Exact Combinatorial Optimization with Graph Convolutional Neural Networks

Angelo Rajendram

25 Mar 2024

Problem Setting

- Goal: Solve combinatorial optimization – wide-ranging applicability:
  - Logistics: Airline routing & freight transportation
  - Network Design: Telecommunication networks & circuit layouts
  - Planning and scheduling: Inventory management & manufacturing lines

Problem Setting

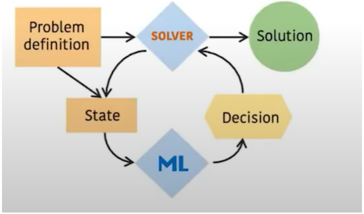
- Goal: Solve combinatorial optimization – wide-ranging applicability:
  - Logistics: Airline routing & freight transportation
  - Network Design: Telecommunication networks & circuit layouts
  - Planning and scheduling: Inventory management & manufacturing lines
- Exact optimization algorithms guarantee optimal solutions despite worst case exponential time-complexity
  - If interrupted prior to termination, provides intermediate solution accompanied by an optimality bound (useful in practice)

Problem Setting

- Goal: Solve combinatorial optimization – wide-ranging applicability:
  - Logistics: Airline routing & freight transportation
  - Network Design: Telecommunication networks & circuit layouts
  - Planning and scheduling: Inventory management & manufacturing lines
- Exact optimization algorithms guarantee optimal solutions despite worst case exponential time-complexity
  - If interrupted prior to termination, provides intermediate solution accompanied by an optimality bound (useful in practice)
  - Traditional solvers are built with hard-coded heuristics **but are not tuned to specific problem families of interest to the end-user**

Problem Setting

- Goal: Solve combinatorial optimization – wide-ranging applicability:
  - Logistics: Airline routing & freight transportation
  - Network Design: Telecommunication networks & circuit layouts
  - Planning and scheduling: Inventory management & manufacturing lines
- Exact optimization algorithms guarantee optimal solutions despite worst case exponential time-complexity
  - If interrupted prior to termination, provides intermediate solution accompanied by an optimality bound (useful in practice)
  - Traditional solvers are built with hard-coded heuristics **but are not tuned to specific problem families of interest to the end-user**
  - Proposal: Use ML to learn heursitic inside of solver



Background: Mixed Integer Linear Programming (MILP)

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{x}} \quad \mathbf{c}^T \mathbf{x} \\ &\text{s.t.} \quad A\mathbf{x} \leq \mathbf{b} \\ &\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ &\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

Background: Mixed Integer Linear Programming (MILP)

objective coefficient vector $\quad \text{argmin}_{\mathbf{x}} \quad \mathbf{c}^T \mathbf{x} \quad$ decision variables

$$
\begin{aligned}
&\text{s.t.} \quad A\mathbf{x} \leq \mathbf{b} \\
&\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\
&\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}
\end{aligned}
$$

Background: Mixed Integer Linear Programming (MILP)

objective coefficient vector $\text{argmin}_{\mathbf{x}} \quad \mathbf{c}^T \mathbf{x}$ decision variables

constraint coefficient matrix s.t. $A\mathbf{x} \leq \mathbf{b}$ constraint RHS vector

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$$

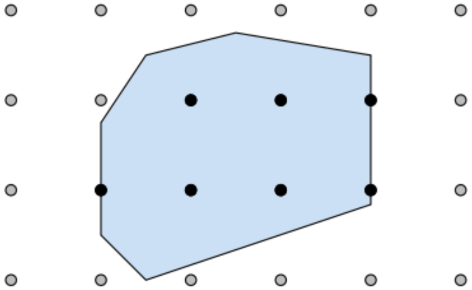
$$\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$$

Background: Mixed Integer Linear Programming (MILP)

objective coefficient vector	$\operatorname{argmin}_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x}$	decision variables
constraint coefficient matrix	s.t. $\quad A\mathbf{x} \leq \mathbf{b}$	constraint RHS vector
variable lower bound	$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$	variable upper bound
	$\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$	

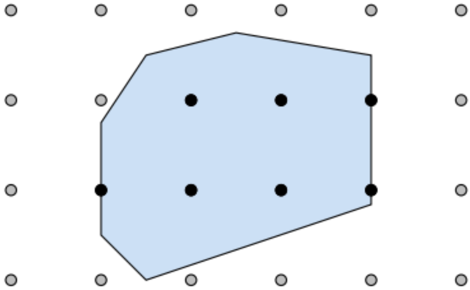
Background: Mixed Integer Linear Programming (MILP)

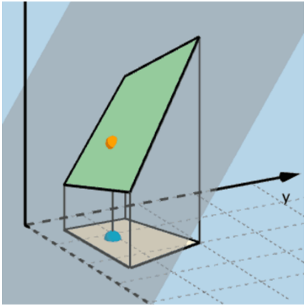
objective coefficient vector	$\text{argmin}_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x}$	decision variables
constraint coefficient matrix	s.t. $\quad A\mathbf{x} \leq \mathbf{b}$	constraint RHS vector
variable lower bound	$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$	variable upper bound
integrality constraints	$\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$	real-valued variables



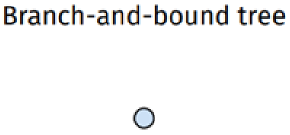
Background: Mixed Integer Linear Programming (MILP)

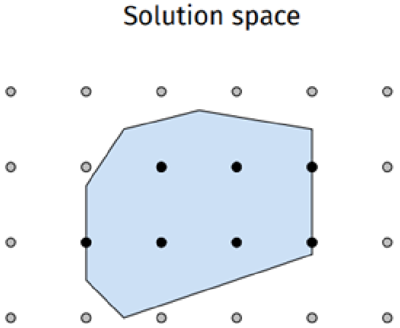
objective coefficient vector	$\text{argmin}_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x}$	decision variables
constraint coefficient matrix	s.t. $\quad A\mathbf{x} \leq \mathbf{b}$	constraint RHS vector
variable lower bound	$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$	variable upper bound
integrality constraints	$\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$	real-valued variables



