Reinforcement Learning for Integer Programming: Learning to Cut review

강성호

Tang, Yunhao, Shipra Agrawal, and Yuri Faenza. "Reinforcement learning for integer programming: Learning to cut." *International Conference on Machine Learning*. PMLR, 2020.

정수 계획법 (Integer Programming, IP)

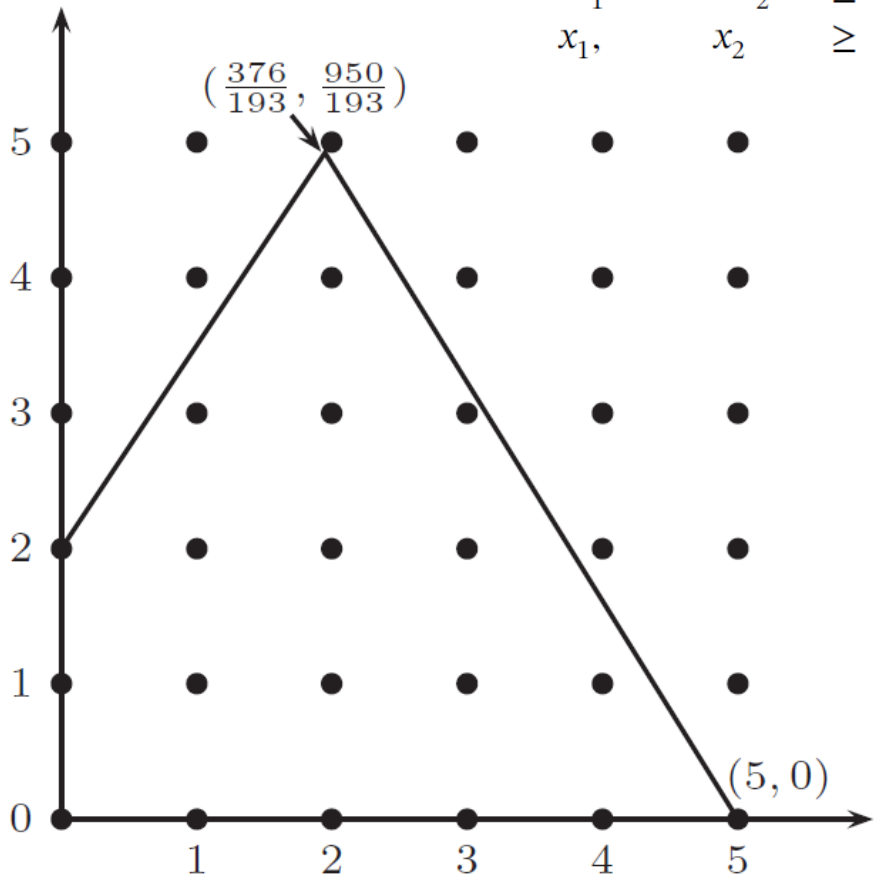
Integer programming. It is well known that any Integer Program (IP) can be written in the following *canonical form*

$$\min\{c^T x \,:\, Ax \leq b,\, x \geq 0,\, x \in \mathbb{Z}^n\} \qquad (1)$$

where x is the set of n decision variables, $Ax \leq b, x \geq 0$ with $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m$ formulates the set of constraints, and the linear objective function is $c^T x$ for some $c \in \mathbb{Q}^n$.

선형 계획법 (Linear Programming, LP)와 IP 차이

$$
\begin{array}{lrcrcl}
\max & 1.00x_1 & & +0.64x_2 & & \\
 & 50x_1 & & +31x_2 & \leq & 250 \\
 & 3x_1 & & -2x_2 & \geq & -4 \\
 & x_1, & & x_2 & \geq & 0
\end{array}
$$



$$
\begin{array}{lrcrcl}
\max & 1.00x_1 & & +0.64x_2 & & \\
 & 50x_1 & & +31x_2 & \leq & 250 \\
 & 3x_1 & & -2x_2 & \geq & -4 \\
 & x_1, & & x_2 & \geq & 0 \text{ and integer.}
\end{array}
$$

Taha, Hamdy A. Operations research: an introduction. Vol. 790. Upper Saddle River, NJ, USA: Pearson/Prentice Hall, 2011.

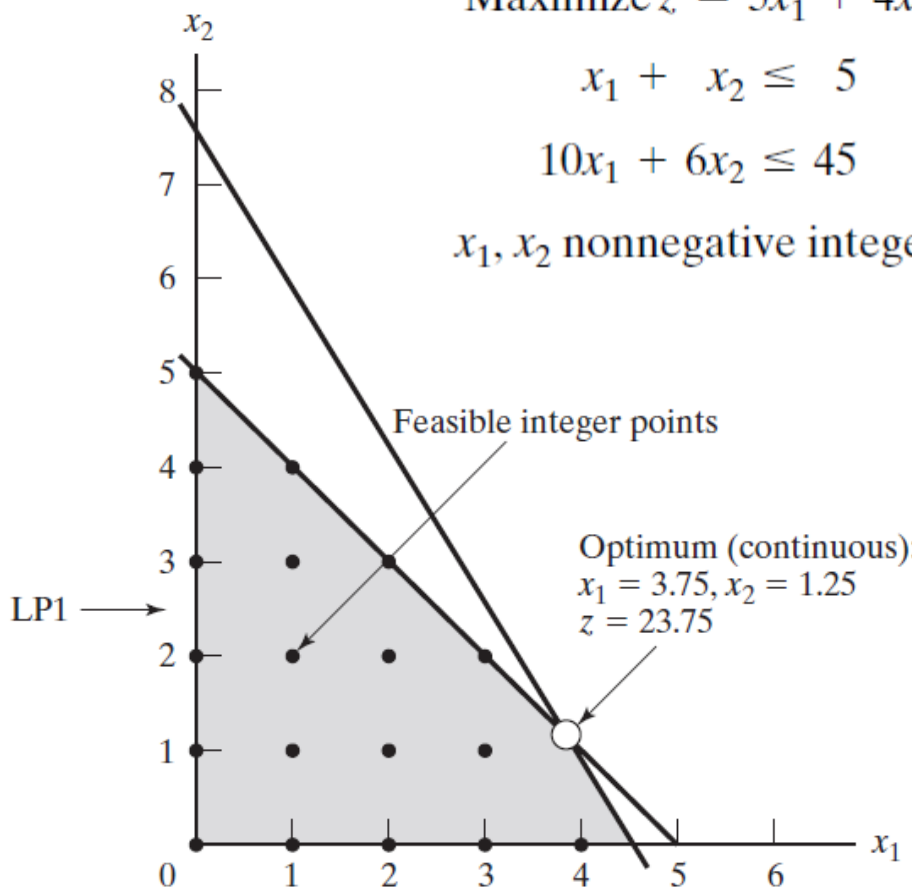
Branch and Bound (B&B, 분지 한계법)

