Semi-supervised Learning Approach to Efficient Cut Selection in the Branch-and-Cut Framework*

Jia He Sun[1] and Salimur Choudhury[2]

[1] Carleton University, Ottawa, ON, Canada
jiahesun@cmail.carleton.ca
[2] Queen's University, Kingston, ON, Canada
s.choudhury@queensu.ca

Abstract. Mixed integer programming (MIP) is an extremely versatile subclass of mathematical optimization problems. Applications of MIP are ubiquitous in our world today, ranging from scheduling to network design to production planning. The standard approach in state-of-the-art commercial solvers is called branch-and-cut. The selection of these cuts is an integral part of the branch-and-cut process as high-quality cuts can greatly increase solving efficiency. Currently, cut selection is decided by heuristics that both require expert knowledge and lack generalizability. In this paper, we propose an efficient and highly generalizable cut selection scheme based on semi-supervised learning. First, we design a cut evaluation metric that labels cuts based on whether they are efficient or not. Then, we train a deep learning classification model with unsupervised pre-training as a ranking function for cuts. In our evaluation, the proposed model outperforms standard heuristics and is comparable to existing machine learning approaches. Furthermore, the model is shown to be generalizable over both problem size and problem class.

Keywords: Machine learning · Semi-supervised learning · Mixed integer programming · Cutting planes.

1 Introduction

MIP problems are linear programming (LP) problems with integrality constraints. This particular subclass of optimization problems can be applied to a plethora of industrial applications including but not limited to: scheduling [7], network design [8], and production planning [9]. Modern commercial MIP solvers take the branch-and-cut approach which is a combination of the branch-and-bound technique and the cutting planes technique [18]. The selection of solution variables for the branching and the selection of cuts are key decisions with a huge impact on the overall efficiency of the branch-and-cut algorithm [18]. Recently, there has been a surge in interest from the ML community in augmenting the branching process of the branch-and-cut framework [4] [5] [6] [3] [10].

* Supported by NSERC.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40

The cut selection process, the focus of this paper, has seen less focus from researchers aiming to integrate machine learning into the branch-and-cut framework. Tang et al. proposed a deep reinforcement learning (RL) formulation for intelligent adaptive cut selection for the cutting planes method [2]. Paulus et al. proposed an imitation-based learning model called "NeuralCut" based on a lookahead expert [13]. While these two works are in the same domain as our work, they are different in terms of target evaluation.

Huang et al. designed a multiple instance supervised machine learning model for cut selection in the branch-and-cut framework called "Cut Ranking" [1]. Huang et al.'s work is the most similar to the work proposed in this paper. However, not only do we propose a different labelling system for generating labelled cut data, but we also take a semi-supervised approach to the machine learning model as opposed to "Cut Ranking"'s completely supervised approach.

In this paper, we propose a generalizable and efficient cut selection scheme for the branch-and-cut framework. This selection scheme includes a cut classification system that differentiates efficient cuts from inefficient cuts and a semi-supervised machine learning model that learns to do the same.

2 Methods

For every MIP problem, and for each node of the branch-and-bound search tree, existing MIP solvers can generate a set of candidate cuts. The goal of our model is to select the most efficient cuts from this candidate set. In our work, the approach taken is multiple instance learning (MIL).

MIL is where the training data is generated based on bags of instances. This approach is chosen because individual cuts will have little measurable effect on the overall efficiency of the branch-and-cut framework, thus, cuts are grouped into bags and are evaluated at the bag level. Then, labels are assigned at the bag level. "Cut Ranking" takes the same approach for data generation [1].

To construct the bags of instances, consider a MIP problem P of the form:

$$max\{c^T x : Ax \leq b, x_j \in Z, \forall j \in N_I\} \tag{1}$$

where $c, x \in R^n, A \in R^{m\times n}$, and $N_I \subseteq N = \{1, ..., n\}$. Let x_{LP} be an optimal solution to P's corresponding LP relaxation and let C be the candidate cut set generated by a solver. For each cut $c_i \in C$, it is of the form:

$$\alpha_i^T x \leq \beta_i \tag{2}$$

Let $f_{c_i} \in R^l$ denote the feature vector of c_i. Let $B = \{B_1, ..., B_k\} \subseteq C$ be all bags of cuts sampled from C. Then, the feature vector of a bag B_u, denoted by f_{B_u}, is the average of the feature vectors f_{c_i} for all $c_i \in B_u$. That is, the feature vector of a bag of cuts is the average of the feature vectors of the cuts in the bag. Furthermore, $|B_u| \geq 0.1 \cdot |C|, \forall j \in \{1, ..., k\}$. In other words, the size of each sampled bag of cuts must be at least 10% of the size of the candidate

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40

cut set. This is to ensure that we do not have samples with not enough cuts to make a measurable difference in run time.

