Learning to Stop Cut Generation for Efficient Mixed-Integer Linear Programming

Haotian Ling, Zhihai Wang, Jie Wang[*]

University of Science and Technology of China
{haotianling,zhwangx}@mail.ustc.edu.cn
jiewangx@ustc.edu.cn

Abstract

Cutting planes (cuts) play an important role in solving mixed-integer linear programs (MILPs), as they significantly tighten the dual bounds and improve the solving performance. A key problem for cuts is when to stop cuts generation, which is important for the efficiency of solving MILPs. However, many modern MILP solvers employ hard-coded heuristics to tackle this problem, which tends to neglect underlying patterns among MILPs from certain applications. To address this challenge, we formulate the cuts generation stopping problem as a reinforcement learning problem and propose a novel **hy**brid **g**raph **r**epresentation m**o**del (**HYGRO**) to learn effective stopping strategies. An appealing feature of HYGRO is that it can effectively capture both the dynamic and static features of MILPs, enabling dynamic decision-making for the stopping strategies. To the best of our knowledge, HYGRO is *the first* data-driven method to tackle the cuts generation stopping problem. By integrating our approach with modern solvers, experiments demonstrate that HYGRO significantly improves the efficiency of solving MILPs compared to competitive baselines, achieving up to 31% improvement.

1 Introduction

Mixed-Integer Linear Programming (MILP) is a widely-used mathematical optimization model employed to solve various real-world problems, such as production planning (Pochet and Wolsey 2010), vehicle routing (Laporte 2009), project scheduling (Berthold et al. 2010), and facility location (Abend 2016). A MILP aims to find the values of the decision variables, including integer and continuous, to optimize a linear objective function while satisfying all the linear constraints. A standard MILP takes the form of

$$\min_{\mathbf{x}}\{\mathbf{c^T x}|\mathbf{A x} \le \mathbf{b}, x_j \in \mathbb{Z}, \forall j \in I\}, \quad (1)$$

where $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m$ and $\mathbf{A} \in \mathbb{R}^{m \times n}$. $\mathbf{x} \in \mathbb{R}^n$ represents a vector of decision variables, where a subset denoted as $\{x_j | j \in I, I \subseteq \{1, 2, \ldots, n\}\}$ consists of integer decision variables, while the remaining decision variables are continuous. MILP solving initiates with its Linear Programming (LP) relaxation, where the integer decision variables in Definition (1) are treated as continuous. The specific form of LP relaxation can be found in the Definition (2).

A key component in modern MILP solvers is the cutting plane method. The cutting plane method aims to generate valid linear constraints—known as **cutting planes** (cuts)—that are able to tighten the LP relaxation of the original problem. They are usually repeatedly generated and added to the original problem in multiple rounds, as illustrated in Figure 1, which has a significant impact on the efficiency of solving MILPs (Achterberg 2009; Wesselmann and Stuhl 2012).

Recently, using machine learning (ML) to improve the cutting plane method in modern MILP solvers has been an active topic of significant interest. Many existing methods (Tang, Agrawal, and Faenza 2020; Huang et al. 2022; Paulus et al. 2022; Turner et al. 2022; Wang et al. 2023a) introduced machine learning into the selection of cuts. Besides, Berthold, Francobaldi, and Hendel (2022) proposed to dynamically determine whether to use local cuts. As for the theoretical analysis, Balcan et al. (2021) provided provable guarantees for learning high-performing cuts. They have demonstrated the strong ability to enhance MILP solvers via incorporating ML into the cutting plane method.

However, the stopping strategy for the cutting plane method—which significantly impacts the efficiency of solving MILPs (see Section 3)—has received limited attention. On one hand, stopping cuts generation too early often struggles to effectively tighten the LP relaxation, making it difficult for the solver to efficiently find optimal solutions (Nemhauser and Wolsey 2020). On the other hand, adding too many cuts leads to a significant increase in the size of MILPs, posing a computational problem and thus degrading the solving efficiency (Wesselmann and Stuhl 2012). Although many modern solvers employ stopping strategies based on hard-coded heuristics, they often tend to neglect underlying patterns among MILPs from certain applications (Bengio, Lodi, and Prouvost 2021; Gupta et al. 2022).

To address this challenge, we propose a novel **hy**brid **g**raph **r**epresentation m**o**del (HYGRO) to learn intelligent stopping strategies based on a novel transformed reinforcement learning formulation. To the best of our knowledge, HYGRO is *the first* data-driven approach to tackle the cuts generation stopping problem, which significantly improves the efficiency of solving MILPs. Specifically, we first transform the stopping strategy learning problem into a reinforcement learning problem to learn simple and efficient stopping conditions, which can significantly reduce the action

---

[*]Corresponding author.

space and thus improve the performance of learned models (see Section 5.2). Then, we propose a novel **hy**brid **g**raph **r**epresentation m**o**del (HYGRO) to learn effective stopping conditions, which captures the underlying patterns among MILPs by integrating their dynamic graph embeddings with their static intrinsic features. We incorporate our proposed HYGRO into the open-source state-of-the-art MILP solver, namely SCIP (Bestuzheva et al. 2021). Extensive experiments demonstrate that HYGRO significantly improves the efficiency of solving MILPs, achieving up to 31% improvement, on six challenging MILP problem benchmarks compared to eight competitive baselines.

We summarize our major contributions as follows. (1) We propose a novel **hy**brid **g**raph **r**epresentation m**o**del (HYGRO) to learn intelligent stopping strategies based on a novel transformed reinforcement learning formulation. (2) To the best of our knowledge, HYGRO is *the first* approach to learn cuts generation stopping strategies, which significantly improves the efficiency of solving MILPs. (3) Experiments demonstrate that HYGRO significantly improves the efficiency of solving MILPs compared to eight competitive baselines on six challenging MILP benchmarks.