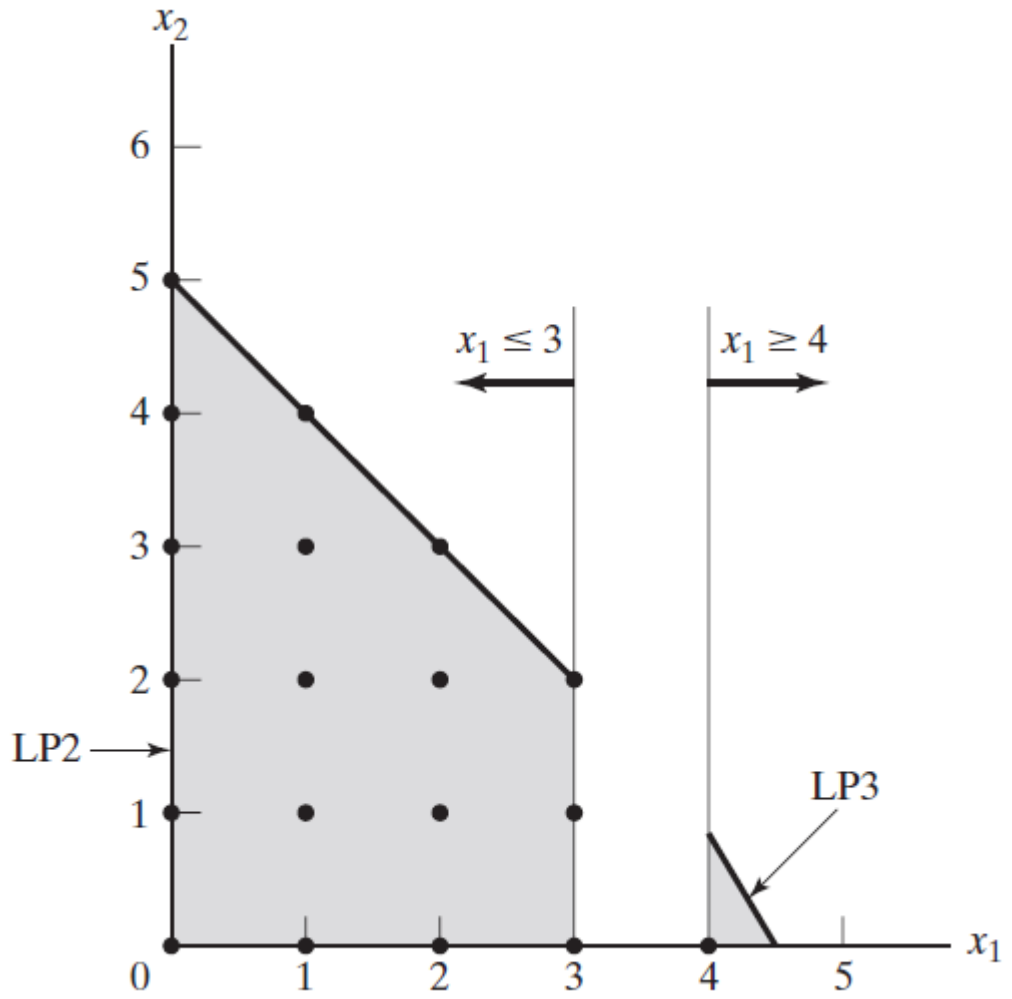
$$\text{Maximize } z = 5x_1 + 4x_2$$

$$x_1 + x_2 \leq 5$$

$$10x_1 + 6x_2 \leq 45$$

x_1, x_2 nonnegative integer

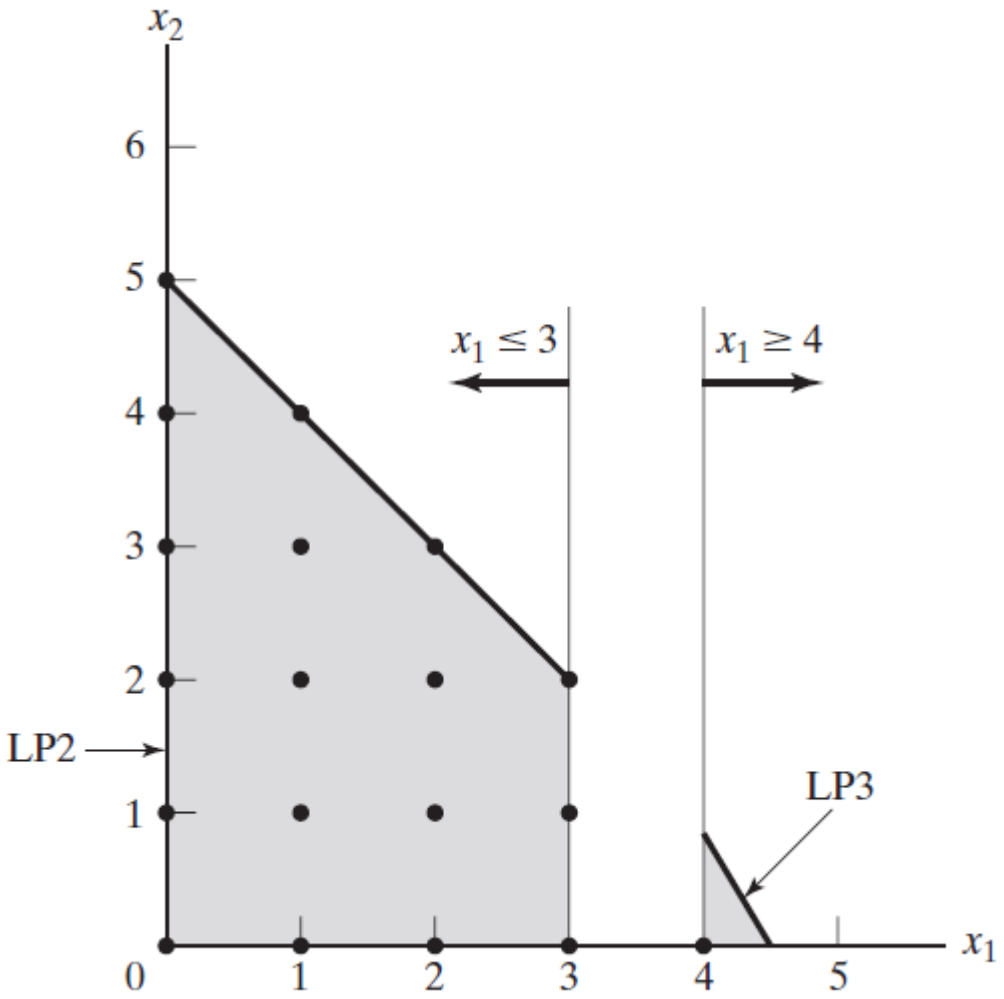