For each cut c_i, the features extracted are as follows:

1. cut coefficients features (4): maximum, minimum, mean, and standard deviation of cut coefficients α_i
2. objective function coefficients features (4): maximum, minimum, mean, and standard deviation of objective function coefficients that correspond to the non-zero cut coefficients
3. support
4. integral support
5. relative violation
6. distance
7. objective function parallelism
8. expected improvement

The first four features are basic structural data of the cut. The rest of the features are popular metrics for measuring the quality of cuts. The exact definitions of these features can be found in Wesselmann and Suhl's work [11].

2.1 Cut Evaluation and Data Labelling

For each MIP problem P, after we have sampled k bags from the generated candidate cut set C, every sampled bag B_u is evaluated by adding all cuts in B_u to P and running the solver. To evaluate the performance of each bag, the metric used in our scheme is normalized run time.

Let r_j be the run time of problem P with appended bag B_j for all $j \in \{1, ..., k\}$. Without loss of generality, assume B_v to be the bag with shortest run time and B_w be the bag with the longest run time. Then, the evaluation value assigned to each sampled bag B_u, normalized run time, is defined by:

$$r_j^* = 1 - \frac{r_j - r_m}{r_n - r_m} \tag{3}$$

In this format, for each MIP problem, the best performing bag will always be evaluated as 1 and the worst performing bag will always be evaluated as 0. The rest of the bags will have an evaluation of some value in $[0, 1]$.

After each bag of cuts has been evaluated, it will be given a discrete label. In our scheme, we will assign 1 to bags with normalized run time over λ_1 and assign 0 to bags with normalized run time under λ_2. λ_1 and λ_2 are both hyperparameters between 0 and 1 and $\lambda_1 > \lambda_2$. All other bags will not be labelled and consequently will not be used in the supervised training portion of the model.

Furthermore, notice that we choose to allow some data points to remain unlabelled. Intuitively, we are labelling the samples we know are good as 1 and the samples we know are bad as 0. The samples in the middle that could be good or could be bad are not labelled.

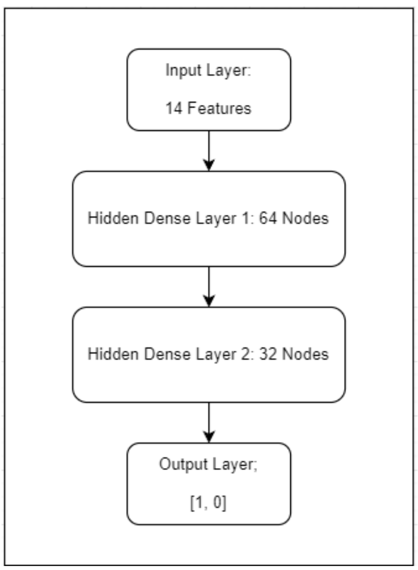
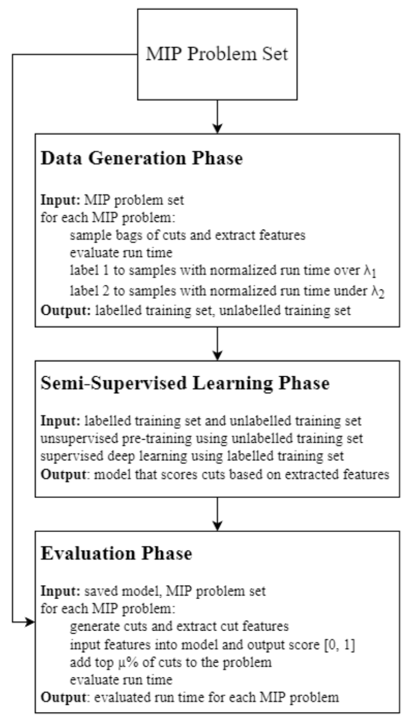
ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40

2.2 Unsupervised Pre-training

The proposed machine learning model is a semi-supervised deep learning model for tabular data which employs an unsupervised pre-training model. The reason that unsupervised pre-training is chosen is that we have an abundance of unlabelled data at our disposal. We implement the pre-training model used in TabNet, a deep tabular data learning model [17]. TabNet's pre-training model, similar to a denoising auto-encoder, is designed to predict missing feature values from corrupted feature input based on observed interdependencies.

