Reinforcement Learning for Integer Programming: Learning to Cut

CS885: Paper Presentation

Authors: Yunhao Tang, Shipra Agrawal, Yuri Faenza
Presenter: Mohsin Hasan

Tang, Y., Agrawal, S. amp; Faenza, Y.. (2020). Reinforcement Learning for Integer Programming: Learning to Cut. *Proceedings of the 37th International Conference on Machine Learning*, in *Proceedings of Machine Learning Research* 119:9367-9376 Available from https://proceedings.mlr.press/v119/tang20a.html.

Table of Contents

Table of Contents

Introduction

This paper is about solving Integer Programming problems (a class of combinatorial optimization problems).

Introduction

This paper is about solving Integer Programming problems (a class of combinatorial optimization problems).
It solves them by using an RL agent to make heuristic choices, within a classic/hard-coded algorithm.

Integer Programming

Integer Programming (IP) is the optimization problem:

Integer Programming Problem

$$\min\ c^\top x$$
$$\text{Subject to } Ax \leq b,\ x \geq 0,\ x \in \mathbb{Z}^n$$

Integer Programming

Integer Programming (IP) is the optimization problem:

Integer Programming Problem

$$\min \; c^\top x$$
$$\text{Subject to } Ax \leq b, \; x \geq 0, \; x \in \mathbb{Z}^n$$

- $A, \; b, \; c$ have rational components ($\in \mathbb{Q}$)
- The x satisfying the constraints are said to form the **feasible set**.
- Denote the optimum as x^*_{IP}, with minimum objective $c^\top x^*_{IP}$

Why IPs?

Many difficult combinatorial problems boil down to this form:

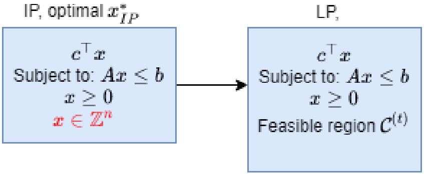
- Graph optimization (eg. traveling salesman problem)
- Scheduling
- Production Planning

The problem is (NP-)hard, and algorithms tend to use heuristics

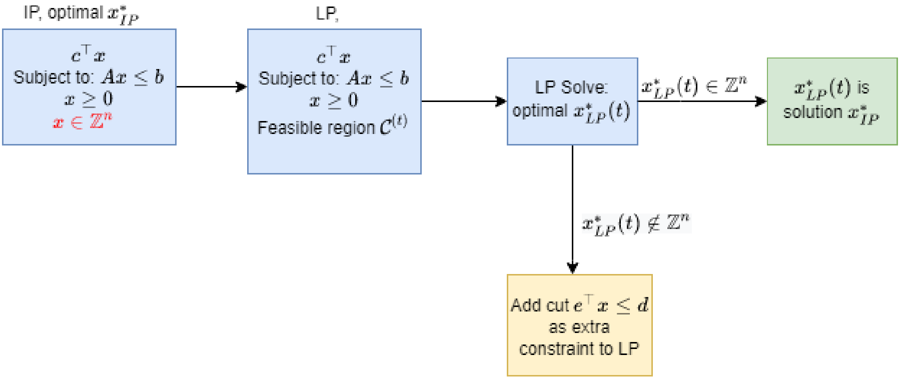
Cutting Plane Method

- One method used by commercial IP solvers
- Used on its own, or as a subroutine in a larger algorithm (called B&C - Branch and Cut)
- Idea is to convert to a linear program, and successively shrink the feasible set

Method: First, we drop integer constraints (results in easy to solve LP - Linear Program)

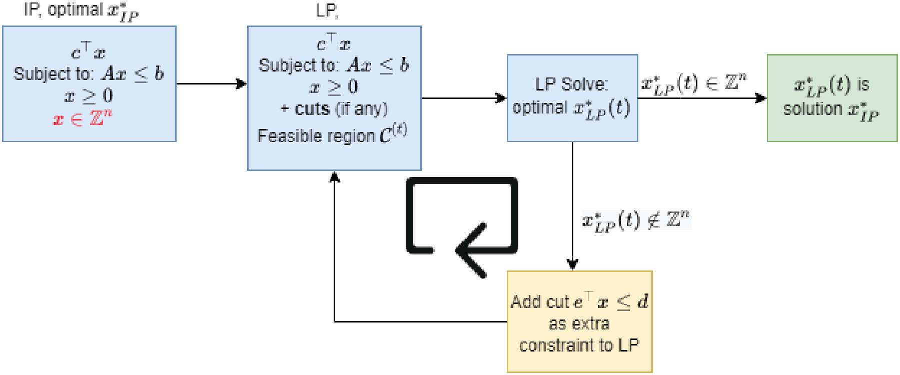


Next, solve the LP, and check if components are integers:



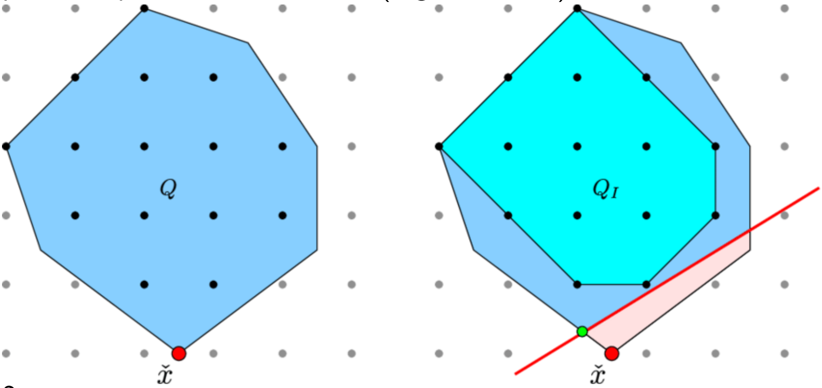
Cutting Plane Method 3

If they aren't, add an extra linear constraint (**called a cutting plane, or cut**) to the problem, and repeat:



Cutting Plane Visualization

A **cut** excludes the previous LP solution $x^*_{LP}(t)$, but keeps all integer points in previous feasible set. (Figure from 2)



2: Achterberg, Tobias. (2009). SCIP: Solving constraint integer programs. Mathematical Programming Computation. 1. 1-41.

Selection of Cuts

Different algorithms propose different cut choices:

- *Gomory's* method proposes a possible cut choice for every non-integer component.
- Cut choices formed using information extracted in solving the LP ("tableau matrix" $\tilde{A}$)
- **It proposes $l_t \leq n$ possible cuts to choose from.**

Typically, **heuristics** are used to select which (single) cut to apply

Selection of Cuts

Different algorithms propose different cut choices:

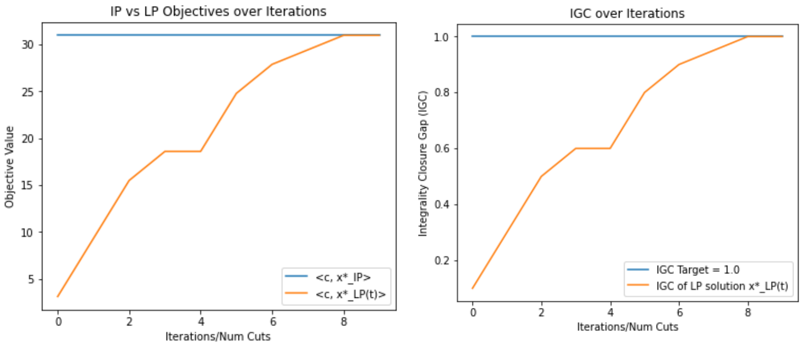
- *Gomory's* method proposes a possible cut choice for every non-integer component.
- Cut choices formed using information extracted in solving the LP ("tableau matrix" $\tilde{A}$)
- **It proposes $l_t \leq n$ possible cuts to choose from.**

Typically, **heuristics** are used to select which (single) cut to apply

In paper: RL agent is trained to select cuts (given by Gomory's method).

Cut Quality Metric

Over the iterations, the (LP's) feasible set $\mathcal{C}^{(t)}$ reduces, and the objective $c^\top x^*_{LP}(t)$ increases to IP optimal $c^\top x^*_{IP}$ (left plot):



Integrality Gap Closure (IGC) rescales axis to $[0, 1]$ (right plot). Cut heuristics are evaluated based on IGC (closer to 1 is better).

