → variables

$$\boldsymbol{v}_j \leftarrow f_V\left(\boldsymbol{v}_j, \sum_{i:(i,j)\in E} g_V(\boldsymbol{c}_i, \boldsymbol{v}_j, \boldsymbol{e}_{ij})\right)$$



GNN structure

3. Compute distribution over variables



Outline

1. Branch-and-bound refresher
2. Learning to imitate strong branching
3. **Benchmarks**
4. Experiments

Reliability pseudo-cost branching (RPB)

Rough idea:

- Goal: estimate $z(j) - z_i^+(j)$ w/o solving the LP with $x_i = 1$
- Estimate = avg change after setting $x_i = 1$ elsewhere in tree
  This is the "pseudo-cost"
- "Reliability": do strong branching if estimate is "unreliable"
  E.g., early in the tree

Default branching rule of SCIP (leading open-source solver):

$$\max\{\tilde{\Delta}_i^+(j), 10^{-6}\} \cdot \max\{\tilde{\Delta}_i^-(j), 10^{-6}\}$$

$\tilde{\Delta}_i^+(j)$: Estimate of $z(j) - z_i^+(j)$

$\tilde{\Delta}_i^-(j)$: Estimate of $z(j) - z_i^-(j)$

Achterberg and Berthold, CPAIOR'09

SVM$^{\text{rank}}$ approach [Khalil et al., AAAI'16]

- For variable x_i on Problem(j):

$$\mathrm{SB}_i^j = \max\{z(j) - z_i^+(j), 10^{-6}\} \cdot \max\{z(j) - z_i^-(j), 10^{-6}\}$$

- Define **binary labels** $y_i^j = \begin{cases} 1 & \text{if } \mathrm{SB}_i^j \geq (1-\alpha)\max \mathrm{SB}_{i'}^j \\ 0 & \text{else} \end{cases}$
- Given features $\boldsymbol{\phi}_i^j \in \mathbb{R}^d$, train **linear model** $f: \mathbb{R}^d \to \mathbb{R}$ so that:

  If $y_i^j > y_{i'}^j$, then $f\left(\boldsymbol{\phi}_i^j\right) > f\left(\boldsymbol{\phi}_{i'}^j\right)$
  - Use **SVM$^{\text{rank}}$** [Joachims, KDD'06]
- Branch on variable with largest $f\left(\boldsymbol{\phi}_i^j\right)$

lambdaMART approach [Hansknecht et al., arXiv'18]

- For variable x_i on Problem(j):

$$\mathrm{SB}_i^j = \max\{z(j) - z_i^+(j), 10^{-6}\} \cdot \max\{z(j) - z_i^-(j), 10^{-6}\}$$

- Define **binary labels** $y_i^j = \begin{cases} 1 & \text{if } \mathrm{SB}_i^j \geq (1-\alpha) \max \mathrm{SB}_{i'}^j \\ 0 & \text{else} \end{cases}$

- Given features $\boldsymbol{\phi}_i^j \in \mathbb{R}^d$, train **linear model** $f: \mathbb{R}^d \to \mathbb{R}$ so that:

  If $y_i^j > y_{i'}^j$, then $f(\boldsymbol{\phi}_i^j) > f(\boldsymbol{\phi}_{i'}^j)$

  - ~~Use **SVM**$^{\text{rank}}$ [Joachims, KDD'06]~~ **lambdaMART** [Burges, Learning'10]

- Branch on variable with largest $f(\boldsymbol{\phi}_i^j)$

Regression tree approach [Alvarez et al., INFORMS JoC'17]

- For variable x_i on Problem(j):
$$\mathrm{SB}_i^j = \max\{z(j) - z_i^+(j), 10^{-6}\} \cdot \max\{z(j) - z_i^-(j), 10^{-6}\}$$
- Given features $\boldsymbol{\phi}_i^j \in \mathbb{R}^d$, train regression tree ensemble model
$$f\left(\boldsymbol{\phi}_i^j\right) \approx \mathrm{SB}_i^j$$
- Branch on variable with largest $f\left(\boldsymbol{\phi}_i^j\right)$

Outline

1. Branch-and-bound refresher
2. Learning to imitate strong branching
3. Benchmarks
4. **Experiments**

Imitation learning accuracy

	Set Covering			Maximum Independent Set		
model	acc@1	acc@5	acc@10	acc@1	acc@5	acc@10
TREES	51.8±0.3	80.5±0.1	91.4±0.2	30.9±0.4	47.4±0.3	54.6±0.3
SVMRANK	57.6±0.2	84.7±0.1	94.0±0.1	48.0±0.6	69.3±0.2	78.1±0.2
LMART	57.4±0.2	84.5±0.1	93.8±0.1	48.9±0.3	68.9±0.4	77.0±0.5
GCNN	**65.5**±0.1	**92.4**±0.1	**98.2**±0.0	**56.5**±0.2	**80.8**±0.3	**89.0**±0.1

- acc@1: **highest-ranked variable** same as strong branching?
- acc@5: 1 of the 5 highest ranked variables same as SB?
- acc@10: 1 of the 10 highest ranked variables same as SB?