2 Background

2.1 Cutting Plane Method

The cutting plane method (Gomory 1958) is employed by modern MILP solvers to iteratively add valid cutting planes to the original MILP problem. Let's start by considering the LP relaxation of the MILP problem. where we treat the decision variables in Definition (1) as continuous, leading to:

$$\min_{\mathbf{x}}\{\mathbf{c^T x}|\mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^n\}, \qquad (2)$$

where $\mathbf{x}$ represents a vector of continuous decision variables. The optimal value of the LP relaxation is termed the **dual bound**. Specifically, if we find an optimal solution $\mathbf{x}^{lp}$ for the relaxation problem, the term $\mathbf{c^T x}^{lp}$ is referred to as the dual bound. Besides, the optimal achievable objective function value during the solving process of the original MILP problem is called the **primal bound**. More precisely, denoting the current optimal solution that satisfies all constraints as $\mathbf{x}^*$, then $\mathbf{c^T x}^*$ represents the primal bound.

In the cutting plane method, valid inequalities are generated from the constraints of the original MILP problem using algorithms like Gomory Cuts (Gomory 1958), added to strengthen the relaxation (Achterberg 2009), resulting in tighter LP relaxation. The process of cut generation and addition **(G&A)** is performed iteratively until a stopping condition is triggered. A cut is represented as: $\sum_{j \in S} a_j x_j \leq d$, where a_j denotes the coefficients in the cut, S represents a set of decision variable indices, and d is the right-hand side of the cut. The set of all cuts generated in a single iteration is denoted as $\mathcal{C}$. From $\mathcal{C}$, we select a subset $\mathcal{C}^*$ to be added to the original MILP problem. Commonly, in modern solvers like SCIP (Bestuzheva et al. 2021), a method called "Branch and Cut" (**B&C**) (Padberg and Rinaldi 1991) is employed.
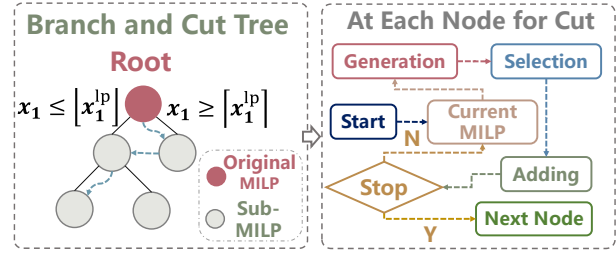

Figure 1: The right gray dashed box illustrates main steps related to cutting planes that each node needs to undergo.

2.2 Stopping Strategies in Cutting Plane Method

We denote π_S as a cutting plane stopping strategy. In modern solvers, within a B&C tree of an MILP instance, at node i, the G&A of cutting planes occurs iteratively. In each iteration, π_S decides whether to proceed to the next iteration. If an iteration is interrupted by π_S, the process will move to the next node j. Related illustrations can be found in Figure 1. Here are several classical heuristic stopping strategies.

Fixed Cut Numbers (FCN)   The strategy will stop the cut generation after a specific number of cuts have been added, ensuring that the total number of added cuts doesn't exceed the predefined limit. Assuming we set the number of cuts to be added as k, then at each node, we add a maximum of k cuts. Specifically, let C_i represent the set of cuts added at node i, then the FCN will have $|C_i| \leq k$, where $|C_i|$ represents the number of cuts added at node i.

Fixed Cut Rounds (FCR)   SCIP uses a round-based approach to add multiple cuts to the original problem. Under the fixed cutting plane rounds strategy, a finite number of rounds for cut generation are executed at each node. Assuming we set a threshold t, then at each node, we will perform a maximum of t rounds for cut generation. To be precise, denoting the actual round for cut generation at node i as t_i, this stopping strategy follows the constraint: $t_i \leq t$.

Stagnation Round Detection (SRD)   The two aforementioned strategies have the advantage of simplicity, yet their drawbacks become apparent due to their inability to consider the dynamic factors during the solving process.

Let C_i^t be the set of cutting planes generated in the tth round at node i. Define $\mathcal{O}^*(C_i^t)$ as the objective value resulting from the tth round of cutting plane generation. Furthermore, a threshold ϵ is set to determine whether the objective value improvement is significant. If we have:

$$\frac{|\mathcal{O}^*(C_i^t) - \mathcal{O}^*(C_i^{t+1})|}{|\mathcal{O}^*(C_i^t)|} \leq \epsilon, \qquad (3)$$

then we consider that the solving process has stagnated at the $(t+1)^{th}$ round. We define the number of consecutive stagnation rounds at node i in the round t^{th} round as s_i^t, and we set $s_i^0 = 0$. If the stagnation persists into the $(t+2)^{th}$ round, we update $s_i^{t+2} = s_i^{t+1} + 1$; otherwise, $s_i^{t+2} = 0$. Let s_i be the maximum allowed consecutive stagnation rounds at node i. The stagnation round detection strategies is formulated as: $s_i^r \leq s_i$, where r represents any given round.
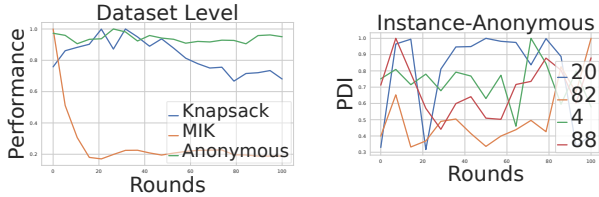
Figure 2: Lower values indicate better performance on the figures. Data for both figures was normalized. The legend in the right figure represents the instance numbers in the Anonymous dataset.

3 Motivation Results

Empirically, different stopping strategies significantly affect the solving performance of MILP problems. We conducted carefully designed experiments to evaluate this impact.