Table of Contents

Setting Up MDP: States

The state is the LP problem in iteration t, specified by $s_t = \{\mathcal{C}^{(t)},\ c,\ x^*_{LP}(t),\ \mathcal{D}^{(t)}\}$:

- Feasible region $\mathcal{C}^{(t)} = \{a_i^\top x \leq b_i\}_{i=1}^N$ (defined by initial constraints + prev cuts)
- Objective parameter c (objective $= c^\top x$)
- Minimizing solution (to LP) $x^*_{LP}(t)$
- $\mathcal{D}^{(t)} = \{e_i^\top x \leq d_i\}_{i=1}^{I_t}$ is the set of candidate cuts given by Gomory's method

Setting up MDP: Actions

Discrete action space: the possible cutting planes given by Gomory's method, $\mathcal{D}^{(t)}$.

- Number of actions varies each t, but is $\leq n$
- Each action parameterized by real vectors $e_i, \ d_i$ (for plane $e_i^\top x \leq d_i$)

Setting up MDP: Reward

In each iteration, we promote as much immediate progress on LP objective as possible:

Reward in iteration t is: $r_t = c^\top x^*_{LP}(t+1) - c^\top x^*_{LP}(t)$ (which is ≥ 0)

Total expected return for a policy is $J(\pi) = \mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r_t]$

(Where $\gamma \in (0, 1]$, and T is **finite** horizon (we stop after fixed number of iterations, or after finding x^*_{IP})).

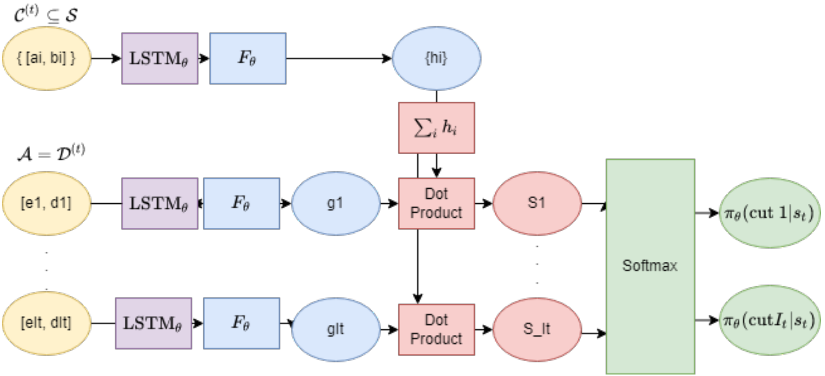
Setting up MDP: Transitions

Starting in state $s_t = \{\mathcal{C}^{(t)},\ c,\ x^*_{LP}(t),\ \mathcal{D}^{(t)}\}$, making action of selecting cut $e_i^\top x \leq d_i$

- Shrink feasible region as: $\mathcal{C}^{(t+1)} = \mathcal{C}^{(t)} \cup \{e_i^\top x \leq d_i\}$
- Solve the new LP (with simplex method) to obtain $x^*_{LP}(t+1)$
- Gomory's method on solution info gives new candidate cuts $\mathcal{D}^{(t+1)}$

Get to next state $s_{t+1} = \{\mathcal{C}^{(t+1)},\ c,\ x^*_{LP}(t+1),\ \mathcal{D}^{(t+1)}\}$ (**deterministic transition**)

Policy Network



- (Yellow): Inputs
- (Purple): LSTM encodes in fixed size (using last hidden state of LSTM)
- (Blue): Encode state and action info with learned function F_θ
- (Red): Attention computation to get scores "S"
- (Green): Softmax to get the **stochastic** policy $\pi_\theta(a_t|s_t)$: distribution over next cut

Learning Algorithm

We use **evolutionary strategies**: essentially a policy gradient method, updates policy parameters as $\theta \leftarrow \theta + \alpha \hat{g}_\theta$

Where $\hat{g}_\theta \approx \nabla_\theta J(\pi_\theta)$, but which estimates gradient without backprop:

- Sample N random "directions" $\epsilon_i \sim \mathcal{N}(0, I)$
- Step in the directions: $\theta' = \theta + \sigma \epsilon_i$
- Gradient estimate is

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

Train using this over batch of IP instances (average gradient over batch).