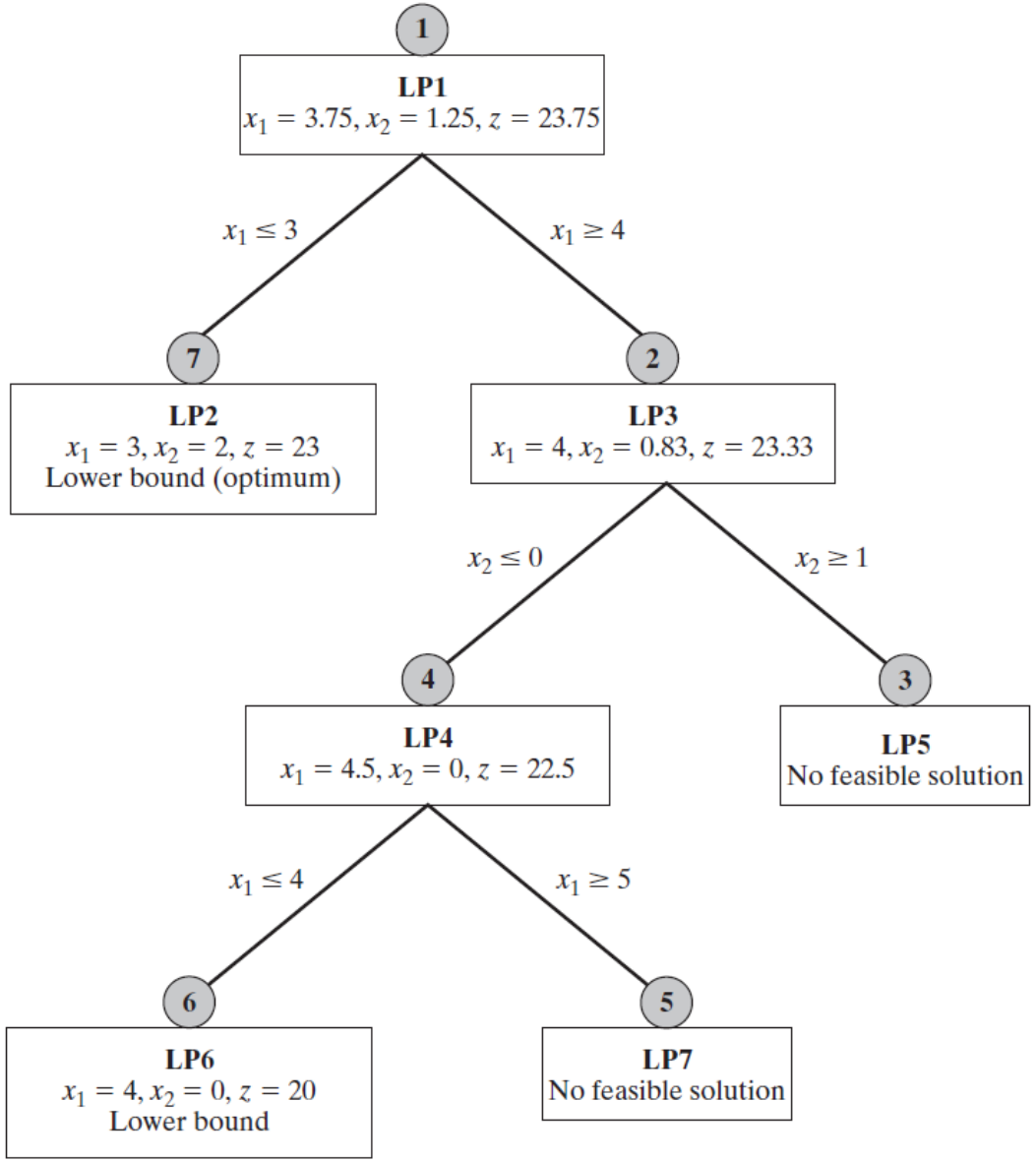


Taha, Hamdy A. Operations research: an introduction. Vol. 790. Upper Saddle River, NJ, USA: Pearson/Prentice Hall, 2011.

Introduction

Branch and Bound (B&B, 분지 한계법)