3.1 Setup

To visually demonstrate the impact of stopping strategies on the solving efficiency of MILPs using the cutting plane method, we conducted exhaustive search experiments within a specified range on three datasets: Anonymous, MIK, and Knapsack, using the SCIP solver. For detailed descriptions of the datasets, please refer to Section 5.1. We employ the Fixed Cut Round strategy, adding cutting planes only at the root node for each instance, with a fixed number of iterations as mentioned in Section 2.2. For each instance, we perform 100 runs. In the jth run, for every instance $\mathcal{I}$, we execute a maximum of j iterations for the G&A of cutting planes at the root node. The Anonymous dataset employs primal-dual bound gap integral (PDI) (Wang et al. 2023a), while other datasets represent solving performance using solving time.

3.2 Results

From Figure 2 (Left), it is evident that as the number of cutting plane G&A rounds varies, the trend in solving performance varies across different datasets. This indicates that different types of MILP problems need diverse stopping strategies to achieve enhanced performance. Furthermore, Figure 2 (Right) demonstrates that even within a single dataset, the performance trends across different instances vary significantly, which emphasizes the requirement for instance-specific stopping strategies instead of solely relying on predefined hardcoded strategies. The analyses conducted on both the dataset level and individual MILP instances clearly demonstrate the importance of using learning-based methods to develop more effective stopping strategies.

4 Method

4.1 Motivation

Our primary task involves dynamically determining the optimal moment to stop the G&A of cutting planes at each node within the B&C tree, enabling efficient solving of MILP. Achieving this goal necessitates a more intelligent stopping strategy. A more intuitive approach is to allow the model to decide, after each G&A of cutting planes, whether to proceed to the next iteration. However, this method presents two notable drawbacks. **Firstly**, direct decision-making regarding stopping would result in an increased frequency of invoking the trained model, resulting in elevated computational costs that cannot be disregarded (Gupta et al. 2020). **Additionally**, if the model were to directly determine whether to stop, given a maximum of k iterations of cutting plane G&A at the current node, the potential action space would exponentially increase to 2^k, making it challenging for the model to learn effective stopping strategies.

To address the aforementioned challenges, we innovatively transformed our approach, shifting from directly learning whether to stop to indirectly learning simple and efficient stopping conditions. This transformation draws inspiration from the SRD strategy mentioned in Section 2.2, incorporating prior knowledge that if cutting planes are unable to effectively tighten the LP relaxations, their G&A should be stopped. Specifically, at each node, the model learns to dynamically determine the stopping condition, which is a threshold for the maximum number of consecutive stagnation rounds, instead of relying on a predetermined value. The above transformation significantly improves the performance of learned models while also greatly lowering computational overhead. The effectiveness of this transformation is demonstrated through experiments in Section 5.2.

4.2 Problem Formulation

Reinforcement learning excels at decision-making tasks, achieving a series of successes (Zhou, Li, and Wang 2020; Yang et al. 2022; Haarnoja et al. 2018; Wang et al. 2022, 2023b; Kuang et al. 2022; Liu et al. 2023; Zhou et al. 2020) and finding increasing applications in specific tasks (Adriaensen et al. 2022). We consider the stopping strategy as a reinforcement learning problem as well.

To learn the aforementioned stopping strategy, we formulate the stopping problem as a Markov Decision Process (MDP) (Bellman 1957; Sutton and Barto 1998). Specifically, we define the state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r, state transition function f, and terminal state $\mathcal{T}$ as follows.

State Space We employ a four-tuple $\{\mathcal{C}, \mathcal{V}, \mathcal{E}, \mathcal{S}^*\}$ to denote the state space $\mathcal{S}$ of the current node, where $\mathcal{C}, \mathcal{V}$, and $\mathcal{E}$ correspond to the dynamic part of $\mathcal{S}$, capturing the features of constraints, variables, and coefficients, respectively. We model these dynamic components using a bipartite graph. $\mathcal{S}^*$ signifies the static part, consisting of the intrinsic attributes of the MILP problem, and we represent these attributes using a vector. The schematic diagram illustrating these four components is presented in Figure 3, and their specific forms will be elaborated upon in Section 4.3.

Action Space The formulation of our action space, represented as $\mathcal{A}$, is based on the discussions regarding method transformation in Section 4.1. At node i, a single action $a_i \in [0, t_{\mathcal{A}})$ is taken, where a_i is an integer representing the learned stopping condition, i.e., maximum consecutive stagnation rounds for node i. The value of $t_{\mathcal{A}}$ corresponds to the size of the action space $\mathcal{A}$. This action a_i guides the
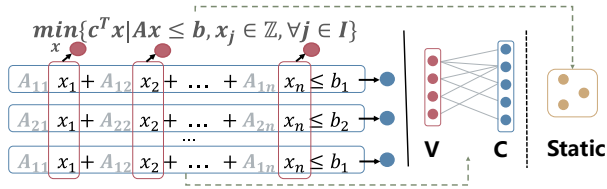
Figure 3: The red, blue, and yellow circles represent variable, constraint, and static features, respectively.

solver to iteratively generate and add cutting planes at node i until the stopping condition indicated by a_i is triggered.

State Transition   A state transition takes place when the number of consecutive stagnation rounds at node i exceeds the threshold set by action a_i, or when other specific conditions are triggered at node i. The state transition function f_t outlines the process of transiting from state $\mathcal{S}_i$ at node i to state $\mathcal{S}_j$ at node j within the Branch and Cut tree: $\mathcal{S}_j = f_t(\mathcal{S}_i, a_i)$. As illustrated in Figure 1, the blue dashed curved line indicates the successor in the state transition, which corresponds to the node where the subsequent iteration of cutting plane G&A takes place.

Terminal State   Upon completing the full solving process for the instance $\mathcal{I}$, which could involve achieving the optimal feasible solution or reaching a predefined time limit, we define this occurrence as a transition to the terminal state $\mathcal{T}$.