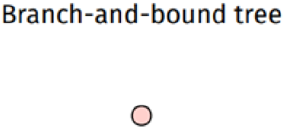


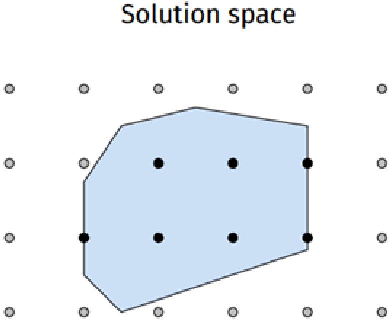
Background: Branch & Bound (An Exact Method)



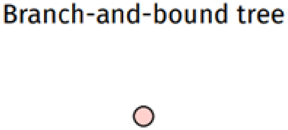


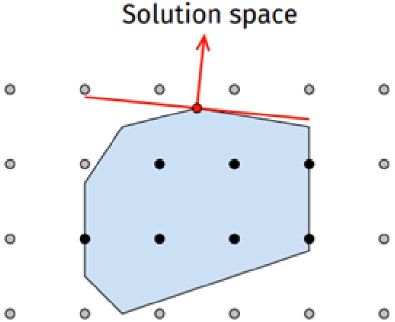
Background: Branch & Bound (An Exact Method)





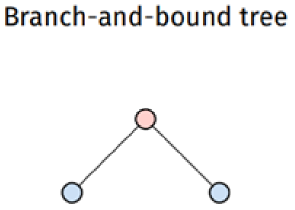
Background: Branch & Bound (An Exact Method)

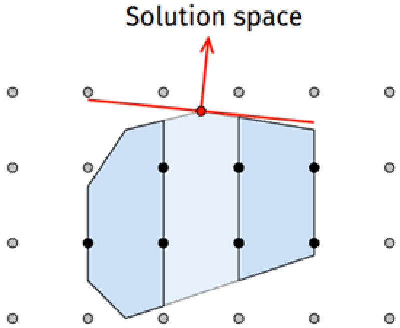




Lower bound: minimial among leaf nodes

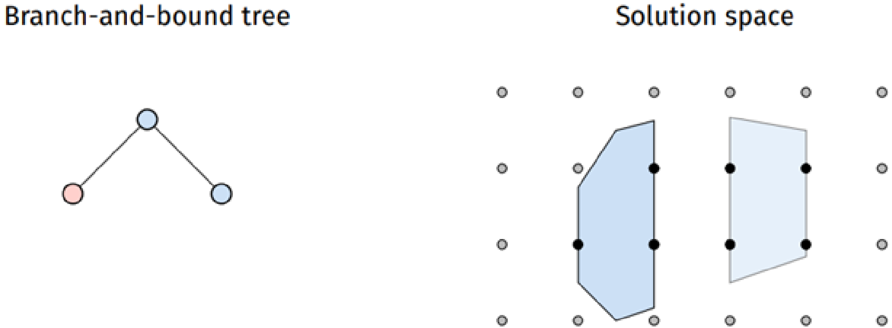
Background: Branch & Bound (An Exact Method)





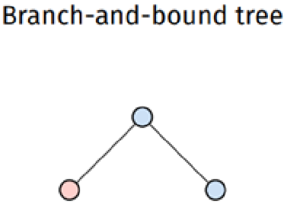
Lower bound: minimial among leaf nodes

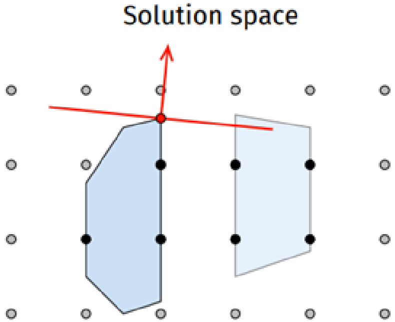
Background: Branch & Bound (An Exact Method)



Lower bound: minimial among leaf nodes

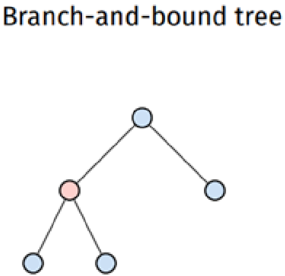
Background: Branch & Bound (An Exact Method)

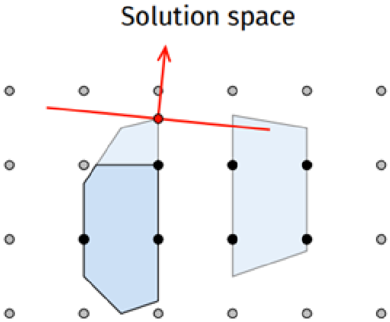




Lower bound: minimial among leaf nodes

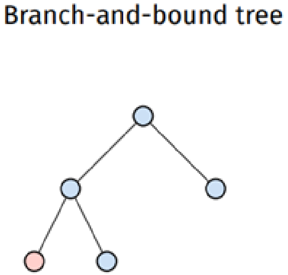
Background: Branch & Bound (An Exact Method)

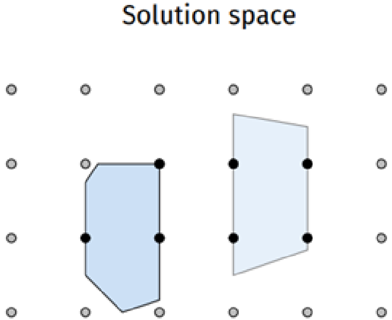




Lower bound: minimial among leaf nodes

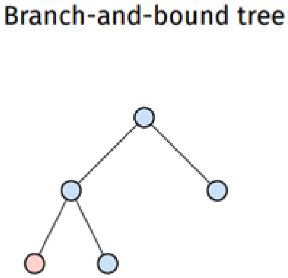
Background: Branch & Bound (An Exact Method)

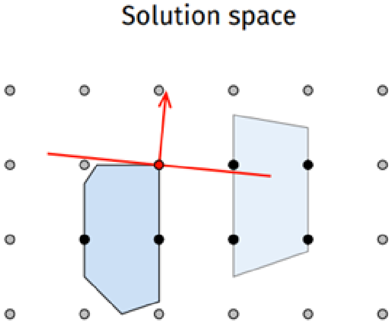




Lower bound: minimial among leaf nodes

Background: Branch & Bound (An Exact Method)