2.3 Model Architecture

Table 1. Diagrams regarding the architecture and cut selection scheme

Model Architecture	Proposed cut selection scheme
	

Following pre-training, the data will be pushed through a supervised classification model. The model consists of 4 fully connected layers with an input layer, an output layer, and 2 hidden dense layers of size 64 and 32 respectively. Since

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40

the proposed model is a binary classification model we chose to use binary cross entropy as our loss function:

$$-\frac{1}{N}\sum_{i=1}^{N} y\log(p) + (1-y)\log(1-p) \tag{4}$$

where N is output size, y is target value, and p is model output. For the same reasoning, we also choose to use sigmoid as our activation function:

$$\sigma(z) = \frac{1}{1+e^{-x}} \tag{5}$$

A regression model was not chosen because, in our internal experiments, the binary classification model consistently outperformed it. We hypothesize that this is due to the relatively small amount of labelled data and the existence of many outliers. A multi-class classification model was also tested with little success. For each cut, the output of the model will be a continuous value between $[0,1]$ and the top $\tau\%$ of cuts will be added to the model, τ is the cut selection threshold hyperparameter. A visual of our implemented architecture as well as a high-level diagram of our proposed cut selection scheme is given in Table 2.3.

2.4 Data Sets

To train our model, we procured a data set consisting of 80 real-world set partitioning problems from the Mixed Integer Programming Library (MIPLIB2017) [12]. Set partitioning is chosen as it is one of the most widely applied mathematical optimization problems [19]. The problems in our chosen problem set are all similar in terms of difficulty as they all take less than 30 minutes to solve. For each problem, 135 samples were extracted from the candidate cut set generated by our solver [15]. The total number of data points generated for training is 10,602.

For comparison, we implement the following evaluation baselines:

1. relative violation
2. objective function parallelism
3. distance
4. “Cut Ranking" [1]
5. proposed model but without unsupervised pre-training

Baselines 1-3 are common heuristics for cut selection [11]. Baseline 4 is the model proposed by Huang et al. that also focuses on run time [1]. Baseline 5 is to confirm the effects of using unlabelled data.

For evaluation, we perform experiments on the following real-world data sets:

1. 50 small set partitioning problems (different problems than the ones used in training) [12]
2. 50 large set partitioning problems [12]

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40

3. 50 mixed integer knapsack problems [14]
4. 50 lot sizing problems [16]
5. 50 general MIP problems [12]

Data sets 1-2 are used to evaluate the performance of the model on similar problems that it was trained on as well as how well it generalizes in terms of problem size. Data sets 3-5 are used to evaluate the model's generalizability on different problem classes.

2.5 Hyperparameters

After tuning, the hyperparameters for data generation are $\lambda_1 = 0.7$ and $\lambda_2 = 0.45$. That is, cut samples with normalized run time over 0.7 are labelled 1, and cut samples with normalized run times under 0.45 are labelled 0. With these hyperparameters, the labelled samples total 5,020, and the unlabelled samples total 5,582. The cut selection threshold hyperparameter τ is set to be 0.7, that is, the top 30% of cuts are added to the model.

For model specific hyperparameters, dropout is set to be 0.01, the learning rate is set to be 0.0001, batch size is set to be 32, unsupervised pre-training is set for 512 epochs, and supervised training is also set for 512 epochs.

2.6 Implementation

The MIP solver used is the Coin-or Cut-and-Branch Solver [15]. The training data used in this study are openly available in Mendeley Data at DOI: 10.17632/thtz8h894m.1. The model implemented is based on Arık and Pfister's proposed model and is available here [17].

3 Results

In our experimentation, the cut selection scheme is implemented only at the root node of the search tree. In other words, cuts are only being added to the original MIP problem. Since each node of the branching search tree can be considered its own MIP problem, we believe that the results of our experimentation extend to all nodes of the branching search tree.

Table 2 displays the evaluation results of our proposed model compared against the selected baselines. First and foremost, the proposed model significantly outperforms all evaluated baselines on the data set it was trained on, set partitioning (small), achieving an average normalized run time of almost 50% better than the next highest baseline.

As for the other data sets, our proposed model is at worst comparable to "Cut Ranking" and the other heuristics. For the set partitioning (large) and the lot sizing data sets, our model outperforms all of the baselines by a comfortable

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40

Table 2. Average normalized run time evaluation between proposed model, "Cut Ranking", and various heuristics (higher is better)

Problem Set	Proposed Model	Cut Ranking	Relative Violation	Distance	Parallelism	Without Pre-training
Set Partitioning (small)	**0.764**	0.515	0.476	0.372	0.351	0.676
Set Partitioning (large)	**0.749**	0.621	0.579	0.475	0.397	0.612
Mixed Integer Knapsack	0.991	**0.998**	0.974	0.762	0.764	0.975
Lot Sizing	**0.795**	0.698	0.667	0.197	0.740	0.596
General MIP	0.695	**0.729**	0.562	0.424	0.392	0.612

margin. For mixed integer knapsack problems, the proposed model performs barely worse than "Cut Ranking". For the general MIP data set, our model is less efficient than "Cut Ranking" by a slight margin but outperforms the other baselines.