Reward   For a given MILP instance $\mathcal{I}$, the agent performs sequential decision-making at multiple nodes, generating a sequence of actions. Before transiting to the terminal state, the immediate rewards for each action are set to 0. Upon transiting to the terminal state, we compute the reward signal r to holistically assess the quality of these actions. Here, a value v quantifies the effectiveness of these actions, associated with solving performance metrics, such as solving time and PDI. The exact approach for computing the reward signal will be elaborated upon in Section 4.3.

4.3   Learning to Stop

In this section, we will present the framework of HYGRO and our strategy for training HYGRO.

Hybrid Graph Representation Model   An MILP problem $\mathcal{I}$, corresponding to a node i in the B&C tree, has underlying patterns that can be represented using the four-tuple $\{\mathcal{C}, \mathcal{V}, \mathcal{E}, \mathcal{S}^*\}$. The bipartite graph comprising $\mathcal{C}, \mathcal{V}$, and $\mathcal{E}$ is an effective model for representing MILP problems, widely employed in MILP research (Gasse et al. 2019). In fact, static intrinsic attributes $\mathcal{S}^*$, such as the scale of $\mathcal{I}$, also play a vital role in the node-level representation of MILP (Berthold, Francobaldi, and Hendel 2022). In order to comprehensively encode $\mathcal{I}$, we propose a novel HYGRO that integrates dynamic graph embeddings with static intrinsic features to learn efficient stopping conditions.

Specifically, assuming there are n constraints each with d_1 features, m variables each with d_2 features, and e nonzero coefficients, we can represent the $\{\mathcal{C}, \mathcal{V}, \mathcal{E}\}$. using matrices $\mathbf{C}^{n \times d_1}$, $\mathbf{V}^{m \times d_2}$, and $\mathbf{E}^{e \times 1}$, collectively forming the

bipartite graph of Figure 3. The bipartite graph formed by these three components undergoes internal interactions, thus we term it the **dynamic part**. As for intrinsic attributes of $\mathcal{I}$, such as its scale, total d_3 in count. To represent these attributes, we employ a vector $\mathbf{s}^{d_3}$. Since $\mathbf{s}^{d_3}$ remains independent throughout the state encoding procedure, we term it the **static part**. Then we proceed to perform a linear transformation on $\{\mathbf{C}^{n \times d_1}, \mathbf{V}^{m \times d_2}, \mathbf{E}^{e \times 1}, \mathbf{s}^{d_3}\}$. This transformation uniformly embeds these four components into a d-dimensional space, resulting in $\{\mathbf{C}^{n \times d}, \mathbf{V}^{m \times d}, \mathbf{E}^{e \times d}, \mathbf{s}^d\}$.

For the bipartite graph part, we define the graph convolution (Kipf and Welling 2017; Defferrard, Bresson, and Vandergheynst 2016) operation as : $h_{\theta g}^{(l)}(\mathbf{left}, \mathbf{right}, \mathbf{edge})$, where θ^g represents the parameters of h, h denote the information propagation between the two sides on layer l, and $\mathbf{edge}$ represents the associations between $\mathbf{left}$ and $\mathbf{right}$. Then, the flow of information can be described as: $h_{\theta g}^{(1)}(\mathbf{C}^{n \times d}, \mathbf{V}^{m \times d}, \mathbf{E}^{e \times d})$, $h_{\theta g}^{(2)}(\mathbf{V}^{m \times d}, \mathbf{C}^{n \times d}, \mathbf{E}^{e \times d})$. Then we leverage the Principal Neighbourhood Aggregation (PNA) (Corso et al. 2020) to aggregate the $\mathbf{C}^{n \times d}$ and $\mathbf{V}^{m \times d}$ into vectors $\mathbf{c}_a^{4d}$ and $\mathbf{v}_a^{4d}$ respectively. Specifically, let $[\cdot]$ represent the concatenation between different vectors, our aggregation method can be described as:

$$\mathbf{d}_a^{4d} = [\max(\mathbf{D}^{q \times d}), \min(\mathbf{D}^{q \times d}), \mathrm{avg}(\mathbf{D}^{q \times d}), \mathrm{std}(\mathbf{D}^{q \times d})],$$

where the four terms in $[\cdot]$ respectively denote taking the maximum value, the minimum value, the average value, and the standard deviation along the columns of matrix $\mathbf{D}^{q \times d}$.

Finally, we employ a Multilayer Perceptron (MLP) (Rosenblatt 1963), denoted as $\mathcal{M}$, serving as the output module, we use: $ratio = 0.5 * \sigma(\mathcal{M}([\mathbf{v}_a^{4d}, \mathbf{c}_a^{4d}, \mathbf{s}^d])) + 0.5$ to compute the output scalar $ratio$, where σ is the $tanh$ activation function (Rumelhart, Hinton, and Williams 1986; Goodfellow, Bengio, and Courville 2016), mapping any real number to the interval (-1, 1). Thus, we will have $ratio \in (0, 1)$. The HYGRO's architecture is illustrated in Figure 4. Additionally, if the current node is not the root node, then $ratio = ratio \times \gamma$, where γ is a constant less than 1. This is based on prior knowledge that cutting planes added at the root node are often more critical. Finally, we let the action at node i to be $a_i = \lfloor ratio \cdot t_{\mathcal{A}} \rfloor$, where $\lfloor \cdot \rfloor$ represents the floor operation, $t_{\mathcal{A}}$ represents the size of the action space.