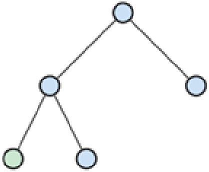




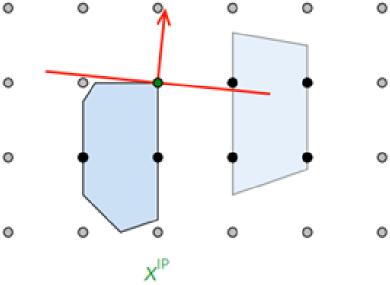
Lower bound: minimial among leaf nodes

Background: Branch & Bound (An Exact Method)

Branch-and-bound tree



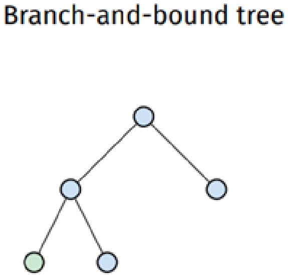
Solution space

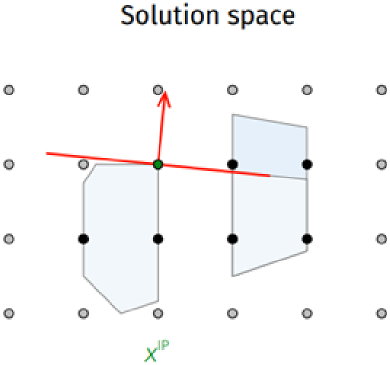


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

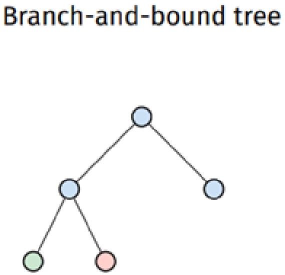


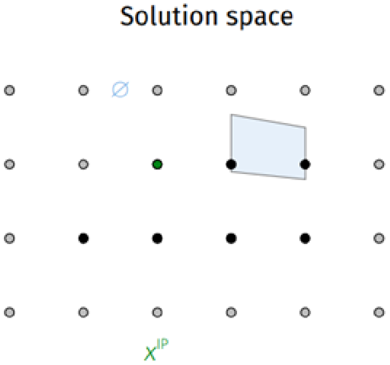


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

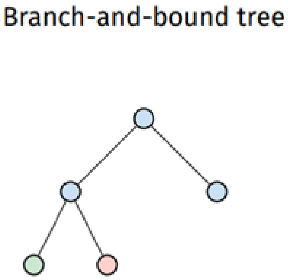


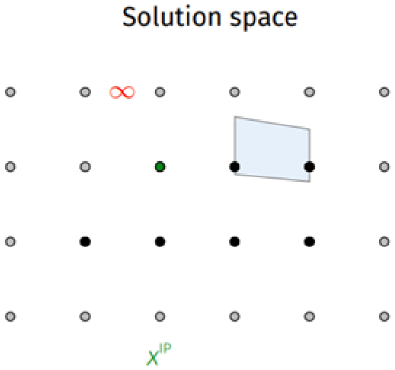


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

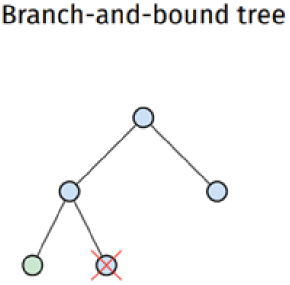


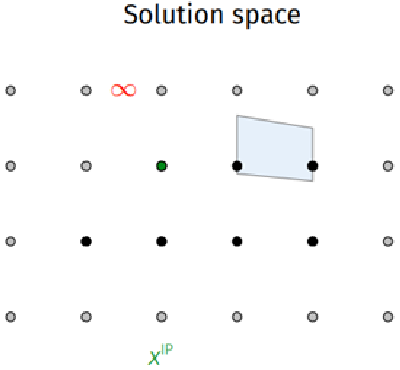


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

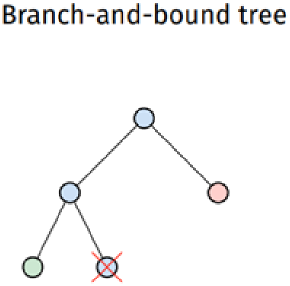


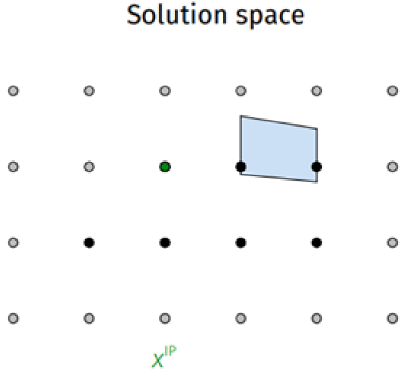


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

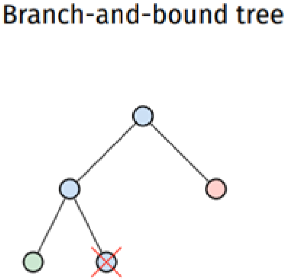


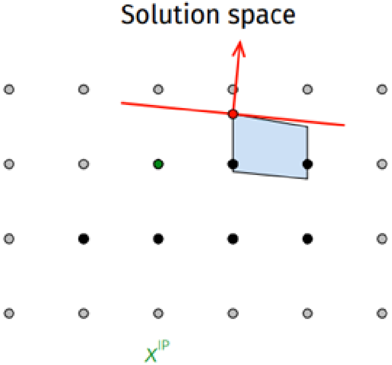


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

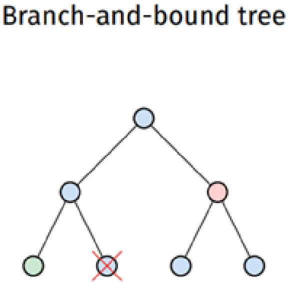


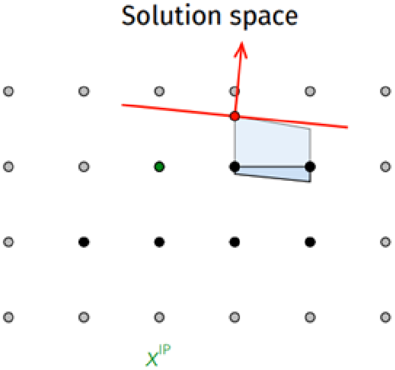


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

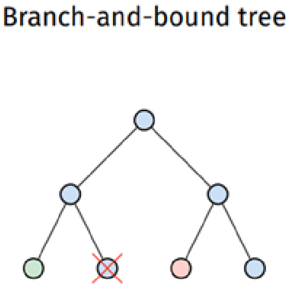


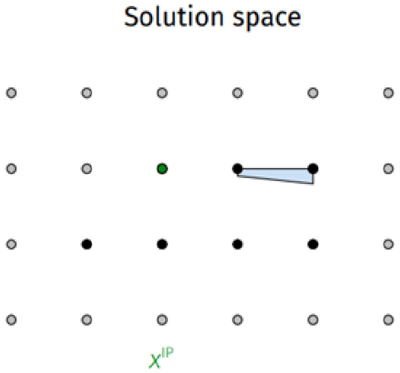


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

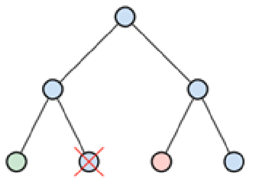


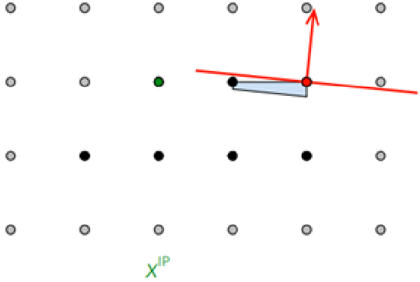


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