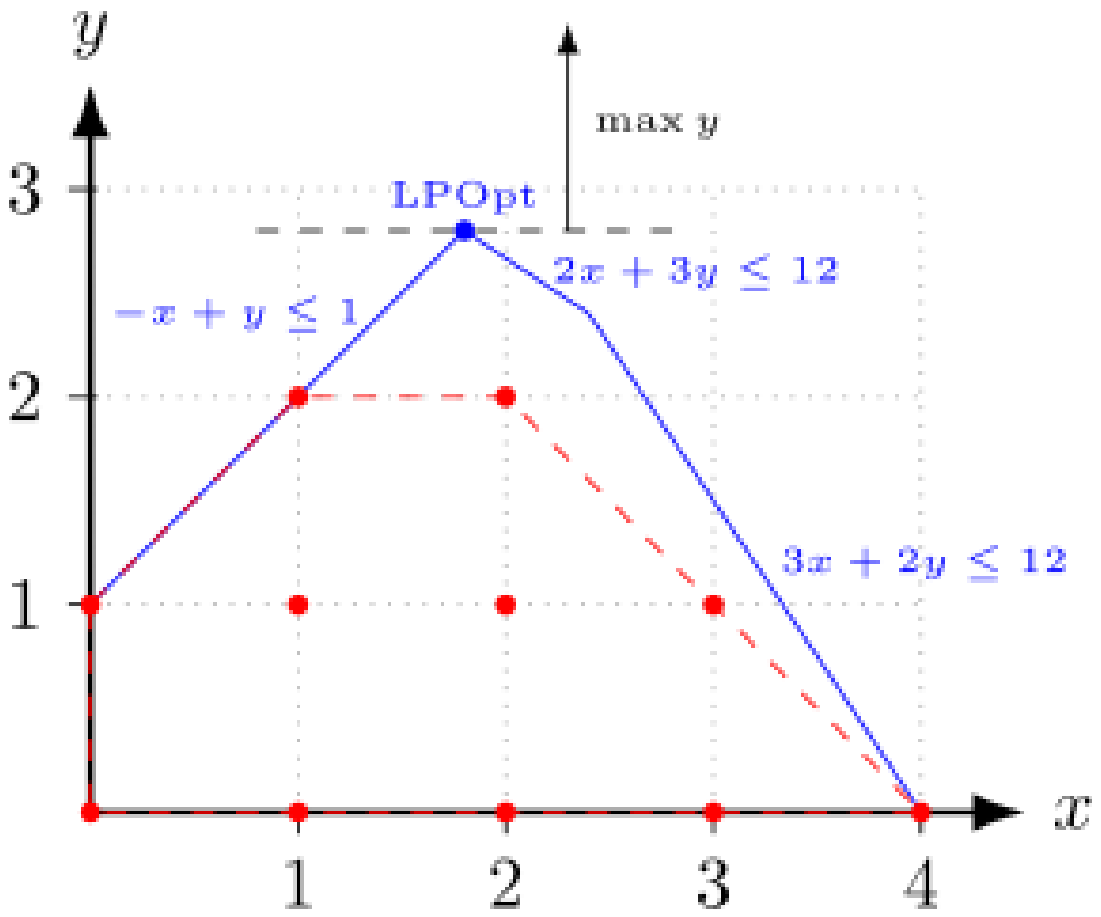


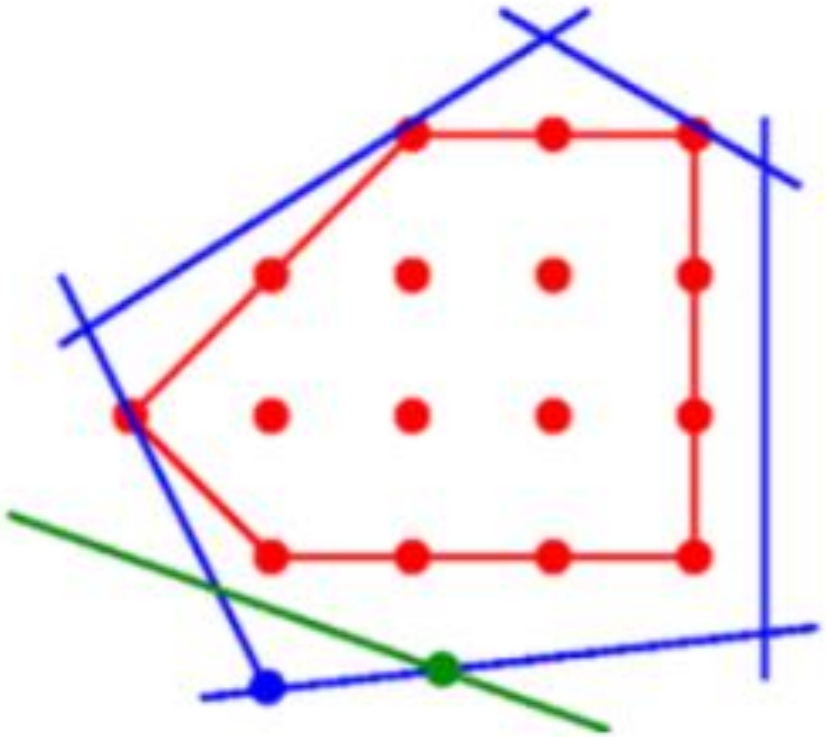


Taha, Hamdy A. Operations research: an introduction. Vol. 790. Upper Saddle River, NJ, USA: Pearson/Prentice Hall, 2011.

Introduction

LP relaxation

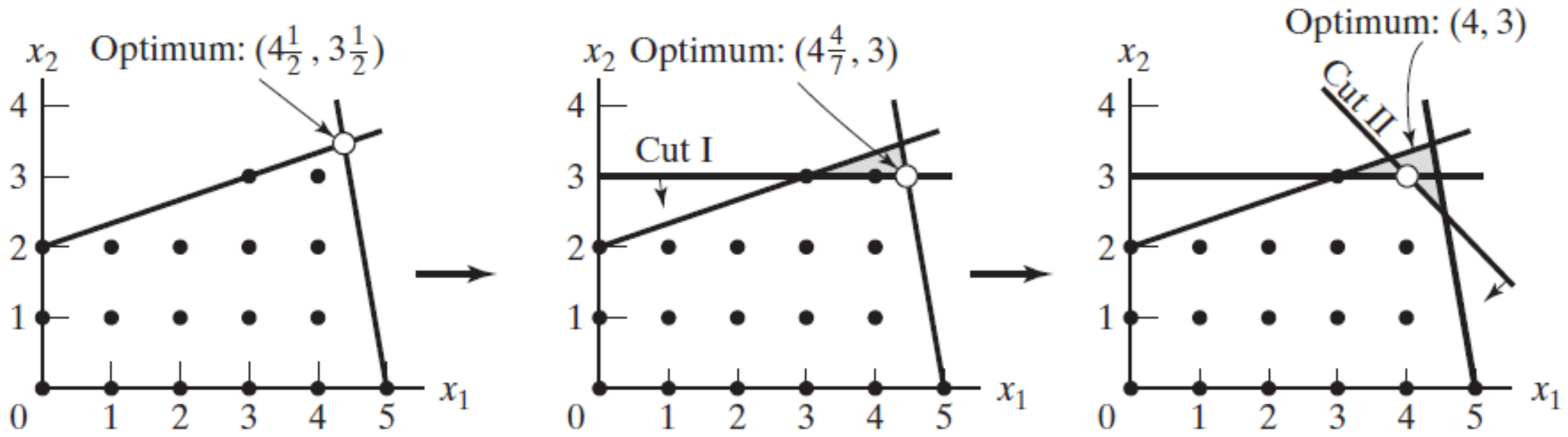




https://en.wikipedia.org/wiki/Linear_programming_relaxation

Introduction

Cutting plane method



$$\text{Maximize } z = 7x_1 + 10x_2$$

$$-x_1 + 3x_2 \leq 6$$

$$7x_1 + x_2 \leq 35$$

$$x_1, x_2 \geq 0 \text{ and integer}$$

Taha, Hamdy A. Operations research: an introduction. Vol. 790. Upper Saddle River, NJ, USA: Pearson/Prentice Hall, 2011.