Training   We use the Evolutionary Strategies (ES) (Rechenberg 1973) to train HYGRO. We opted for ES due to its simplicity in training, circumventing the need for explicit gradient computations, enabling us to exploit training parallelism to its fullest extent (Salimans et al. 2017). We treat the training process of HYGRO as a black-box optimization problem, where we input the parameters θ_0 of HYGRO and obtain perturbations ε along with the corresponding reward signals $\mathbf{r}$ induced by them (Tang, Agrawal, and Faenza 2020). Specifically, for the HYGRO $\mathcal{H}_0$, we generate k perturbations ε: $\{\epsilon_1, \epsilon_2, ..., \epsilon_k\}$, and then introduce them into $\mathcal{H}_0$ to create: $\{\mathcal{H}_0^{\epsilon_1}, \mathcal{H}_0^{\epsilon_2}, ..., \mathcal{H}_0^{\epsilon_k}\}$. Subsequently, we integrate these new models into the SCIP solver respectively, obtaining the solving performance, such as solving time, $\mathbf{p}$ of each new model: $\{p_1, p_2, ..., p_k\}$. Then, we compute the reward r_i for each perturbation ϵ_i using $\mathbf{r} = \mathrm{softmax}(\mathbf{p})$,
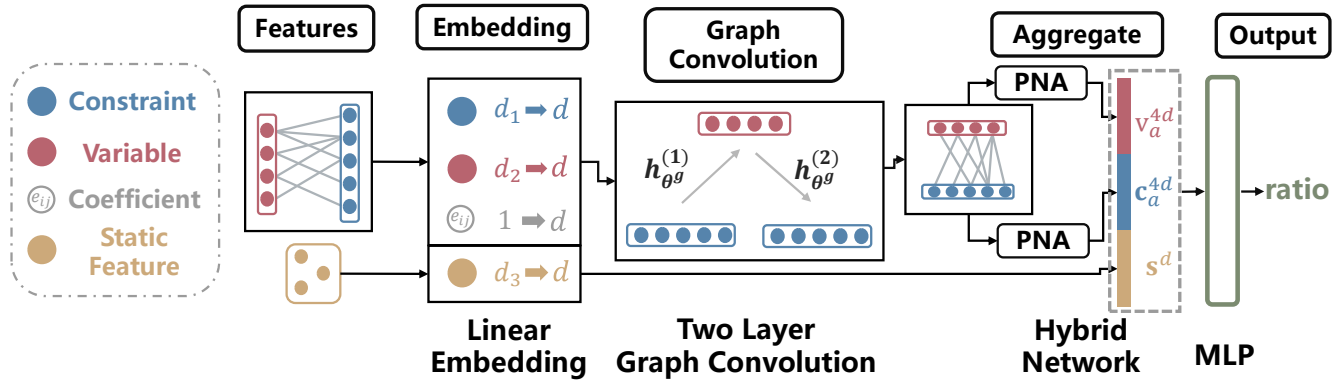
Figure 4: Diagram illustrating the structure of HYGRO. First, it employs a two-layer graph neural convolutional network to encode the dynamic part. Then, it aggregates the dynamic part with the static part and finally outputs the $ratio$. In summary, HYGRO takes the four features shown in the diagram as inputs and generates the $ratio$ value for the current node as output.

leveraging $\mathbf{r}$ and ε to approximate implicit gradient $\hat{g}_\theta$, iteratively adjusting the parameters θ_0 of HYGRO.

5 Experiment

We designed experiments to evaluate the effectiveness of our approach and assess the efficiency of stopping conditions learned by HYGRO. Our experiments mainly focus on the following four aspects. Here is a brief introduction: **(1) Solving Performance:** We conducted experiments comparing the solving performance of our method with eight competitive baselines on six NP-hard datasets. **(2) Ablation Study:** As mentioned earlier in Section 4.1, we designed ablation experiments to validate the effectiveness of the transformation. **(3) Generalization Study:** We evaluate the ability of HYGRO to generalize across different sizes of MILPs. **(4) Visualization:** We compared the solving performance achieved by the HYGRO with the optimal performance obtained through an exhaustive search and depicted the results.

5.1 Experiment Details

Dataset We employed six NP-hard problem datasets as benchmarks, which are categorized into two groups.

Classical NP-hard combinatorial optimization problems widely served as benchmarks, including Multiple Knapsack Problem (Kellerer, Pferschy, and Pisinger 2004), Maximum Independent Set (MIS) (Hartmanis 1982), and Set Covering (Balas and Ho 1980). These datasets are artificially generated using the methods provided by (Prouvost et al. 2020).

Harder datasets, including Corlat (Atamtürk 2003), MIK (Gomes, van Hoeve, and Sabharwal 2008), and Anonymous (Gasse et al. 2022) dataset. These datasets are related to real-world optimization problems, often employed as benchmarks in MILP research (Nair et al. 2021).

Setup We integrate HYGRO into SCIP, a widely utilized backend for MILP research (Gasse et al. 2019; Paulus et al. 2022; Wang et al. 2023a), and one of the state-of-the-art open-source solvers. During solving, we set a 300-second time limit for each MILP instance. HYGRO's training utilizes the ADAM optimizer (Kingma and Ba 2014) through

PyTorch (Paszke et al. 2019). To ensure fairness and replicability, we maintain default SCIP parameter settings, except for those pertaining to the cutting plane stopping strategy. We divided each dataset into training and testing sets with 75% and 25% of instances, respectively, and selected the best model trained on the training set for testing.

In the formulation presented in Section 4.2, we consider making decisions in each node. In fact, many studies concentrate primarily on decisions made at the root node (Tang, Agrawal, and Faenza 2020). On one hand, this approach will reduce computational overhead; on the other hand, cutting planes added at the root node tend to have a greater impact on solving efficiency. However, for experimental completeness, in the method HYGRO I, we make decisions for cutting plane stopping at multiple nodes (up to a depth of 2) – whereas all other methods only focus on the root node.

Performance metrics In this study, we assess each method's performance using two key metrics: solving time and PDI. We employ the metric of average improvement to assess the overall performance of each method. Specifically, we calculated the average percentage performance improvement of a method across all datasets in this experiment.

