Use graph neural networks for cutting-plane selection

February 2022

1 Introduction

This research proposal concerns a new machine learning-based cutting-plane algorithm. The cutting-plane algorithm is now an important accelerate component in commercial and noncommercial solvers for Mixed Integer Linear Program (MILP) in the form

$$\min\{c^T x : Ax \geq b,\ x \geq 0,\ x_j \in \mathbb{Z}\ \forall j \in I\} \tag{1}$$

The modern MILP solvers implement a linear programming-based branch-and-bound algorithm. This divide-and-conquer approach allows the solvers to reduce the original problem into a sequence of smaller subproblems, which can be handled by solving linear programming relaxation. However, this approach also leads to much larger time cost than heuristics. To address this disadvantage, algorithms have been developed to make solvers faster, such as size-reducing presolve algorithms and primal heuristics to find feasible integer solutions before the branching procedure begins. Among them are the cutting-plane algorithms. More details of the development of MILP solver are described in the survey of Bixby[12]. The cutting-plane algorithms function in the branch-and-bound framework. At each fractional relaxation node, the built-in cutting-plane algorithms will add some *properly chosen* cutting-plane inequalities to the LP formulation, delivering a tighter dual bound at the child nodes and speeding up the overall branch-and-bound procedure.

The cutting-planes have been playing an increasingly important role to accelerate the modern MILP solvers. The research of the cutting-planes began in the 1960s with Gomory inventing the Gomory mixed-integer(GMI) cuts[20]. These cuts were regarded by researchers as beautiful but impractical by the time. The research of the computational application of cuts made a breakthrough until the 1990s. Balas, Ceria, and Cornuéjols published a series of papers showing that using a well-designed algorithm for cut generation, the cuts result in a significant reduction in computing time[8, 7]. GMI cuts were first implemented in commercial MILP solvers, such as Cplex, in 1999. Other cutting planes were later implemented, resulting in solvers that were orders of magnitude quicker.

Many families of cutting planes have been discovered in the past three decades. The cutting planes can be divided into general cutting planes, which are independent of problem structure, and cutting planes that exploit the problem structure. The general cutting planes include Gomory Mixed Integer cuts[20], split cuts[6, 7, 16], Chvátal-Gomory cuts[13, 18, 21]. The cutting planes based on problem structures include cover inequalities[4] and knapsack inequalities[5]. Now, more cutting plane families are known. However, in this research proposal, we do not attempt to survey the wide literature on cutting plane families. We concentrate on the details of the cutting plane application.

Vast literature have discussed issues of cut separation, which is one part of cutting-pane application. The cut separation problem means to find whether there exists a cut violated by a given relaxation node. The cut separation problem is an $\mathcal{NP}$-hard problem except for the lift-and-project cuts. This property makes directly solving cut separation problem within the branch-and-bound framework impractical. Efficient cut-generating heuristics for Chvátal-Gomory cuts[19, 26] and split cuts[14], flow cover cuts[1] and knapsack inequalities[24] and other cuts families have been raised to address this issue.

Although many cut-generating heuristics have been developed and discussed, the problem of adding cuts during the branch-and-bound method is still far from solved. The problem of cut selection must be considered, after a list of candidate cuts is generated by heuristics. This problem is critical for the cutting-plane algorithm's overall performance, Because determining which cuts to add directly affects the LP solution and computing time. A naive approach to add cuts from the candidate list may increase computing time while the dual bound remains unchanged.

In the last few years, researchers have been developing machine learning (ML)- based algorithms to solve combinatorial optimation. An excellent overview on ML techniques for combinatorial optimation problems is given by Yoshua Bengio et al.[11]. Some ML algorithms were studied to accelerate the MILP solving procedure. These algorithms focus on certain steps of the MILP solving procedure. Alvarez et al.[2] proposed an approach to variable branching in branch and bound, based on using a special decision tree to learn strong branching strategies. Khalil et al.[25] provided a similiar resolvent, in which a linear model is learned on the fly for each instance via strong branching at the tree's top, and then replaced by its ML approximation. Hutter et al.[23] proposed an automated algorithm configuration procedure to find parameter settings of high performance for moerdern MILP solvers.

Little work has been done so far for the application of ML techniques on cut generation and selection. Tang et al.[28] proposed a reinforcement learning framework for cut selection. This study introduced a Markov decision process (MDP) formulation for iteratively selecting cuts and trained the reinforcement learning agent by evolutionary strategies. Zeren Huang et al.[22] presented a data-driven approach to train the scoring function, by which the cuts in the *cut-pool* are ranked. The trained scoring function performs better than the cut measurement of the modern MILP solvers.