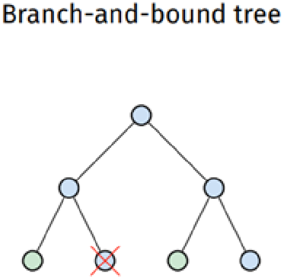


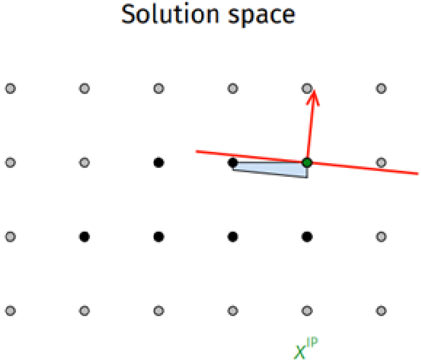


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)



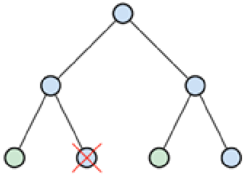


Lower bound: minimial among leaf nodes

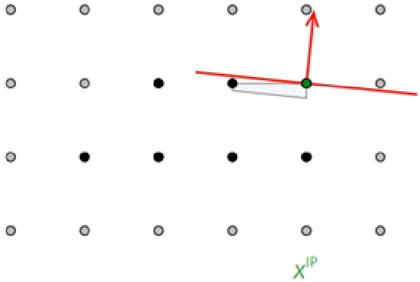
Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

Branch-and-bound tree



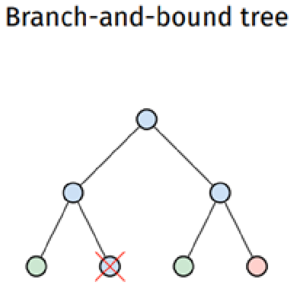
Solution space

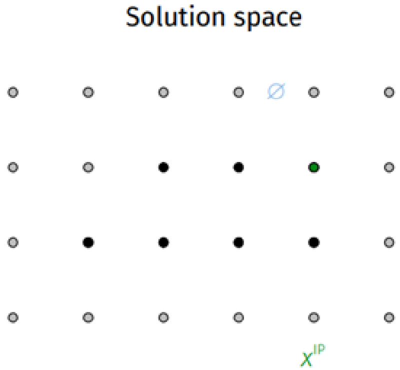


Lower bound: minimial among leaf nodes

Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)



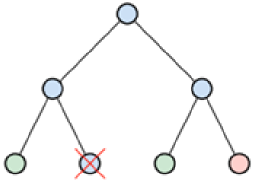


Lower bound: minimial among leaf nodes

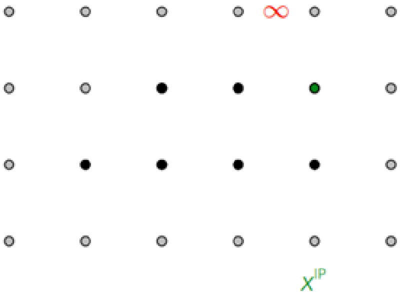
Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

Branch-and-bound tree



Solution space

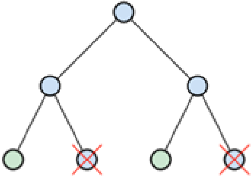


Lower bound: minimial among leaf nodes

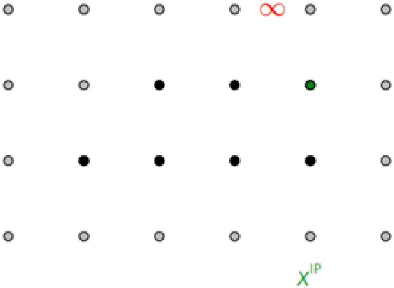
Upper bound: minimial among leaf nodes with integral solution

Background: Branch & Bound (An Exact Method)

Branch-and-bound tree



Solution space



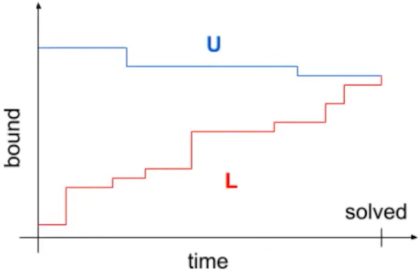
Lower bound: minimal among leaf nodes

Upper bound: minimal among leaf nodes with integral solution

Background: Branch & Bound: Sequential Decision Making

Sequential Decisions in B&B

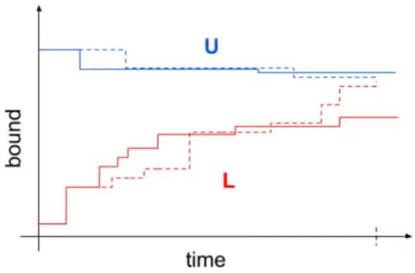
- variable selection (branching)
- node selection
- cutting plane selection
- primal heuristic selection
- simplex initialization
- ...