Baseline In our experiments, we employed several baseline strategies, including: **(1) Default**: The default cutting plane stopping strategy offered by the SCIP solver, utilizing the Stagnation Round Detection strategy. **(2) No Cuts:** This strategy disables the G&A of cuts. **(3) Always:** This strategy performs G&A of cuts without any conditions until a specific event of SCIP is triggered. **(4) FCN:** As discussed in Section 2.2, this strategy ensures that the number of cuts added in each run does not exceed 200. **(5) FCR:** As explained in Section 2.2, this strategy ensures that the number of cutting plane rounds in each run does not exceed 100. **(6) Immediate:** This strategy involves setting the value of ϵ in Inequality (3) to a very small value of 10^{-5}, and as soon as it becomes smaller than ϵ, cutting plane G&A is immediately stopped. **(7) Random I:** This strategy incorporates a 0.5% probability of instantly stopping the G&A of cutting planes before each iteration. **(8) Random II:** This strategy involves

	Classical Dataset									
	Set Covering (1000V, C=500)			Max Independent Set (500V, C≈2000)			Knapsack (1000V, C≈200)			Imprv.
Method	Time(s)↓	PDI↓	Imprv.(%)↑	Time(s)↓	PDI↓	Imprv.(%)↑	Time(s)↓	PDI↓	Imprv.(%)↑	Avg.(%)
HYGRO*	**3.91(3.0)**	**41.94(27.2)**	**24.08**	**5.78(3.8)**	**31.75(21.5)**	**10.53**	**9.12(12.6)**	**17.67(14.3)**	**31.63**	**22.08**
HYGRO I*	**3.99(2.3)**	47.75(26.4)	**22.52**	**5.61(3.4)**	**29.18(19.9)**	**13.16**	10.34(12.5)	19.23(16.3)	22.48	**19.39**
Default	5.15(3.5)	55.23(29.4)	N/A	6.46(4.1)	33.62(23.4)	N/A	13.34(15.6)	22.17(18.9)	N/A	N/A
No Cuts	4.04(2.8)	**43.86(26.1)**	21.55	9.71(8.4)	74.29(59.8)	-50.31	11.56(13.5)	20.14(16.8)	13.34	-5.14
Always	9.56(4.9)	89.01(50.7)	-85.63	6.64(4.5)	34.44(23.5)	-2.79	40.25(49.3)	37.27(22.6)	-201.72	-96.71
FCN	9.58(4.9)	87.84(51.5)	-86.02	6.53(4.5)	32.33(21.8)	-1.08	32.74(37.8)	34.39(18.7)	-145.43	-77.15
FCR	9.81(5.2)	88.31(49.4)	-90.48	6.66(4.3)	33.14(22.7)	-3.10	16.25(16.0)	23.84(14.2)	-21.81	-38.46
Immediate	7.10(4.1)	70.64(40.3)	-37.86	6.63(4.3)	35.46(25.2)	-2.63	**7.74(13.3)**	**16.89(10.8)**	**41.97**	0.49
Random I	9.08(5.0)	84.07(48.6)	-76.31	7.04(6.8)	37.52(39.6)	-8.98	28.25(28.9)	26.57(13.8)	-111.77	-65.69
Random II	8.56(4.8)	81.85(49.5)	-66.21	6.81(4.4)	35.73(24.2)	-5.42	13.93(15.4)	19.05(12.1)	-4.42	-25.35
	Harder Dataset									
	Corlat			MIK			Anonymous			Imprv.
Method	Time(s)↓	PDI↓	Imprv.(%)↑	Time(s)↓	PDI↓	Imprv.(%)↑	Time(s)↓	PDI↓	Imprv.(%)↑	Avg.(%)
HYGRO*	**37.7(77.5)**	1235(2893)	**17.3**	**9.5(14.0)**	**59.0(82.8)**	**7.4**	241(102)	**14973(7984)**	1.6	**8.8**
HYGRO I*	40.3(82.3)	1035(3313)	11.8	**8.5(12.0)**	**51.6(70.8)**	**16.9**	237(109)	**14126(8054)**	**3.1**	**10.6**
Default	45.6(85.8)	1234(4078)	N/A	10.2(14.3)	64.3(87.8)	N/A	245(96)	16247(8624)	N/A	N/A
No Cuts	123.7(138.3)	3228(6788)	-171.1	108.5(132.1)	337.6(427.3)	-961.3	**238(108)**	15701(8763)	3.0	-376.5
Always	**32.8(65.9)**	**889(1259)**	**28.1**	25.9(26.1)	88.7(84.4)	-153.4	253(91)	16704(8766)	-3.2	-42.8
FCN	41.8(82.7)	1141(2108)	8.4	41.4(69.3)	110.3(120.9)	-305.1	252(92)	17313(9730)	-3.0	-99.9
FCR	51.5(100.4)	1742(4901)	-12.9	23.5(26.2)	88.9(94.7)	-129.5	253(86)	15410(7715)	-3.4	-48.6
Immediate	52.7(91.8)	1410(4401)	-15.5	13.9(20.7)	73.6(95.4)	-36.2	238(108)	16245(9151)	3.0	-16.2
Random I	42.6(83.6)	1106(3553)	6.7	22.6(23.5)	88.1(90.4)	-121.3	261(84)	16195(7991)	-6.4	-40.3
Random II	41.9(85.4)	**995(2280)**	8.3	19.2(24.1)	84.0(88.6)	-88.0	245(103)	15235(8780)	-0.0	-26.6

Table 1: We presented the data for Experiments 1 and 2, which involve evaluations on classical datasets and harder datasets. The top two performances are highlighted in bold. The performance metrics are presented as average (standard deviation) for each test. Solving time improvement (Imprv.) over the default SCIP's stopping strategy and average solving time improvement (Imprv. Avg.) are listed. V and C denote variable and constraint counts, respectively. * represents our approach.