Related Methods

Other methods consist of human designed heuristics for cut selection (in Gomory's method) eg:

- Random
- Max Violation (MV)
- Max Normalized Violation (MNV)
- Lexicographical Rule (LE)

RL agent needs to learn selection process, so is expensive to train, but can perform better.

No directly comparable ML methods for cut selection

Table of Contents

IP Problems

Train and test on multiple different classes of IP problems:

- Packing
- Production Planning
- Binary Packing
- Max-Cut

Each class gives problem with different structure and form.

Train and test on multiple sizes $n \times m$ (number of variables and constraints), ranging from $n \times m = 200$ to 5000.

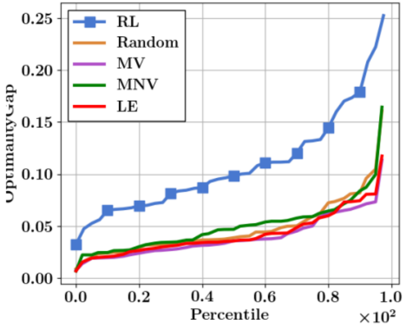
Experiment 1: Cut Efficiency

For small problems, monitor how many iterations (cuts) needed until reaching IP solution (table from paper):

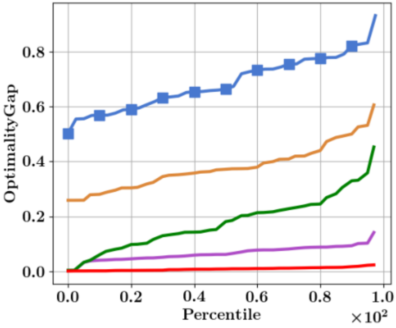
Table 1: Number of cuts it takes to reach optimality. We show mean $\pm$ std across all test instances.

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	$\mathbf{14 \pm 11}$	$\mathbf{10 \pm 12}$	$\mathbf{22 \pm 27}$	$\mathbf{13 \pm 4}$

For small problems, monitor IGC attained in percentage of instances for RL heuristic vs human-designed baselines (plots from paper):



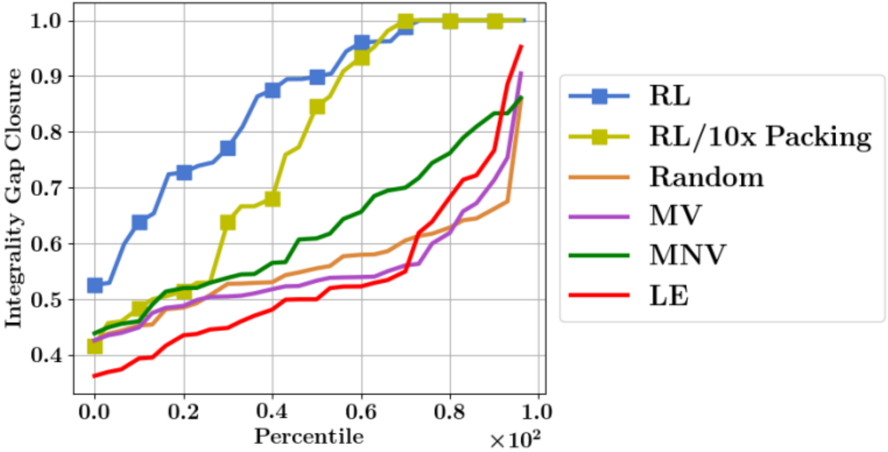
(a) Packing



(b) Planning

Experiment 3: Generalization

Evaluate difference in performance (IGC) for an agent trained on smaller problem instances AND different problem classes (plot from paper):



Experiment 4 and 5

- **Experiment 4 (Impact on efficiency of B&C)**: When using cutting plane method as subroutine in a larger B&C - branch and cut algorithm, the RL agent helps the downstream application more than the baselines.
- **Experiment 5 (Interpretability of Cuts)**: When applied to a simple, well-studied IP problem (knapsack problem), with known good cuts, the RL agent produces cuts resembling them.

Table of Contents

Conclusion

- This paper presented a novel application of RL to selecting cuts in IP problems.
- The RL agent can learn to select cuts better than human designed heuristics, with a capacity to generalize.
- Further work can improve efficiency and possibly integrate ML into commercial IP solvers.