2 Basic Model and Cut Selection

Consider the following generic mixed integer problem:

$$\begin{aligned} \max \quad & c^T x + h^T y \\ & Ax + Gy \leq b \\ & x \in \mathbb{Z}_+^n \\ & y \in \mathbb{R}_+^p \end{aligned} \tag{2}$$

Let $P := \{(x, y) \in \mathbb{R}_+^n \times \mathbb{R}_+^p : Ax + Gy \leq b\}$ be a polyhedra. Let $S := P \bigcap (\mathbb{Z}_+^n \times \mathbb{R}_+^p)$. The set S is a set of all feasible solutions to this mixed integer linear program. It can be shown that the optimum point can only be found in the extreme points of the conv(S). Thus the cutting-plane approach is to find inequalities that form approximations of conv(S). An inequality $\alpha x + \beta y \leq \gamma$ is said to be valid for conv(S) if it is satisfied by every point in conv(S). A cut with respect to a point $(x,\, y) \notin$ conv(S) is a valid inequality for conv(S) that is violated by $(x,\, y)$.

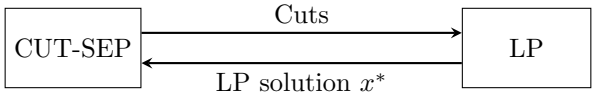

Figure 1: Standard Cutting-plane algorithm

The cutting-plane algorithms work within the branch-and-bound framework in an iterative way (See Figure 1). At each turn, the LP solver produce a solution point $(x,\, y)$ of the relaxation problem (2) added previous cuts . The cutting-plane algorithms add one or more cuts to the LP formulation, by solving the following cut separation problem with $(x,\, y)$:

$$\begin{aligned} &\text{CUT-SEP:} \quad \text{Given } (x^*, y^*) \in P, \text{ solve} \\ &\qquad z^*_{SEP} = \min\{\alpha x^* + \beta y^* - \gamma : (\alpha, \beta, \gamma) \in \mathcal{F}(A, G, b)\} \end{aligned} \tag{3}$$

Here $\mathcal{F}(A, G, b)$ denotes a family of valid inequalities. If $z^*_{SEP} < 0$ in problem (3), no cut can be found, otherwise the solution $(\alpha^*, \beta^*, \gamma^*)$ is a cut that cut off point $(x,\, y)$.

The cut separation problem is an $\mathcal{NP}$-hard problem except for the lift-and-project cuts. It is impractical to seek a exact solution for a cut separation problem at every step of the Branch-and-Cut process. Heuristics to generate cuts are developed and well-discussed for this topic [19, 26, 24].

The MILP solvers handle the cut selection problem by putting the cuts maintained by heuristics into a list called *cut-pool*, ranking them, and adding a few of them to the LP formulation. Many questions, however, remain unanswered. referring to views of Dey et al.[17], we discuss some representative examples.

Cut quality measure The quality measure of cuts, by which the modern solvers rate the cuts in the *cut-pool*, is the most direct problem. Balas et al.[9]

proposed the the *depth-of-cut* measure, ie., $\frac{\alpha x^* - \beta}{||\alpha||_2}$, for a cut $\alpha x \leq \beta$ violated by a fractional point x^*. The *depth-of-cut* measure is commonly used[3]. However, A recent paper[10] proves that the depth-of-cut measure may select feeble cuts.

Cut quantity Determining how many cuts to include on the LP is also an unsolved problem. Experimental results show that adding separately leads to a very slow convergence rate. But adding too many cuts at the same time may lead to a larger LP computing time. To the best of our knowledge, no study has discussed what is the best way to decide the number of cuts to add.

Cut Combination The best situation is that the polyhedron is improved by a combination of cuts in diverse directions. This means that the relationship between cuts shall be considered. Some papers[9, 3] use the angle of two cutting planes as a measure to avoid adding similar cuts. However, this approach do not fully solve the issue of cut combination.

3 Methodology

At each Branch-and-Cut iteration, a set $\mathcal{C}$ of cuts is obtained by cut-generation heuristics. The goal is to select a subset $\mathcal{C}_s \subseteq \mathcal{C}$ of cuts to be added to the LP formulation. Unlike the ranking approaches, we consider this problem as a binary classification problem (supervised learning problem).