	MIK		Imprv.(%)
Metrics	Direct	Indirect (ours)	
Solving Time(s)↓	21.5(15.3)	**10.9(5.4)**	49.32
Extra Time(s)↓	7.0(5.8)	**1.9(2.6)**	72.70
PDI↓	403.0(458.4)	**239.2(262.7)**	40.62
Call Times↓	116.1(108.7)	**1.0(0.0)**	99.14
	MIS		Imprv.(%)
Metrics	Direct	Indirect (ours)	
Solving Time(s)↓	9.2(4.6)	**7.1(4.7)**	22.18
Extra Time(s)↓	2.9(2.4)	**1.1(2.2)**	63.44
PDI↓	55.4(36.2)	**45.0(34.0)**	18.82
Call Times↓	24.4(2.7)	**1.0(0.0)**	95.90

Table 2: The experimental results of the Ablation Study, the best-performing results are highlighted in bold.

generating a random integer t within the range of $[0, 30)$ to serve as the threshold for consecutive stagnation rounds.

Baseline strategies (1)-(5) are provided by SCIP, while the strategies (6)-(8) are designed by us.

5.2 Results

Experiment 1: Classical Datasets Performance Experiment To evaluate HYGRO's ability to find efficient stopping conditions for classical MILP problems, we conducted performance comparison experiments with eight baseline strategies. Our HYGRO strategies, including HYGRO I, consistently demonstrate remarkably superior performance, outperforming other strategies in the majority of tests. Although there were two occurrences where baseline strategies slightly outperformed HYGRO, our methods exhibited substantial performance improvements over these two strategies on other datasets. It is worth noting that the default strategy of SCIP, based on the SRD strategy, also proves to be highly effective as a stopping strategy, showing distinct advantages over other baseline strategies. Nevertheless, in comparison to this default strategy, HYGRO achieved an average improvement of approximately 22%, with the highest observed improvement reaching up to 31% on specific datasets.

Experiment 2: Harder Datasets Performance Experiment To assess HYGRO's performance on more challenging MILP problems related to real-world applications, we conducted comparative experiments with baseline strategies. Our HYGRO strategies consistently demonstrate the highest overall performance. Relative to the SCIP default strategy, which represents one of the best-performing baseline strategies, our methods achieved performance improvement of approximately 8% and 10%, along with a notable peak improvement of 17% on specific datasets. HYGRO exhibited a greater capacity for efficiently solving real-world MILP problems on more challenging datasets.

Experiment 3: Ablation Study In the previous Section 4.1, we transformed the problem of learning stopping strate-

	HYGRO (ours)		Default		Imprv.
Datasets	Time(s)↓	PDI↓	Time(s)↓	PDI↓	%(Time)
Set Covering I	**11.28**	**102.83**	14.42	133.28	21.77
(1000V, 1000C)	**(29.99)**	**(180.20)**	(24.69)	(155.80)	
Set Covering II	**26.74**	**239.19**	29.61	257.52	9.69
(1500V, 1000C)	**(45.93)**	**(348.32)**	(47.69)	(337.41)	
Set Covering III	**59.76**	**504.76**	64.23	550.48	6.96
(2000V, 1000C)	**(84.28)**	**(669.90)**	(84.52)	(671.31)	
MIS I	28.05	105.93	**27.08**	**102.80**	-3.58
(750V, 3200C)	(24.95)	(84.28)	**(24.70)**	**(82.50)**	
MIS II	**141.13**	**394.57**	147.68	409.31	4.43
(1000V, 4200C)	**(128.76)**	**(393.85)**	(126.69)	(391.22)	
MIS III	**724.34**	**1965.79**	728.61	1972.64	0.58
(1500V, 6400C)	**(172.01)**	**(807.40)**	(162.21)	(802.92)	

Table 3: The generalization ability of HYGRO, the best-performing results are highlighted in bold. In the MIS dataset, the count of constraints (C) is an approximate value.

gies. We refer to the methods before and after this transformation as **"Direct"** and **"Indirect"**, respectively. To validate the effectiveness of this transformation, we compared the performance of these two methods when using HYGRO.

Specifically, in the Direct method, we determine whether to continue the G&A of cutting planes based on the output of HYGRO. If the $ratio$ surpasses 0.5, the iteration stops immediately; otherwise, it continues. We evaluated the performance of these two methods on the MIK and MIS datasets. It is crucial to compare the extra computational costs of the two methods, thus we introduce the metric "Extra Time" to measure the total runtime of HYGRO during the solving process, and "Call times" to indicate the frequency of invoking HYGRO. It's worth noting that the hardware utilized in this experiment differs from that of the other experiments, resulting in slight performance deviations. Nevertheless, the experimental setup maintains its consistency and fairness.

The results are presented in Table 2, from which we can find that the Indirect method exhibits notable advantages in terms of Extra Time metrics, significantly reducing the increased computational overhead from using HYGRO. It still maintains a substantial advantage in overall runtime, indicating that through the transformation, we have learned a more efficient stopping strategy. Even considering only the pure MILP solving time (excluding the Extra Time), the Indirect method still holds a significant advantage. These facts strongly validate the effectiveness of our transformation.

Experiment 4: Generalization Ability To evaluate the ability of HYGRO to generalize across various sizes of MILPs, we conducted generalization experiments. We trained HYGRO on the dataset used in Experiment 1 and then assessed the performance of HYGRO and SCIP's default strategy on more challenging datasets within the same domain. Similar settings are commonly employed to evaluate the generalization ability of data-driven models for MILP, as seen in studies like (Gasse et al. 2019; Wang et al. 2023a). Each instance has an 800-second time limit.

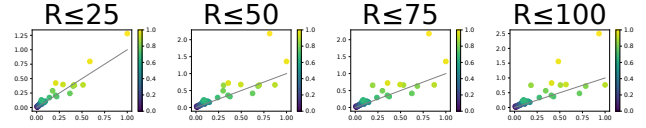The results of these experiments are elaborated in Table

Figure 5: The y-axis shows HYGRO's solving time, and the x-axis indicates the best performance within the specified range of rounds (R). Points below the gray dashed line signify superior performance for HYGRO.