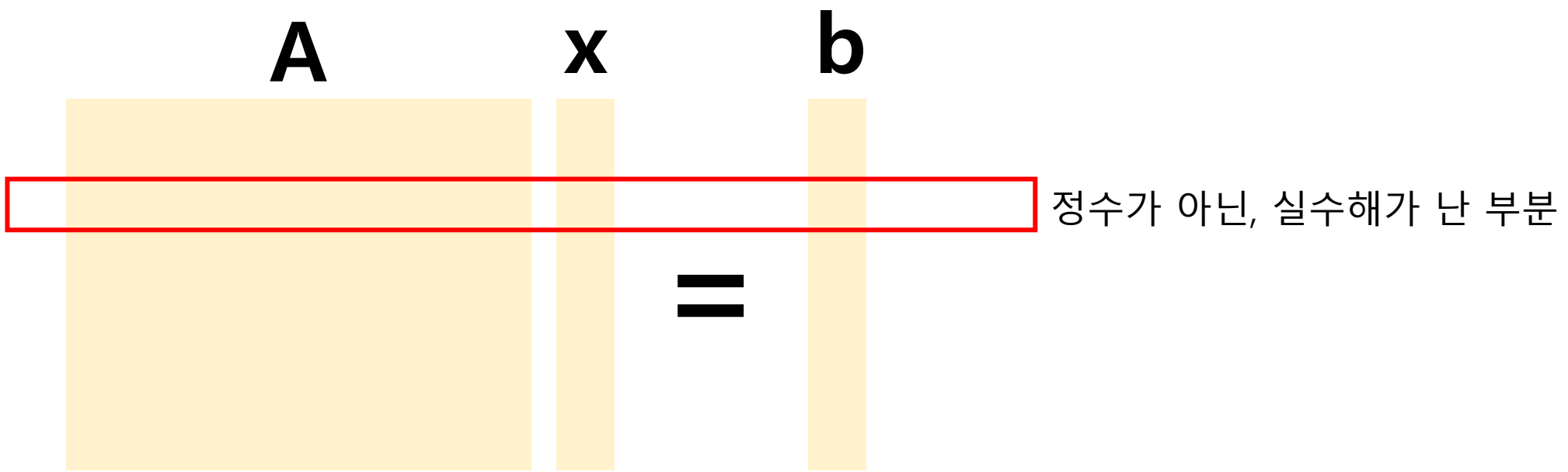
Introduction

Gomory's cutting plane

we can generate a *Gomory cut* (Gomory, 1960)

$$(-\tilde{A}_{(i)} + \lfloor \tilde{A}_{(i)} \rfloor)^T x \leq -\tilde{b}_i + \lfloor \tilde{b}_i \rfloor,$$

$$Ax \leq b,\ x \geq 0$$



Introduction

Gomory's cutting plane

we can generate a *Gomory cut* (Gomory, 1960)

$$(-\tilde{A}_{(i)} + \lfloor \tilde{A}_{(i)} \rfloor)^T x \leq -\tilde{b}_i + \lfloor \tilde{b}_i \rfloor,$$

If x^* is not integer, there exist a row u with $\bar{a}_{u0} \notin Z^1$. For such a row, the Chvátal-Gomory cut is

$$x_{B_u} + \sum_{j \in NB} \lfloor \bar{a}_{uj} \rfloor x_j \leq \lfloor \bar{a}_{u0} \rfloor$$

$$\sum_{j \in NB} (\bar{a}_{uj} - \lfloor \bar{a}_{uj} \rfloor) x_j \geq \bar{a}_{u0} - \lfloor \bar{a}_{u0} \rfloor \quad (\because x_{B_u} + \sum_{j \in NB} \bar{a}_{uj} x_j = \bar{a}_{u0})$$

$\sum_{j \in NB} f_{uj} x_j \geq f_{u0}$ where $f_{uj} = \bar{a}_{uj} - \lfloor \bar{a}_{uj} \rfloor$ for $j \in NB$, and $f_{u0} = \bar{a}_{u0} - \lfloor \bar{a}_{u0} \rfloor$

Wolsey, Laurence A. Integer programming. John Wiley & Sons, 2020.

Introduction

Branch and cut algorithm

1. Add the initial ILP to L, the list of active problems
2. Set $x^* = \text{null}$ and $v^* = -\infty$
3. *while* L is not empty
    1. Select and remove (de-queue) a problem from L
    2. Solve the LP relaxation of the problem.
    3. If the solution is infeasible, go back to 3 (while). Otherwise denote the solution by x with objective value v.
    4. If $v \leq v^*$, go back to 3.
    5. If x is integer, set $v^* \leftarrow v, x^* \leftarrow x$ and go back to 3.
    6. If desired, search for cutting planes that are violated by x. If any are found, add them to the LP relaxation and return to 3.2.
    7. Branch to partition the problem into new problems with restricted feasible regions. Add these problem to L and go back to 3
4. return x^*

https://en.wikipedia.org/wiki/Branch_and_cut

RL framework

State space

$$s_t = \{\mathcal{C}^{(t)}, c, x^*_{\mathrm{LP}}(t), \mathcal{D}^{(t)}\}.$$

$C^{(t)}$: t 시점까지 생성된 feasible region

C: 목적함수의 계수

$x^*_{LP}(t)$: t 시점의 최적해

D(t): t 시점의 gomory's cut

https://en.wikipedia.org/wiki/Branch_and_cut

RL framework

Action space

represented as an inequality $e_i^T x \leq d_i$, and therefore is parameterized by $e_i \in \mathbb{R}^n, d_i \in \mathbb{R}$.

➡ **Gomory cut의 후보들의 집합**

https://en.wikipedia.org/wiki/Branch_and_cut

RL framework

Reward

Reward r_t**.** To encourage adding cutting plane aggressively, we set the instant reward in iteration t to be the gap between objective values of consecutive LP solutions, that is, $r_t = c^T x^*_{\text{LP}}(t+1) - c^T x^*_{\text{LP}}(t) \geq 0$. With a discount factor $\gamma < 1$, this encourages the agent to reduce the integrality gap and approach the integer optimal solution as fast as possible.

https://en.wikipedia.org/wiki/Branch_and_cut

RL framework

Transition

$$s_t = \{\mathcal{C}^{(t)}, c, x^*_{\mathrm{LP}}(t), \mathcal{D}^{(t)}\},$$

$$s_{t+1} = \{\mathcal{C}^{(t+1)}, c, x^*_{\mathrm{LP}}(t+1), \mathcal{D}^{(t+1)}\}.$$

$x^*_{\mathrm{LP}}(t)$ are integer-valued, s_t is a terminal state and $\mathcal{D}^{(t)}$ is an empty set.