Background: Branch & Bound: Sequential Decision Making

Sequential Decisions in B&B

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



Objective?

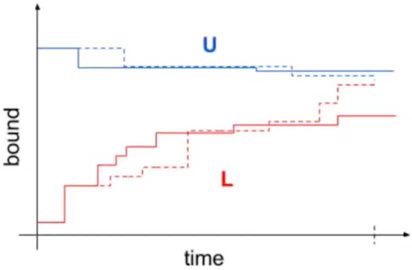
- L=U fast?
- U-L $\searrow$ fast?
- L $\nearrow$ fast?
- U $\searrow$ fast?
- fast on {*problem family*}

(e.g. TSP, sheduling, etc.)

Background: Branch & Bound: Sequential Decision Making

Sequential Decisions in B&B

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



Objective?

- L=U fast?
- U-L $\searrow$ fast?
- L $\nearrow$ fast?
- U $\searrow$ fast?
- fast on {*problem family*}

(e.g. TSP, sheduling, etc.)

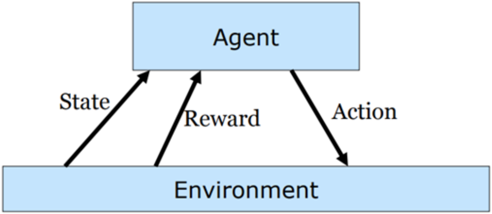
Strong branching (SB): one-step forward looking

- solve both LPs for each candidate variable (that does not satisfy integrality constraint)
- pick variable resulting in tightest relaxation

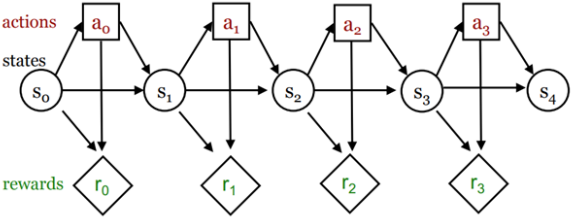
\+ small trees

− computationally expensive

Background: Markov Decision Process (MDP) Framework



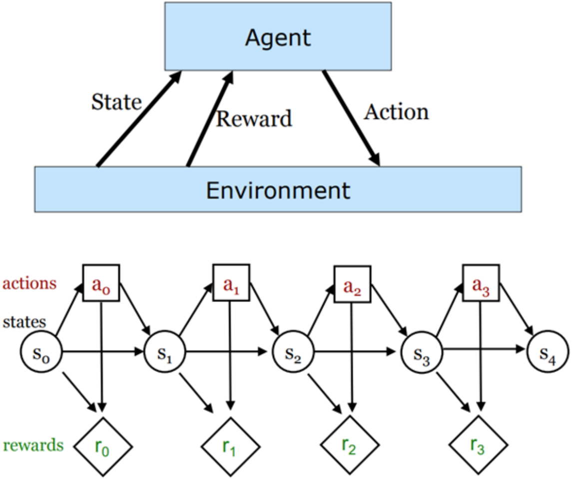
Reinforcement Learning (RL)



$(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$
$\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
$\gamma \in (0, 1]$
$\pi : \mathcal{S} \to \mathcal{A}$

Goal: Take actions (learn policy π^*) that maximize $\sum_{t=0}^{\infty} \gamma^t r(s_t)$

Background: Markov Decision Process (MDP) Framework



Imitation Learning

$(\mathcal{S}, \mathcal{A}, \mathcal{T}, \cancel{r}, \cancel{\gamma})$

$\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

$\cancel{r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}}$

$\cancel{\gamma \in (0, 1]}$

$\pi : \mathcal{S} \to \mathcal{A}$

~~Goal: Take actions (learn policy π^*) that maximize $\sum_{t=0}^{\infty} \gamma^t r(s_t)$~~

Goal: Imitate expert policy π^*

Branching as a Markov Decision Process

MDP: $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \cancel{r}, \cancel{\gamma})$

Branching as a Markov Decision Process

MDP: $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \cancel{r}, \cancel{\gamma})$

- States $\mathcal{S}$: the internal state of the solver and the search tree
- Actions $\mathcal{A}$: branch on variable not satisfying integrality constraint, $a \in \{1, ..., p\}$
- Transition function $\mathcal{T}$: Depends on MILP instance, solver, and heuristics

Branching as a Markov Decision Process

MDP: $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \cancel{r}, \cancel{\gamma})$

- States $\mathcal{S}$: the internal state of the solver and the search tree
- Actions $\mathcal{A}$: branch on variable not satisfying integrality constraint, $a \in \{1, ..., p\}$
- Transition function $\mathcal{T}$: Depends on MILP instance, solver, and heuristics

Trajectory: $\tau = (s_0, ..., s_T)$ with $s_t \in \mathcal{S}$

- initial state s_0: a MILP $\sim p(s_0)$ and solver configuration
- terminal state s_T: the MILP is solved
- intermediate states: branching

$$s_{t+1} \sim p_\pi(s_{t+1}|s_t) = \sum_{a \in \mathcal{A}} \overbrace{\pi(a|s_t)}^{\text{branching policy}} \overbrace{p(s_{t+1}|s_t, a)}^{\text{solver internals}}$$

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

– dynamically growing search tree

– variable-size instances

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

– dynamically growing search tree

– variable-size instances (Use GNN!)

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

– dynamically growing search tree
– variable-size instances (Use GNN!)

Sampling trajectories for training: $\tau \sim p_\pi$

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

– dynamically growing search tree
– variable-size instances (Use GNN!)