3.1 Data Set

A data set $\mathcal{D} = \{(\mathcal{X}, \mathcal{Y}_1), (\mathcal{X}_2, \mathcal{Y}_2), ..., (\mathcal{X}_n, \mathcal{Y}_n)\}$ is required for our supervised learning approach, where n denotes the size of the data set, $\mathcal{X}_i$ is the vector representing the features of cut i, and $\mathcal{Y}_i \in \{0, 1\}$ the cut label.

We obtain the data set for training by solving MILP for cut selection. we fist regard cut selection problem as a mixed integer problem. We want the selected subset $\mathcal{C}_s$ lead to a maximum reduction of the dual bound, while we also want a relatively small number of selected cuts.

A maximum reduction of the dual bound means we need to choose the optimal bound reduction cuts. It can be formulated as the following bi-level program.

$$\min_{l} \max_{x,y} (c^T x + h^T y + \epsilon \sum_{i=1}^{|C|} l_i) \tag{4}$$

$$\text{s.t. } Ax + Gy \leq b \tag{5}$$

$$\text{(previous cuts)} \tag{6}$$

$$\alpha_i x + \beta_i y \leq \gamma_i l_i + (1 - l_i)M \quad i \in C \tag{7}$$

$$x \in \mathbb{R}^n_+ \tag{8}$$

$$y \in \mathbb{R}^p_+ \tag{9}$$

$$l \in \{0, 1\}^{|C|} \tag{10}$$

Let $C_t := \{1, 2, 3, ...\}$ be the set of generated cuts of iteration t. For each cut $i \in C_t$, we define a decision variable l_i that equals to one if cut i is selected and zero otherwise. The objective function (25) is the sum of original objective function of problem (2) and a ϵ-weighted sum of l_i, which is set in the same order of magnitude as $cx + hy$. The coefficient ϵ is used to control the number of selected cuts. Constraints (26)-(27) are original constraints and previous cuts. The big-M constraints (28), where $M = \max\{\alpha_i x + \beta_i y | i \in C, (x, y) \in P\}$ would only add cut i for which $l_i = 1$. Constraints (36)-(38) define the domain of variables.

For simplification, we denote the previous cuts as

$$Dx + Hy \leq e$$

the cut-selecting constraints as

$$\alpha x + \beta y \leq \gamma_M l + M \cdot \mathbf{1}.$$

Where $\gamma_M = diag(\gamma_1 - M, \gamma_2 - M, ..., \gamma_{|C|} - M)$ and $\mathbf{1}$ is $|C|$-dimensional vector $(1, 1, 1, ..., 1)$. Denote the number of rows of A by a, D by d.

We involve dual variables of the program 2. The bi-level program can be convert to a quadratically constrained linear program:

$$\min(c^T x + h^T y + \epsilon \sum_{i=1}^{|C|} l_i) \tag{11}$$

$$\text{s.t. } Ax + Gy \leq b \tag{12}$$

$$Dx + Hy \leq e \tag{13}$$

$$\alpha x + \beta y \leq \gamma_M l + M \cdot \mathbf{1} \tag{14}$$

$$\mu_1^T A + \mu_2^T D + \lambda^T \alpha \geq c^T \tag{15}$$

$$\mu_1^T G + \mu_2^T H + \lambda^T \beta \geq h^T \tag{16}$$

$$c^T x + h^T y = \mu_1^T b + \mu_2^T e + \lambda \gamma_M y + \lambda^T M \cdot \mathbf{1} \tag{17}$$

$$x \in \mathbb{R}_+^n \tag{18}$$

$$y \in \mathbb{R}_+^p \tag{19}$$

$$l \in \{0, 1\}^{|C|} \tag{20}$$

$$\mu_1 \in \mathbb{R}_+^a \tag{21}$$

$$\mu_2 \in \mathbb{R}_+^d \tag{22}$$

$$\lambda \in \mathbb{R}_+^{|C|} \tag{23}$$

$$\tag{24}$$

We linearize the system above by substituting variable w_i for $\lambda_i y_i$. $w_i = \lambda_i$ if $y_i = 1$ and 0 otherwise. Denote the upper bound of λ by scalar U. We have the following MILP:

$$\min(c^T x + h^T y + \epsilon \sum_{i=1}^{|C|} l_i) \tag{25}$$

$$\text{s.t. } Ax + Gy \leq b \tag{26}$$

$$Dx + Hy \leq e \tag{27}$$

$$\alpha x + \beta y \leq \gamma_M l + M \cdot \mathbf{1} \tag{28}$$

$$\mu_1^T A + \mu_2^T D + \lambda^T \alpha \geq c^T \tag{29}$$

$$\mu_1^T G + \mu_2^T H + \lambda^T \beta \geq h^T \tag{30}$$

$$c^T x + h^T y = \mu_1^T b + \mu_2^T e + \gamma_M w + \lambda^T M \cdot \mathbf{1} \tag{31}$$

$$w \leq \lambda \tag{32}$$

$$w \geq \lambda - U(1 - y) \tag{33}$$

$$w \geq 0 \tag{34}$$

$$w \leq Uy \tag{35}$$

$$x \in \mathbb{R}_+^n \tag{36}$$

$$y \in \mathbb{R}_+^p \tag{37}$$

$$l \in \{0, 1\}^{|C|} \tag{38}$$

$$\mu_1 \in \mathbb{R}_+^a \tag{39}$$

$$\mu_2 \in \mathbb{R}_+^d \tag{40}$$

$$\lambda \in \mathbb{R}_+^{|C|} \tag{41}$$

$$w \in \mathbb{R}_+^{|C|} \tag{42}$$

$$\tag{43}$$

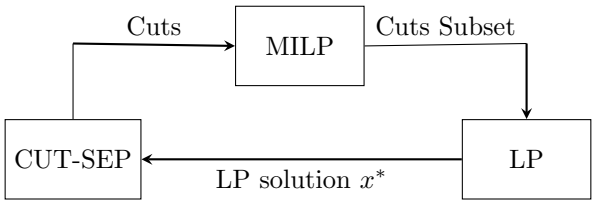

Figure 2: Cutting-plane algorithm With MILP

At each step, we solve this MILP above(See Figure 2), then a data set for training are produced. The vector of coefficients of cut i is the vector $\mathcal{X}_i$. The integer solution of l in the MILP becomes label $\mathcal{Y}_i$. In the following steps, we train the neural networks with this data set. In other words, we view the solution of this MILP as "good" choices for each iteration situation, and we want our algorithm learn form them.

Here we give a simple illustrative computational example to show the strength of this MILP. Codato and Fischetti gave a framework of using combinatorial

Benders cuts for mixed-integer problems in their paper [15]. They defined a master Integer Linear Problem (ILP) which initially has no constraints. The master problem solutions are sent to a slave Linear Program (LP), which gives feasibility information for generating benders cuts add to the master problem. When the master problem solution is feasible for the slave problem, the original mixed-integer problem is solved. We take a simple exam on an example named `'busvan-445'` used by Codato and Fischetti. In the 1st round, 30 cuts are generated, we use our MILP to give the best subset of cuts that give maximum bound improvement but a minimum number of cuts.

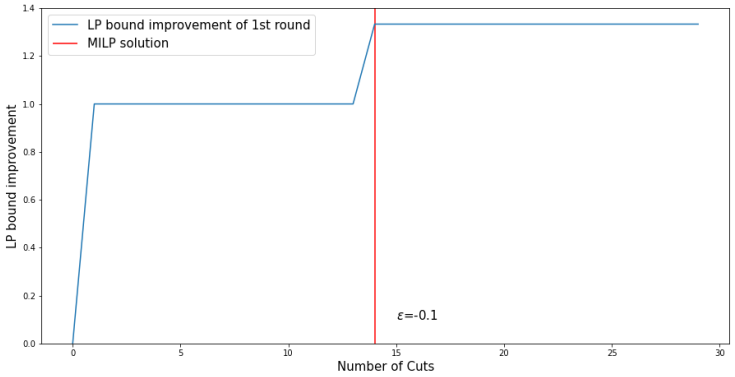

Figure 3: MILP for cut selection

3.2 Graph neural networks

Graph neural networks (GNNs) can capture the neighborhood properties of nodes in graphs. Thus, the GNNs can be used for tasks with graph-structured data, like node classification, graph classification, link prediction, etc. Because of the the advantages of GNNs in capturing dependencies between nodes for graph-structured data, we assume that GNNs have great potential for the cut selection problem. A similar study has been conducted by Morabit et al.[27]. In this paper are proposed to solve the column selection problem in the column generation process. In this study, a bipartite graph are structured to represent columns and their relationships. Because of similarities of data structure between sets of cuts and sets of columns, we consider that our approach is highly feasible.