Set covering instances

Always train on "easy" instances

	1000 columns, 500 rows			1000 columns, 2000 rows		
	Easy			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes
FSB	17.30 ± 6.1%	0 / 100	17 ±13.7%	3600.00 ± 0.0%	0 / 0	n/a ± n/a %
RPB	8.98 ± 4.8%	0 / 100	**54** ±20.8%	1677.02 ± 3.0%	4 / 65	47 299 ± 4.9%
TREES	9.28 ± 4.9%	0 / 100	187 ± 9.4%	2869.21 ± 3.2%	0 / 35	59 013 ± 9.3%
SVMRANK	8.10 ± 3.8%	1 / 100	165 ± 8.2%	2389.92 ± 2.3%	0 / 47	42 120 ± 5.4%
LMART	7.19 ± 4.2%	14 / 100	167 ± 9.0%	2165.96 ± 2.0%	0 / 54	45 319 ± 3.4%
GCNN	**6.59** ± 3.1%	**85** / 100	134 ± 7.6%	**1489.91** ± 3.3%	**66** / 70	**29 981** ± 4.9%

Set covering instances

Runtime in seconds with a timeout of 1 hour

Number instances with fastest runtime / number solved

Size of B&B tree

Model	Easy			Hard		
	Time	Wins	Nodes	Time	Wins	Nodes
FSB	17.30 ± 6.1%	0 / 100	17 ±13.7%	3600.00 ± 0.0%	0 / 0	n/a ± n/a %
RPB	8.98 ± 4.8%	0 / 100	**54** ±20.8%	1677.02 ± 3.0%	4 / 65	47 299 ± 4.9%
TREES	9.28 ± 4.9%	0 / 100	187 ± 9.4%	2869.21 ± 3.2%	0 / 35	59 013 ± 9.3%
SVMRANK	8.10 ± 3.8%	1 / 100	165 ± 8.2%	2389.92 ± 2.3%	0 / 47	42 120 ± 5.4%
LMART	7.19 ± 4.2%	14 / 100	167 ± 9.0%	2165.96 ± 2.0%	0 / 54	45 319 ± 3.4%
GCNN	**6.59** ± 3.1%	**85** / 100	134 ± 7.6%	**1489.91** ± 3.3%	**66** / 70	**29 981** ± 4.9%

Set covering instances

- GNN is **faster than SCIP** default VSP (RPB)
- Performance generalizes to **larger instances**
- Similar results for auction design & facility location problems

Model	Easy			Hard		
	Time	Wins	Nodes	Time	Wins	Nodes
FSB	17.30 ± 6.1%	0 / 100	17 ±13.7%	3600.00 ± 0.0%	0 / 0	n/a ± n/a %
RPB	8.98 ± 4.8%	0 / 100	**54** ±20.8%	1677.02 ± 3.0%	4 / 65	47 299 ± 4.9%
TREES	9.28 ± 4.9%	0 / 100	187 ± 9.4%	2869.21 ± 3.2%	0 / 35	59 013 ± 9.3%
SVMRANK	8.10 ± 3.8%	1 / 100	165 ± 8.2%	2389.92 ± 2.3%	0 / 47	42 120 ± 5.4%
LMART	7.19 ± 4.2%	14 / 100	167 ± 9.0%	2165.96 ± 2.0%	0 / 54	45 319 ± 3.4%
GCNN	**6.59** ± 3.1%	**85** / 100	134 ± 7.6%	**1489.91** ± 3.3%	**66** / 70	**29 981** ± 4.9%

Max independent set instances

RPB is catching up to GNN on MIS instances

Model	Easy Time	Easy Wins	Easy Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	23.58 ±29.9%	9 / 100	7 ±35.9%	3600.00 ± 0.0%	0 / 0	n/a ± n/a %
RPB	8.77 ±11.8%	7 / 100	**20** ±36.1%	2045.61 ±18.3%	22 / 42	**2675** ±24.0%
TREES	10.75 ±22.1%	1 / 100	76 ±44.2%	3565.12 ± 1.2%	0 / 3	38 296 ± 4.1%
SVMRANK	8.83 ±14.9%	2 / 100	46 ±32.2%	2902.94 ± 9.6%	1 / 18	6256 ±15.1%
LMART	7.31 ±12.7%	30 / 100	52 ±38.1%	3044.94 ± 7.0%	0 / 12	8893 ± 3.5%
GCNN	**6.43** ±11.6%	**51** / 100	43 ±40.2%	**2024.37** ±30.6%	**25** / 29	2997 ±26.3%

Overview

Proposed a **variable selection policy** based on GNNs

Outperforms default policy of **SCIP**

Generalizes to **larger MIPs** than trained on