Sampling trajectories for training: $\tau \sim p_\pi$

- obtaining a single sample $\tau =$ solving MILP

– NP-hard, and π likely not optimal
– expensive, so must train on small instances

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

– dynamically growing search tree

– variable-size instances (Use GNN!)

Sampling trajectories for training: $\tau \sim p_\pi$

- obtaining a single sample τ = solving MILP

– NP-hard, and π likely not optimal

– expensive, so must train on small instances (GNNs can generalize!)

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

– dynamically growing search tree

– variable-size instances (Use GNN!)

Sampling trajectories for training: $\tau \sim p_\pi$

- obtaining a single sample τ = solving MILP

– NP-hard, and π likely not optimal

– expensive, so must train on small instances (GNNs can generalize!)

Reward function: r

- no consensus

Branching as a Markov Decision Process

State representation: $s \in \mathcal{S}$

- global information: original MILP, search tree, bounds, focused node, ...
- node information: variable bounds, LP solution, simplex statistics, ...

– dynamically growing search tree
– variable-size instances (Use GNN!)

Sampling trajectories for training: $\tau \sim p_\pi$

- obtaining a single sample $\tau =$ solving MILP

– NP-hard, and π likely not optimal
– expensive, so must train on small instances (GNNs can generalize!)

Reward function: r

- no consensus

+ imitate strong branching (collect $\mathcal{D} = \{(s, a^*), ...\}$)
+ estimate $\pi^*(a|s)$ from $\mathcal{D}$ by training offline

State Encoding

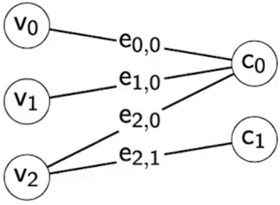
Natural representation: variable/constraint bipartite graph $s_t = (\mathcal{G}, \mathbf{C}, \mathbf{V}, \mathbf{E})$

$$\begin{aligned} &\text{argmin}_x \quad \mathbf{c}^T\mathbf{x} \\ &\text{s.t.} \quad A\mathbf{x} \le \mathbf{b} \\ &\mathbf{l} \le \mathbf{x} \le \mathbf{u} \\ &\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$$\mathbf{x} \to \mathbf{v}$$

$$\mathbf{A}(\mathbf{i}, \mathbf{j}) \to e_{i,j}$$

$$\mathbf{b} \to \mathbf{c}$$

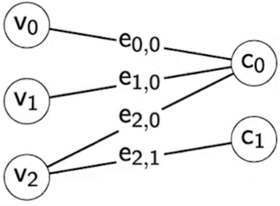


State Encoding

Natural representation: variable/constraint bipartite graph $s_t = (\mathcal{G}, \mathbf{C}, \mathbf{V}, \mathbf{E})$

$$\begin{aligned} &\text{argmin}_x \quad \mathbf{c}^T\mathbf{x} \\ &\text{s.t.} \quad A\mathbf{x} \le \mathbf{b} \\ &\mathbf{l} \le \mathbf{x} \le \mathbf{u} \\ &\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$$\mathbf{x} \to \mathbf{v}$$
$$\mathbf{A}(\mathbf{i}, \mathbf{j}) \to e_{i,j}$$
$$\mathbf{b} \to \mathbf{c}$$



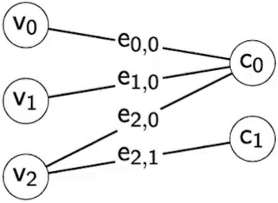
- v_i: variable features (type, coef., bounds, LP solution, ...)

State Encoding

Natural representation: variable/constraint bipartite graph $s_t = (\mathcal{G}, \mathbf{C}, \mathbf{V}, \mathbf{E})$

$$\begin{aligned} \text{argmin}_x \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$$\mathbf{x} \rightarrow \mathbf{v}$$
$$\mathbf{A}(\mathbf{i}, \mathbf{j}) \rightarrow e_{i,j}$$
$$\mathbf{b} \rightarrow \mathbf{c}$$



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

State Encoding

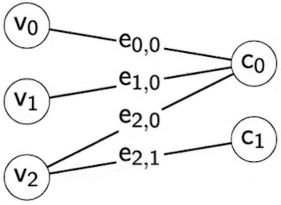
Natural representation: variable/constraint bipartite graph $s_t = (\mathcal{G}, \mathbf{C}, \mathbf{V}, \mathbf{E})$

$$\begin{aligned} \text{argmin}_x \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$$\mathbf{x} \rightarrow \mathbf{v}$$

$$\mathbf{A}(\mathbf{i}, \mathbf{j}) \rightarrow e_{i,j}$$

$$\mathbf{b} \rightarrow \mathbf{c}$$



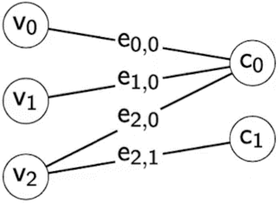
- v_i: variable features (type, coef., bounds, LP solution, ...)
- c_j: constraint features (right-hand-side constraint, ...)
- $e_{i,j}$: non-zero coefficients in A

State Encoding

Natural representation: variable/constraint bipartite graph $s_t = (\mathcal{G}, \mathbf{C}, \mathbf{V}, \mathbf{E})$

$$\begin{aligned} &\mathrm{argmin}_x \quad \mathbf{c}^T\mathbf{x} \\ &\text{s.t.} \quad A\mathbf{x} \leq \mathbf{b} \\ &\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ &\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$$\mathbf{x} \to \mathbf{v}$$
$$\mathbf{A(i,j)} \to e_{i,j}$$
$$\mathbf{b} \to \mathbf{c}$$



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

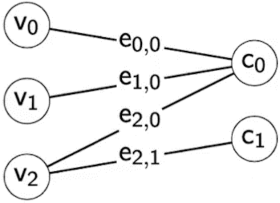
Graph structure is fixed through time during solve (low cost!)