RL framework

Policy network

Attention network for order-agnostic cut selection. Given current LP constraints in $\mathcal{C}^{(t)}$, when computing distributions over the I_t candidate constraints in $\mathcal{D}^{(t)}$, it is desirable that the architecture is agnostic to the ordering among the constraints (both in $\mathcal{C}^{(t)}$ and $\mathcal{D}^{(t)}$), because the ordering does not reflect the geometry of the feasible set. To achieve this, we adopt ideas from the attention network (Vaswani et al., 2017). We use a parametric function $F_\theta : \mathbb{R}^{n+1} \mapsto \mathbb{R}^k$ for some given k (encoded by a network with parameter θ). This function is used to compute projections $h_i = F_\theta([a_i, b_i]), i \in [N_t]$ and $g_j = F_\theta([e_j, d_j]), j \in [I_t]$ for each inequality in $\mathcal{C}^{(t)}$ and $\mathcal{D}^{(t)}$, respectively. Here $[\cdot, \cdot]$ denotes concatenation. The score S_j for every candidate cut $j \in [I_t]$ is computed as

$$S_j = \frac{1}{N_t} \sum_{i=1}^{N_t} g_j^T h_i \tag{4}$$

Intuitively, when assigning these scores to the candidate cuts, (4) accounts for each candidate's interaction with all the constraints already in the LP through the inner products $g_j^T h_i$. We then define probabilities $p_1, \ldots, p_{I_t}$ by a softmax function $\text{softmax}(S_1, \ldots, S_{I_t})$. The resulting I_t-way categorical distribution is the distribution over actions given by policy $\pi_\theta(\cdot|s_t)$ in the current state s_t.

LSTM network for variable sized inputs. We want our RL agent to be able to handle IP instances of different sizes (number of decision variables and constraints). Note that the number of constraints can vary over different iterations of a cutting plane method even for a fixed IP instance. But this variation is not a concern since the attention network described above handles that variability in a natural way. To be able to use the same policy network for instances with different number of variables , we embed each constraint using a LSTM network LSTM_θ (Hochreiter and Schmidhuber, 1997) with hidden state of size $n+1$ for a fixed n. In particular, for a general constraint $\tilde{a}_i^T \tilde{x} \leq \tilde{b}_i$ with $\tilde{a}_i \in \mathbb{R}^{\tilde{n}}$ with $\tilde{n} \neq n$, we carry out the embedding $\tilde{h}_i = \text{LSTM}_\theta([\tilde{a}_i, \tilde{b}_i])$ where $\tilde{h}_i \in \mathbb{R}^{n+1}$ is the last hidden state of the LSTM network. This hidden state $\tilde{h}_i$ can be used in place of $[\tilde{a}_i, \tilde{b}_i]$ in the attention network. The idea is that the hidden state $\tilde{h}_i$ can properly encode all information in the original inequalities $[\tilde{a}_i, \tilde{b}_i]$ if the LSTM network is powerful enough.

RL framework

Policy network

Algorithm 1 Rollout of the Policy

Input: policy network parameter θ, IP instance parameterized by c, A, b, number of iterations T.
Initialize iteration counter $t = 0$.
Initialize minimization LP with constraints $\mathcal{C}^{(0)} = \{Ax \leq b\}$ and cost vector c. Solve to obtain $x^*_{\text{LP}}(0)$. Generate set of candidate cuts $\mathcal{D}^{(0)}$.
while $x^*_{\text{LP}}(t)$ not all integer-valued and $t \leq T$ **do**
    Construct state $s_t = \{\mathcal{C}^{(t)}, c, x^*_{\text{LP}}(t), \mathcal{D}^{(t)}\}$.
    Sample an action using the distribution over candidate cuts given by policy π_θ, as $a_t \sim \pi_\theta(\cdot|s_t)$. Here the action a_t corresponds to a cut $\{e^T x \leq d\} \in \mathcal{D}^{(t)}$.
    Append the cut to the constraint set, $\mathcal{C}^{(t+1)} = \mathcal{C}^{(t)} \cup \{e^T x \leq d\}$. Solve for $x^*_{\text{LP}}(t+1)$. Generate $\mathcal{D}^{(t+1)}$.
    Compute reward r_t.
    $t \leftarrow t + 1$.
end while

[Policy update]

update $\theta \leftarrow \theta + \hat{g}_\theta$ for some $\alpha > 0$. The gradient estimator takes the following form

$$\hat{g}_\theta = \frac{1}{N} \sum_{i=1}^{N} J(\pi_{\theta'_i}) \frac{\epsilon_i}{\sigma}, \tag{5}$$

where $\epsilon_i \sim \mathcal{N}(0, \mathbb{I})$ is a sample from a multivariate Gaussian, $\theta'_i = \theta + \sigma\epsilon_i$ and $\sigma > 0$ is a fixed constant. Here the return $J(\pi_{\theta'})$ can be estimated as $\sum_{t=0}^{T-1} r_t \gamma^t$ using a single trajectory (or average over multiple trajectories) generated on executing the policy $\pi_{\theta'}$, as in Algorithm 1. To train the

Experiments

View point

1. **Efficiency of cuts.** Can the RL agent solve an IP problem using fewer number of Gomory cuts?
2. **Integrality gap closed.** In cases where cutting planes alone are unlikely to solve the problem to optimality, can the RL agent close the integrality gap effectively?
3. **Generalization properties.**
   - (size) Can an RL agent trained on smaller instances be applied to 10X larger instances to yield performance comparable to an agent trained on the larger instances?
   - (structure) Can an RL agent trained on instances from one class of IPs be applied to a very different class of IPs to yield performance comparable to an agent trained on the latter class?
4. **Impact on the efficiency of B&C.** Will the RL agent trained as a cutting plane method be effective as a subroutine within a B&C method?
5. **Interpretability of cuts: the knapsack problem.** Does RL have the potential to provide insights into effective and meaningful cutting plane strategies for specific problems? Specifically, for the knapsack problem, do the cuts learned by RL resemble lifted cover inequalities?

Experiments

View point: Efficiency of cuts (small-sized instances).