Lastly, we can see that our proposed model consistently outperformed its no pre-training counterpart. This confirms that the unsupervised pre-training portion of our model is indeed beneficial.

4 Conclusion

The backbone of modern state-of-the-art MIP solvers is the branch-and-cut framework. The selection of cutting planes to be implemented at each node of the branching search tree is an important task and, to tackle this, we proposed a semi-supervised deep learning based cut selection scheme. An unsupervised pre-training model is trained to reconstruct features based on inter-feature dependencies using unlabelled data. Then, the labelled data are trained using a standard binary classification approach. From our experiments on real-world MIP problem sets, we found that our model is not only comparable to current state-of-the-art approaches but also generalizable over both problem size and problem class.

Currently, both the branching process and the cutting process have received attention from the machine learning community. However, they are often considered completely separate. It can be interesting and fruitful to study the dependencies between variable/node selection in the branching process and cut selection in the cutting process.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40

References

1. Huang, Z., Wang, K., Liu, F., Zhen, H.L., Zhang, W., Yuan, M., Hao, J., Yu, Y. and Wang, J., 2022. Learning to select cuts for efficient mixed-integer programming. Pattern Recognition, 123, p.108353.
2. Tang, Y., Agrawal, S. and Faenza, Y., 2020, November. Reinforcement learning for integer programming: Learning to cut. In International conference on machine learning (pp. 9367-9376). PMLR.
3. Khalil, E.B., Dilkina, B., Nemhauser, G.L., Ahmed, S. and Shao, Y., 2017, August. Learning to Run Heuristics in Tree Search. In Ijcai (pp. 659-666).
4. He, H., Daume III, H. and Eisner, J.M., 2014. Learning to search in branch and bound algorithms. Advances in neural information processing systems, 27.
5. Balcan, M.F., Dick, T., Sandholm, T. and Vitercik, E., 2018, July. Learning to branch. In International conference on machine learning (pp. 344-353). PMLR.
6. Khalil, E., Le Bodic, P., Song, L., Nemhauser, G. and Dilkina, B., 2016, February. Learning to branch in mixed integer programming. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 30, No. 1).
7. Pan, C.H., 1997. A study of integer programming formulations for scheduling problems. International Journal of Systems Science, 28(1), pp.33-41.
8. Guihaire, V. and Hao, J.K., 2008. Transit network design and scheduling: A global review. Transportation Research Part A: Policy and Practice, 42(10), pp.1251-1273.
9. Díaz-Madroñero, M., Mula, J. and Peidro, D., 2014. A review of discrete-time optimization models for tactical production planning. International Journal of Production Research, 52(17), pp.5171-5205.
10. Huang, L., Chen, X., Huo, W., Wang, J., Zhang, F., Bai, B. and Shi, L., 2021. Branch and bound in mixed integer linear programming problems: A survey of techniques and trends. arXiv preprint arXiv:2111.06257.
11. Wesselmann, F. and Stuhl, U., 2012. Implementing cutting plane management and selection techniques. In Technical Report. University of Paderborn.
12. Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J. and Lübbecke, M., 2021. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. Mathematical Programming Computation, 13(3), pp.443-490.
13. Paulus, M.B., Zarpellon, G., Krause, A., Charlin, L. and Maddison, C., 2022, June. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In International conference on machine learning (pp. 17584-17600). PMLR.
14. Atamtürk, A., 2003. On the facets of the mixed–integer knapsack polyhedron. Mathematical Programming, 98(1-3), pp.145-175.
15. Forrest, J. coin-or/Cbc: Release releases/2.10.8. Zenodo. https://doi.org/10.5281/zenodo.6522795. 2022
16. Atamtürk, A. and Munoz, J.C., 2004. A study of the lot-sizing polytope. Mathematical Programming, 99(3), pp.443-465.
17. Arik, S.Ö. and Pfister, T., 2021, May. Tabnet: Attentive interpretable tabular learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 8, pp. 6679-6687).
18. Mitchell, J.E., 2002. Branch-and-cut algorithms for combinatorial optimization problems. Handbook of applied optimization, 1(1), pp.65-77.
19. Diaby, M., 2010. Linear programming formulation of the set partitioning problem. International Journal of Operational Research, 8(4), pp.399-427.

ICCS Camera Ready Version 2023
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-36024-4_40