3.3 Research Objects

For computational experiments, we consider MIP problem classes consist of Set Cover, Knapsack, Planning and General MIP. The instances will be randomly generated as in paper of Zeren Huang et al [22].

Our goal is to design a GNNs-based algorithm for cut selection. The study has the following sub-objectives, and here we give a half-year schedule:

1 In the first two months: We will mainly work on the MILP model. Our research topic includes the choice of penalty coefficient ϵ, the experiment effect of sparsity and cuts from multiple sources, which have been discussed in the paper of Dey et al[17], the strength of the bound-optimal model compared to classic *depth-of-cut* measure.

2 In the next two months: we will find a graph network architecture and train the graph network to learn from solutions of our mature MILP model.

3 In the last two months: we will conduct computational experiments in multiple problems in the field of mixed-integer programming to compare our algorithm with modern MILP solvers' algorithms and the existing cut-selecting algorithms

References

[1] Karen Aardal. Capacitated facility location: separation algorithms and computational experience. *Mathematical programming*, 81(2):149–175, 1998.

[2] Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. A Machine Learning-Based Approximation of Strong Branching. *INFORMS JOURNAL ON COMPUTING*, 29(1):185–195, WIN 2017.

[3] Giuseppe Andreello, Alberto Caprara, and Matteo Fischetti. Embedding {0, 1/2 }-cuts in a branch-and-cut framework: A computational study. *INFORMS Journal on Computing*, 19(2):229–238, 2007.

[4] E BALAS and E ZEMEL. FACETS OF KNAPSACK POLYTOPE FROM MINIMAL COVERS. *SIAM JOURNAL ON APPLIED MATHEMATICS*, 34(1):119–148, 1978.

[5] Egon Balas. Facets of the knapsack polytope. *Mathematical programming*, 8(1):146–164, 1975.

[6] Egon Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3–44, 1998.

[7] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1):295–324, 1993.

[8] Egon Balas, Sebastian Ceria, Gérard Cornuéjols, and N Natraj. Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9, 1996.

[9] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42:1229–1246, 09 1996.

[10] Amitabh Basu, Michele Conforti, Marco Di Summa, and Giacomo Zambelli. Optimal cutting planes from the group relaxations. *Mathematics of Operations Research*, 44(4):1208–1220, 2019.

[11] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[12] Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, 2012:107–121, 2012.

[13] Alberto Caprara and Matteo Fischetti. {0, 1/2}-chvátal-gomory cuts. *Mathematical Programming*, 74(3):221–235, 1996.

[14] Alberto Caprara and Adam N Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming*, 94(2):279–294, 2003.

[15] Gianni Codato and Matteo Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54:756–766, 08 2006.

[16] William Cook, Ravindran Kannan, and Alexander Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47(1):155–174, 1990.

[17] Santanu S. Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170:237–266, 2018.

[18] Santanu S Dey, Jean-Philippe P Richard, Yanjun Li, and Lisa A Miller. On the extreme inequalities of infinite group problems. *Mathematical Programming*, 121(1):145–170, 2010.

[19] Matteo Fischetti, Andrea Lodi, and Andrea Tramontani. On the separation of disjunctive cuts. *Mathematical Programming*, 128:205–230, 2011.

[20] Ralph Gomory. An algorithm for the mixed integer problem. Technical report, RAND CORP SANTA MONICA CA, 1960.

[21] Ralph E Gomory. Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. In *50 Years of Integer Programming 1958-2008*, pages 77–103. Springer, 2010.

[22] Zeren Huang, Kerong Wang, Furui Liu, Hui-Ling Zhen, Weinan Zhang, Mingxuan Yuan, Jianye Hao, Yong Yu, and Jun Wang. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353, 2022.

[23] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Automated configuration of mixed integer programming solvers. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 186–202. Springer, 2010.

[24] Konstantinos Kaparis and Adam N Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical programming*, 124(1):69–91, 2010.

[25] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[26] Arie M. C. A. Koster, Adrian Zymolka, and Manuel Kutschka. Algorithms to separate 0,1/2-chvatal-gomory cuts. *Algorithmica*, 55:375–391, 2009.

[27] Mouad Morabit, Guy Desaulniers, and Andrea Lodi. Machine-learning–based column selection for column generation. *Transportation Science*, 55(4):815–831, 2021.

[28] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9367–9376. PMLR, 13–18 Jul 2020.