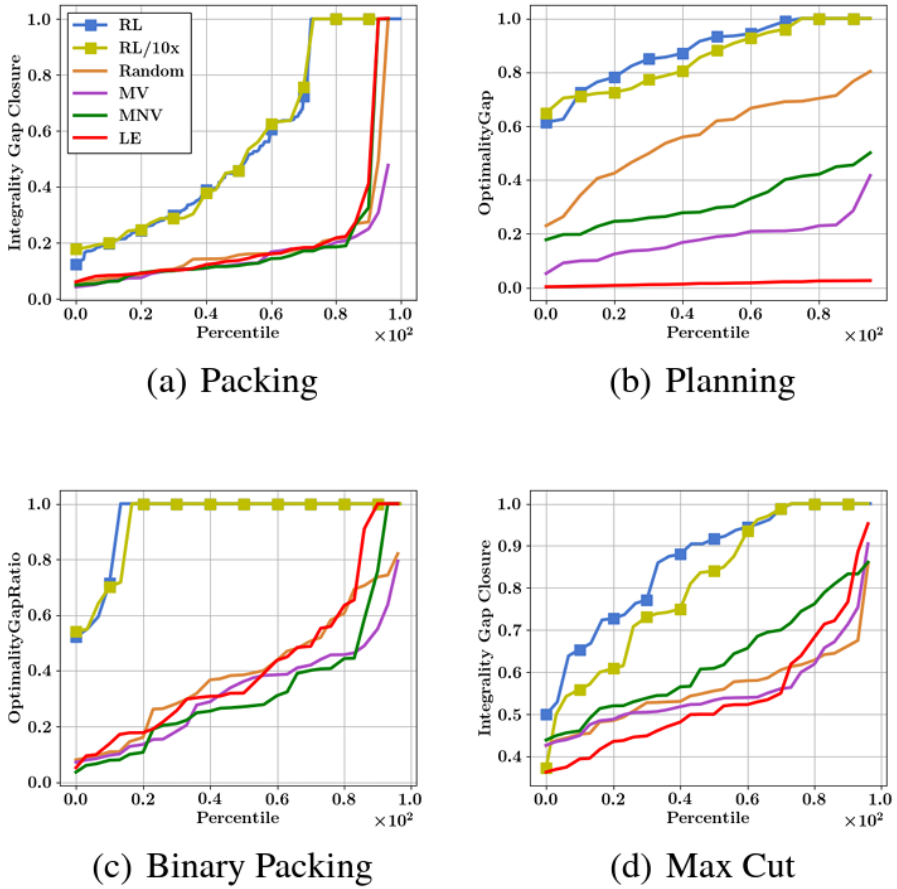
Table 1: Number of cuts it takes to reach optimality. We show mean $\pm$ std across all test instances.

Tasks	Packing	Planning	Binary	Max Cut
Size	10×5	13×20	10×20	10×22
RANDOM	48 ± 36	44 ± 37	81 ± 32	69 ± 34
MV	62 ± 40	48 ± 29	87 ± 27	64 ± 36
MNV	53 ± 39	60 ± 34	85 ± 29	47 ± 34
LE	34 ± 17	310 ± 60	89 ± 26	59 ± 35
RL	$\mathbf{14 \pm 11}$	$\mathbf{10 \pm 12}$	$\mathbf{22 \pm 27}$	$\mathbf{13 \pm 4}$

Random, Max Violation (MV), Max Normalized Violation (MNV) and Lexicographical Rule (LE).

Experiments

View point: Integrality gap closure for large-sized instances.



(a) Packing (b) Planning

(c) Binary Packing (d) Max Cut

Random, Max Violation (MV), Max Normalized Violation (MNV) and Lexicographical Rule (LE).

This yellow curve is for RL/10X, which is an RL agent trained on 10X smaller instances in order to evaluate generalization properties, as we describe next.

Figure 2: Percentile plots of IGC for test instances of size roughly 1000. X-axis shows the percentile of instances and y-axis shows the IGC achieved on adding $T = 50$ cuts. Across all test instances, RL achieves significantly higher IGC than the baselines.

Experiments

View point: Integrality gap closure for large-sized instances.

Random, Max Violation (MV), Max Normalized Violation (MNV) and Lexicographical Rule (LE).

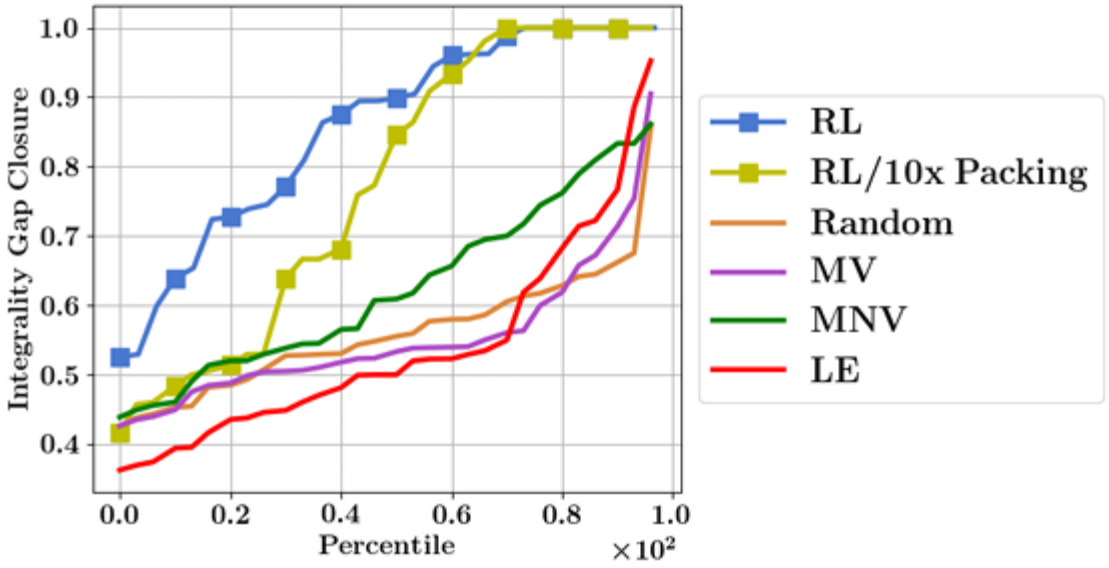
Table 3: IGC for test instances of size roughly 5000. We show mean $\pm$ std of IGC achieved on adding $T = 250$ cuts.

Tasks	**Packing**	**Planning**	**Binary**	**Max Cut**
Size	60×60	121×168	66×132	54×134
RANDOM	0.05 ± 0.03	0.38 ± 0.08	0.17 ± 0.12	0.50 ± 0.10
MV	0.04 ± 0.02	0.07 ± 0.03	0.19 ± 0.18	0.50 ± 0.06
MNV	0.05 ± 0.03	0.17 ± 0.10	0.19 ± 0.18	0.56 ± 0.11
LE	0.04 ± 0.02	0.01 ± 0.01	0.23 ± 0.20	0.45 ± 0.08
RL	$\mathbf{0.11 \pm 0.05}$	$\mathbf{0.68 \pm 0.10}$	$\mathbf{0.61 \pm 0.35}$	$\mathbf{0.57 \pm 0.10}$

Experiments

View point: Generalization.

small sized instances of the packing problem, and applying it to add cuts to 10X larger instances of the maximum-cut problem.