State Encoding

Natural representation: variable/constraint bipartite graph $s_t = (\mathcal{G}, \mathbf{C}, \mathbf{V}, \mathbf{E})$

$$\begin{aligned} &\text{argmin}_x \quad \mathbf{c}^T \mathbf{x} \\ &\text{s.t.} \quad A\mathbf{x} \leq \mathbf{b} \\ &\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ &\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$$\mathbf{x} \rightarrow \mathbf{v}$$
$$\mathbf{A(i,j)} \rightarrow e_{i,j}$$
$$\mathbf{b} \rightarrow \mathbf{c}$$



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

Graph structure is fixed through time during solve (low cost!)

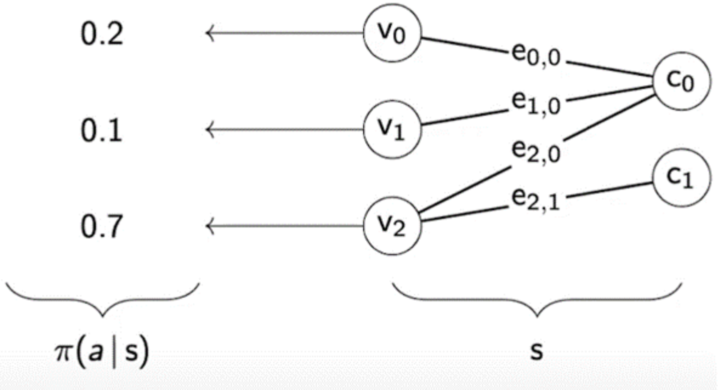
Complete state-information not captured, so this is a partially-observed MDP

+ Several good branching policies do not have access to the full state description

Branching Policy as a GCNN Model

Neighbourhood-based updates: $v_i \leftarrow \sum_{j \in \mathcal{N}_i} f_\theta(v_i, e_{i,j}, c_j)$

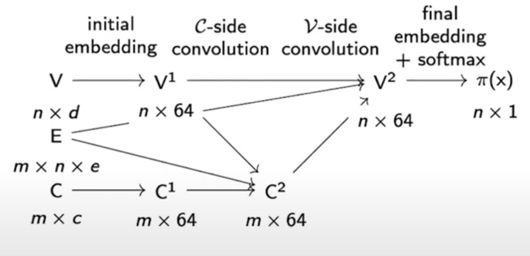
- permutation invariance
- sparse representation



Branching Policy as a GCNN Model

Bipartite graph representation permits graph convolution in the form of two interleaved half-convolutions

$$\mathbf{c_i} \leftarrow f_{\mathbf{C}}\left(\mathbf{c}_i, \sum_{j}^{(i,j)\in\mathbf{E}} g_{\mathbf{C}}(\mathbf{c}_i, \mathbf{v}_j, \mathbf{e}_{i,j})\right) \qquad \mathbf{v_i} \leftarrow f_{\mathbf{V}}\left(\mathbf{v}_i, \sum_{i}^{(i,j)\in\mathbf{E}} g_{\mathbf{v}}(\mathbf{c}_i, \mathbf{v}_j, \mathbf{e}_{i,j})\right)$$



Prenorm Layer: messages are normalized after aggregation to stabilize learning,

$$\mathbf{x} \leftarrow (\mathbf{x} - \beta)/\sigma$$

β and σ are the mean and standard deviation of $\mathbf{x}$ on the training dataset $\mathcal{D}$

Empirical Results

Comparing ML methods for predicting branching variable scores (ranking) of SB

Table 1: Imitation learning accuracy on the test sets.

	Set Covering			Combinatorial Auction			Capacitated Facility Location			Maximum Independent Set		
model	acc@1	acc@5	acc@10	acc@1	acc@5	acc@10	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	52.9±0.3	84.3±0.1	94.1±0.1	63.0±0.4	97.3±0.1	99.9±0.0	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	57.2±0.2	86.9±0.2	95.4±0.1	67.8±0.1	98.1±0.1	99.9±0.0	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	57.3±0.3	86.9±0.2	95.3±0.1	68.0±0.2	98.0±0.0	99.9±0.0	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	**61.6**±0.1	**91.0**±0.1	**97.8**±0.1	**71.2**±0.2	**98.6**±0.1	**99.9**±0.0	**56.5**±0.2	**80.8**±0.3	**89.0**±0.1

Empirical Results

Search tree size and solving time comparison

Table 2: Policy evaluation on separate instances in terms of solving time, number of wins (fastest method) over number of solved instances, and number of resulting B&B nodes (lower is better). For each problem, the models are trained on easy instances only. See Section 5.1 for definitions.

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	17.30 ± 6.1%	0 / 100	17 ±13.7%	411.34 ± 4.3%	0 / 90	171 ± 6.4%	3600.00 ± 0.0%	0 / 0	n/a ± n/a %
RPB	8.98 ± 4.8%	0 / 100	**54** ±20.8%	60.07 ± 3.7%	0 / 100	1741 ± 7.9%	1677.02 ± 3.0%	4 / 65	47 299 ± 4.9%
TREES	9.28 ± 4.9%	0 / 100	187 ± 9.4%	92.47 ± 5.9%	0 / 100	2187 ± 7.9%	2869.21 ± 3.2%	0 / 35	59 013 ± 9.3%
SVMRANK	8.10 ± 3.8%	1 / 100	165 ± 8.2%	73.58 ± 3.1%	0 / 100	1915 ± 3.8%	2389.92 ± 2.3%	0 / 47	42 120 ± 5.4%
LMART	7.19 ± 4.2%	14 / 100	167 ± 9.0%	59.98 ± 3.9%	0 / 100	1925 ± 4.9%	2165.96 ± 2.0%	0 / 54	45 319 ± 3.4%
GCNN	**6.59** ± 3.1%	**85** / 100	134 ± 7.6%	**42.48** ± 2.7%	**100** / 100	**1450** ± 3.3%	**1489.91** ± 3.3%	**66** / 70	**29 981** ± 4.9%

