Reinforcement Learning for Cut Generation in Branch-and-Cut

Thi Quynh Trang Vo[1], Viet Hung Nguyen[1], Mourad Baiou[1], Paul Weng[2]

[1] LIMOS Université Clermont Auvergne, France
{thi_quynh_trang.vo,viet_hung.nguyen,mourad.baiou}@uca.fr
[2] UM-SJTU Joint Institute, Shanghai Jiao Tong University
paul.weng@sjtu.edu.cn

Keywords : *Reinforcement Learning, Combinatorial Optimization, Branch-and-Cut, Cut Generation Strategy*

1 Introduction

Branch-and-Cut (B&C) is a popular method for exactly solving integer programming (IP) problems. B&C is the combination of two methods: branch-and-bound and cutting-plane. While branch-and-bound divides the problem into subproblems by a divide-and-conquer strategy, the cutting-plane method tightens these subproblems by adding valid inequalities. B&C contains a sequence of decision problems such as variable selection, node selection, and cut generation. Consequently, its performance heavily depends on decision-making strategies.

A core B&C component is the cutting plane method, which strengthens the linear programming (LP) relaxations of the IP problem by introducing additional valid inequalities, termed "cuts". Adding cuts can substantially remove infeasible regions and boost efficiency. In general, cuts are categorized into general-purpose cuts obtained by the variable's integrality conditions and combinatorial cuts arising from the underlying combinatorial structure of the problem.

However, generating cuts in B&C is a delicate process due to the challenge of balancing the computational cost of the separation procedure against the benefits of cuts produced. Generating cuts in a naive way can reduce the branch-and-bound tree's size but potentially increase the overall computing time due to the time spent executing the separation routine and solving the LP relaxations in the enumeration tree. Thus, learning a deft policy for cut generation is crucial.

In our previous work [1], we proposed a Machine Learning framework to enhance the generation of subtour elimination constraints for the Traveling Salesman Problem (TSP). In this paper, we extend this framework to the Max-Cut problem.

2 Cut generation problem

Consider a combinatorial optimization problem $\mathcal{P}$ defined on a graph $G = (V, E)$ and a class of combinatorial cuts $\mathcal{C}$ associated with $\mathcal{P}$. A cut generation strategy $\pi_{(\mathcal{P},\mathcal{C})}$ for cut type $\mathcal{C}$ in B&C for $\mathcal{P}$ decides whether to generate $\mathcal{C}$-type cuts or to branch at each node of the enumeration tree. The cut generation problem is to learn a cut generation strategy $\pi_{(\mathcal{P},\mathcal{C})}$ that obtains the best average performance τ on a given set of problem instances $\mathcal{I}_{\mathcal{P}}$, i.e.,

$$\pi_{(\mathcal{P},\mathcal{C})} \in \arg\min_{\pi \in \Pi} \mathbb{E}_{p \in \mathcal{I}_{\mathcal{P}}}[\tau(p, \pi)]$$

where $\tau(p, \pi)$ is the B&C's running time for solving instance p with cut generation policy π.

3 Methodology

To learn cut generation policies, we also formulate the cut generation problem as a Markov Decision Problem (MDP) and then use reinforcement learning to train the agent as in [1]. However, instead of defining the reward function based on the IP relative gap [1], which requires extensive effort to tune hyperparameters, we introduce a new reward function based on the running time. This reward function directly drives our ultimate goal, which is to accelerate the B&C's performance. However, this time-based reward function causes difficulties for the agent to learn due to the long-term benefits of generating cuts, which can only be observed at trajectory endpoints.

To tackle this issue, we simplify the MDP of cut generation as follows. Starting from the root node, the agent iteratively chooses between cut generation and branching, aiming to reach a leaf node. The reward for reaching a leaf node is the negative of the relative IP gap at that node. Each action incurs a cost equal to its runtime. The final reward is the leaf node reward subtracted by the total running time of all performed actions. The agent's objective is to discover a cut-generation policy that maximizes this final reward.

With this simplified version, we represent a state by only two components: an optimal solution to the LP relaxation at the considered node and features of the current node. Then, we model the cut evaluator as the Q-value function of the MDP and parameterize it as a neural network with two parts: one embedding states into feature vectors and one approximating action Q-values. To train the cut evaluator, we use the DQN algorithm, i.e., the parameters of the cut evaluator are updated to minimize an L2 loss defined with a target network using data sampled from a replay buffer filled with transitions generated during online interactions with the environment.

4 Numerical results

To evaluate the effectiveness of our proposed method, we use this framework to learn generation strategies of cycle inequalities for the Max-Cut problem. We train the model on small random instances and then test it on larger instances. Table 1 shows that our framework can significantly accelerate the B&C's performance on all instance groups.

	Strategy	Solved	CPU Time	Nodes	Cuts
SMALL	`Without RL`	100/100	275.4	**297.4**	14403.4
	`With RL`	100/100	**60.7**	449.1	**4443.8**
MEDIUM	`Without RL`	47/100	2777.8	**1666.9**	38111.5
	`With RL`	**87/100**	**1310.2**	3557.8	**30723.8**
LARGE	`Without RL`	0/100	3600.1	**3806.2**	**56115.7**
	`With RL`	**2/100**	**3559.2**	4202.8	59108.2

TAB. 1: The numerical results of the cycle inequality generation strategies

References

[1] T. Q. T. Vo, M. Baiou, V. H. Nguyen, and P. Weng. Improving subtour elimination constraint generation in branch-and-cut algorithms for the tsp with machine learning. In M. Sellmann and K. Tierney, editors, *Learning and Intelligent Optimization*, pages 537–551, Cham, 2023. Springer International Publishing.