Random, Max Violation (MV), Max Normalized Violation (MNV) and Lexicographical Rule (LE).

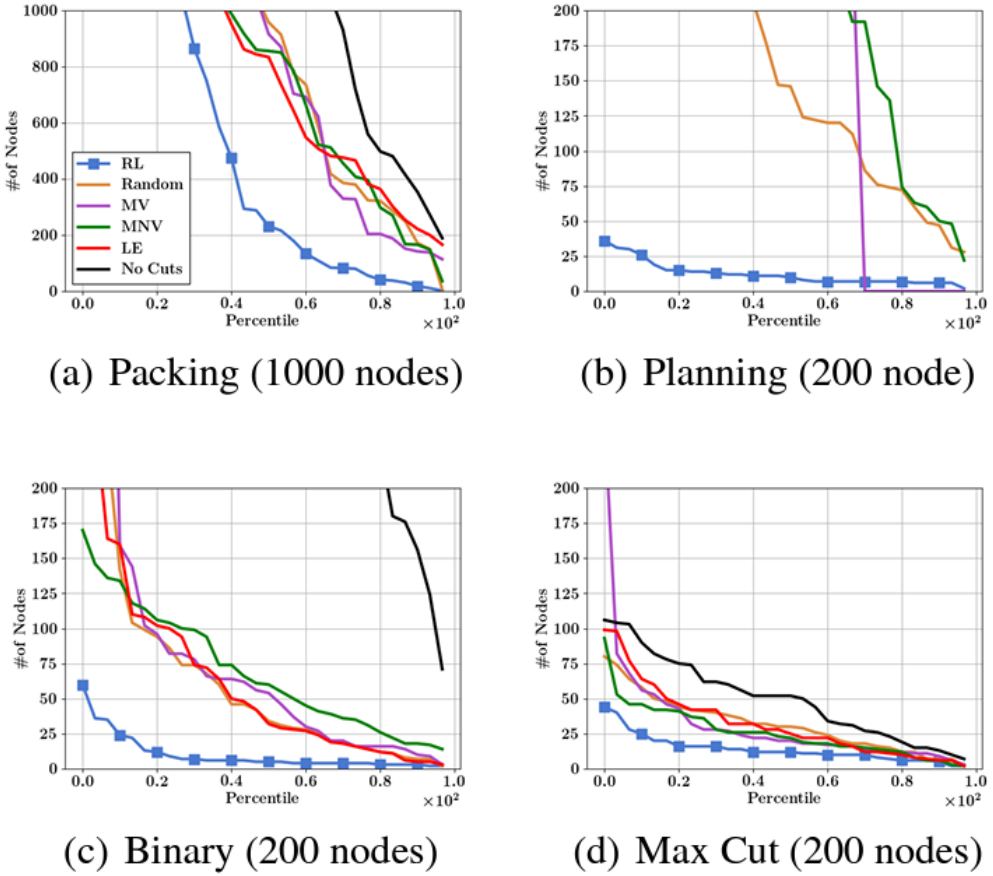
Figure 4: Percentile plots of Integrality Gap Closure. 'RL/10X packing' trained on instances of a completely different IP problem (packing) performs competitively on the maximum-cut instances.

Experiments

View point: Impact on the efficiency of B&C.

Table 4: IGC in B&C. We show mean ± std across test instances.

Tasks	Packing	Planning	Binary	Max Cut
Size	30×30	61×84	33×66	27×67
NO CUT	0.57±0.34	0.35±0.08	0.60±0.24	1.0 ± 0.0
RANDOM	0.79±0.25	0.88±0.16	0.97±0.09	1.0 ± 0.0
MV	0.67±0.38	0.64±0.27	0.97±0.09	0.97±0.18
MNV	0.83±0.23	0.74±0.22	1.0 ± 0.0	1.0 ± 0.0
LE	0.80±0.26	0.35±0.08	0.97±0.08	1.0 ± 0.0
RL	$\mathbf{0.88 \pm 0.23}$	$\mathbf{1.0 \pm 0.0}$	1.0 ± 0.0	1.0 ± 0.0



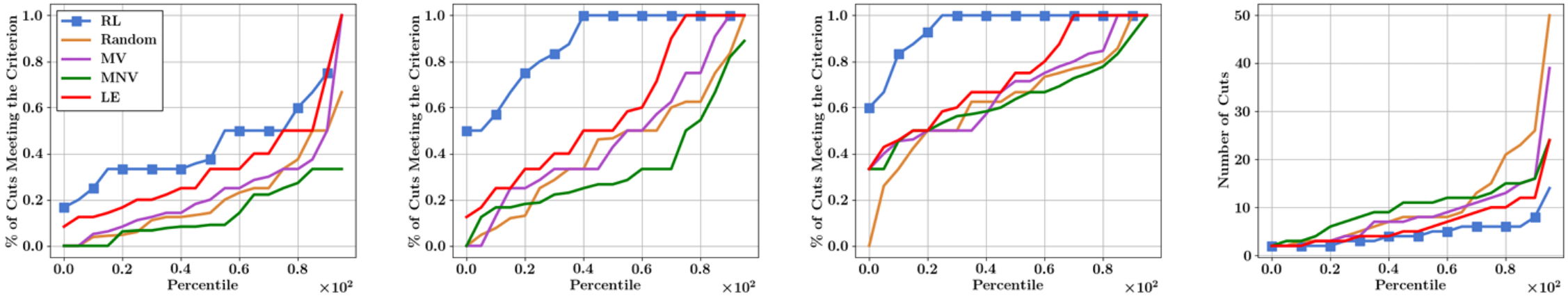
(a) Packing (1000 nodes) (b) Planning (200 node)

(c) Binary (200 nodes) (d) Max Cut (200 nodes)

Figure 5: Percentile plots of number of B&C nodes expanded. X-axis shows the percentile of instances and y-axis shows the number of expanded nodes to close 95% of the integrality gap.

Experiments

View point: Interpretability of cuts.



(a) Criterion 1 (b) Criterion 2 (c) Criterion 3 (d) Number of cuts

Figure 6: Percentage of cuts meeting the designed criteria and number of cuts on Knapsack problems. We train the RL agent on 80 instances. All baselines are tested on 20 instances. As seen above, RL produces consistently more 'high-quality' cuts.

Q&A
감사합니다.