Set Covering

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	4.11 ±12.1%	0 / 100	6 ±30.3%	86.90 ± 12.9%	0 / 100	72 ±19.4%	1813.33 ± 5.1%	0 / 68	400 ± 7.5%
RPB	2.74 ± 7.8%	0 / 100	**10** ±32.1%	17.41 ± 6.6%	0 / 100	689 ±21.2%	136.17 ± 7.9%	13 / 100	5511 ±11.7%
TREES	2.47 ± 7.3%	0 / 100	86 ±15.9%	23.70 ± 11.2%	0 / 100	976 ±14.4%	451.39 ±14.6%	0 / 95	10 290 ±16.2%
SVMRANK	2.31 ± 6.8%	0 / 100	77 ±15.0%	23.10 ± 9.8%	0 / 100	867 ±13.4%	364.48 ± 7.7%	0 / 98	6329 ± 7.7%
LMART	**1.79** ± 6.0%	**75** / 100	77 ±14.9%	14.42 ± 9.5%	1 / 100	873 ±14.3%	222.54 ± 8.6%	0 / 100	7006 ± 6.9%
GCNN	1.85 ± 5.0%	25 / 100	70 ±12.0%	**10.29** ± 7.1%	**99** / 100	**657** ±12.2%	**114.16** ±10.3%	**87** / 100	**5169** ±14.9%

Combinatorial Auction

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	30.36 ±19.6%	4 / 100	14 ±34.5%	214.25 ± 15.2%	1 / 100	76 ±15.8%	742.91 ± 9.1%	15 / 90	55 ± 7.2%
RPB	26.55 ±16.2%	9 / 100	**22** ±31.9%	156.12 ± 11.5%	8 / 100	**142** ±20.6%	631.50 ± 8.1%	14 / 96	**110** ±15.5%
TREES	28.96 ±14.7%	3 / 100	135 ±20.0%	159.86 ± 15.3%	3 / 100	401 ±11.6%	671.01 ±11.1%	1 / 95	381 ±11.1%
SVMRANK	23.58 ±14.1%	11 / 100	117 ±20.5%	130.86 ± 13.6%	13 / 100	348 ±11.4%	586.13 ±10.0%	21 / 95	321 ± 8.8%
LMART	23.34 ±13.6%	16 / 100	117 ±20.7%	128.48 ± 15.4%	23 / 100	349 ±12.9%	582.38 ±10.5%	15 / 95	314 ± 7.0%
GCNN	**22.10** ±15.8%	**57** / 100	107 ±21.4%	**120.94** ± 14.2%	**52** / 100	339 ±11.8%	**563.36** ±10.7%	**30** / 95	338 ±10.9%

Capacitated Facility Location

Model	Easy Time	Easy Wins	Easy Nodes	Medium Time	Medium Wins	Medium Nodes	Hard Time	Hard Wins	Hard Nodes
FSB	23.58 ±29.9%	9 / 100	7 ±35.9%	1503.55 ± 20.9%	0 / 74	38 ±28.2%	3600.00 ± 0.0%	0 / 0	n/a ± n/a %
RPB	8.77 ±11.8%	7 / 100	**20** ±36.1%	**110.99** ± 24.4%	41 / 100	**729** ±37.3%	2045.61 ±18.3%	22 / 42	**2675** ±24.0%
TREES	10.75 ±22.1%	1 / 100	76 ±44.2%	1183.37 ± 34.2%	1 / 47	4664 ±45.8%	3565.12 ± 1.2%	0 / 3	38 296 ± 4.1%
SVMRANK	8.83 ±14.9%	2 / 100	46 ±32.2%	242.91 ± 29.3%	1 / 96	546 ±26.0%	2902.94 ± 9.6%	1 / 18	6256 ±15.1%
LMART	7.31 ±12.7%	30 / 100	52 ±38.1%	219.22 ± 36.0%	15 / 91	747 ±35.1%	3044.94 ± 7.0%	0 / 12	8893 ± 3.5%
GCNN	**6.43** ±11.6%	**51** / 100	43 ±40.2%	192.91 ±110.2%	**42** / 82	1841 ±88.0%	**2024.37** ±30.6%	**25** / 29	2997 ±26.3%

Maximum Independent Set

Ablation Study

Table 3: Ablation study of our GCNN model on the set covering problem. Sum convolutions generalize better to larger instances, especially when combined with a prenorm layer.

	Accuracies			Easy			Medium			Hard		
Model	acc@1	acc@5	acc@10	time	wins	nodes	time	wins	nodes	time	wins	nodes
MEAN	65.4 ±0.1	**92.4** ±0.1	**98.2** ±0.0	6.7 ±3%	13 / 100	134 ±6%	43.7 ±3%	19 / 100	1894 ±4%	1593.0 ±4%	6 / 70	62 227 ±6%
SUM	**65.5** ±0.2	92.3 ±0.2	98.1 ±0.1	**6.6** ±3%	27 / 100	134 ±6%	**42.5** ±3%	**45** / 100	1882 ±4%	1511.7 ±3%	22 / 70	57 864 ±4%
GCNN	**65.5** ±0.1	**92.4** ±0.1	**98.2** ±0.0	**6.6** ±3%	**60** / 100	134 ±8%	**42.5** ±3%	36 / 100	**1870** ±3%	**1489.9** ±3%	**42** / 70	**56 348** ±5%

- sum aggregation is more expressive than mean aggregation
- prenorm layers stabilize learning and permits better generalization to harder instances

Related Work

- Khalil et al. (2016) attempts to learn branching rules during B&B customized to a single instance (trained online)
  - *Learning to branch in mixed integer programming.*
- Alvarez et al (2017) treats problem of variable selection as a regression problem on variable scores (trained offline)
  - *A machine learning-based approximation of strong branching.*
- Hansknecht et al. (2018) treats problem of variable selection as a ranking problem, learning a partial ordering of candidates from an expert policy
  - *Cuts, primal heuristics, and learning to branch for the time-dependent traveling salesman problem.*

Gasse et al. in this work:

+ learns variable selection heuristics for branching in an offline manner
+ treats the problem as classification and can learn from expert rules that do not produce variable rankings
+ The approach generalizes to harder instances, competitive with traditional MILP solvers

Conclusions

Heuristic vs. data-driven branching:

+ tune B&B to problem families of interest (e.g. TSP)
+ can generalize to harder problems
– no guarantees outside training distribution
– requires training instances $\mathcal{D}$ (NP-hard problems)
– will never surpass expert wrt. decision quality
+ but may be able to solve instances faster