3. It is evident that, apart from one occurrence where the default strategy of SCIP slightly outperformed our approach, HYGRO consistently performed better, achieving up to a 21% improvement in performance. As evident from the results, our approach significantly surpasses the heuristic default stopping strategy when applied to larger datasets. This highlights the robust generalization capacity of HYGRO.

Experiment 5: Visualization for Strategy Performance To present an intuitive portrayal of the performance of HYGRO, we conducted visualization experiments. For instances in the MIK test dataset, we conducted 100 solving runs. These run data were divided into four distinct sets: S_1, S_2, S_3, and S_4. Each set employs a maximum number of cutting plane rounds set at 25, 50, 75, and 100, correspondingly. For example, S_2 encompasses the solving performance data for all instances with cutting plane rounds varying from 1 to 50. MILPs in the MIK test dataset are represented by four data points $\{(P_H, P^*_{S_i}), i \in \{1, 2, 3, 4\}\}$, plotted on four individual scatter plots, respectively. Here, P_H denotes the solving time of HYGRO, while $P^*_{S_i}$ represents the best solving time within S_i.

The results are visualized in Figure 5. It is evident that in most cases, the scatter plots cluster around the vicinity of the gray dashed line, indicating that even when expanding the search range to 100, the performance achieved by HYGRO remains comparable to the optimal values obtained through exhaustive search. Moreover, in certain instances, HYGRO even demonstrates notable advantages. It is important to note that conducting such exhaustive searches comes with a substantial computational cost. This clearly illustrates that HYGRO has effectively learned an efficient stopping condition.

6 Conclusion

In this paper, we focus on the stopping strategy for the cutting plane method. Motivational experiments demonstrate the significance of stopping strategies and highlight the crucial role of learning-based methods. To learn more intelligent stopping strategies, we introduce an innovative transformation to learn a simple and efficient stopping condition and propose HYGRO to learn such conditions. Extensive experiments demonstrate the remarkable performance of HYGRO and provide evidence for the effectiveness of our transformation. We believe HYGRO offers a new perspective on improving the solving efficiency of MILP.

Acknowledgements

References

Abend, F. 2016. Facility Location Concepts Models Algorithms And Case Studies.

Achterberg, T. 2009. *Constraint Integer Programming*. Ph.D. thesis.

Adriaensen, S.; Biedenkapp, A.; Shala, G.; Awad, N. H.; Eimer, T.; Lindauer, M. T.; and Hutter, F. 2022. Automated Dynamic Algorithm Configuration. *J. Artif. Intell. Res.*, 75: 1633–1699.

Atamtürk, A. 2003. On the facets of the mixed–integer knapsack polyhedron. *Mathematical Programming*, 98(1): 145–175.

Balas, E.; and Ho, A. 1980. *Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study*, 37–60. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-00802-3.

Balcan, M.-F. F.; Prasad, S.; Sandholm, T.; and Vitercik, E. 2021. Sample Complexity of Tree Search Configuration: Cutting Planes and Beyond. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 4015–4027. Curran Associates, Inc.

Bellman, R. 1957. A Markovian Decision Process. *Indiana University Mathematics Journal*, 6: 679–684.

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

Berthold, T.; Francobaldi, M.; and Hendel, G. 2022. Learning to use local cuts. *arXiv preprint arXiv:2206.11618*.

Berthold, T.; Heinz, S.; Lübbecke, M. E.; Möhring, R. H.; and Schulz, J.-U. 2010. A Constraint Integer Programming Approach for Resource-Constrained Project Scheduling. In *Integration of AI and OR Techniques in Constraint Programming*.

Bestuzheva, K.; Besançon, M.; Chen, W.-K.; Chmiela, A.; Donkiewicz, T.; van Doornmalen, J.; Eifler, L.; Gaul, O.; Gamrath, G.; Gleixner, A.; Gottwald, L.; Graczyk, C.; Halbig, K.; Hoen, A.; Hojny, C.; van der Hulst, R.; Koch, T.; Lübbecke, M.; Maher, S. J.; Matter, F.; Mühmer, E.; Müller, B.; Pfetsch, M. E.; Rehfeldt, D.; Schlein, S.; Schlösser, F.; Serrano, F.; Shinano, Y.; Sofranac, B.; Turner, M.; Vigerske, S.; Wegscheider, F.; Wellner, P.; Weninger, D.; and Witzig, J. 2021. The SCIP Optimization Suite 8.0. Technical report, Optimization Online.

Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; and Veličković, P. 2020. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33: 13260–13271.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Lee, D.; Sugiyama, M.; Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Gasse, M.; Bowly, S.; Cappart, Q.; Charfreitag, J.; Charlin, L.; Chételat, D.; Chmiela, A.; Dumouchelle, J.; Gleixner, A.; Kazachkov, A. M.; Khalil, E.; Lichocki, P.; Lodi, A.; Lubin, M.; Maddison, C. J.; Christopher, M.; Papageorgiou, D. J.; Parjadis, A.; Pokutta, S.; Prouvost, A.; Scavuzzo, L.; Zarpellon, G.; Yang, L.; Lai, S.; Wang, A.; Luo, X.; Zhou, X.; Huang, H.; Shao, S.; Zhu, Y.; Zhang, D.; Quan, T.; Cao, Z.; Xu, Y.; Huang, Z.; Zhou, S.; Binbin, C.; Minggui, H.; Hao, H.; Zhiyu, Z.; Zhiwu, A.; and Kun, M. 2022. The Machine Learning for Combinatorial Optimization Competition (ML4CO): Results and Insights. In Kiela, D.; Ciccone, M.; and Caputo, B., eds., *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, 220–231. PMLR.

Gasse, M.; Chételat, D.; Ferroni, N.; Charlin, L.; and Lodi, A. 2019. Exact Combinatorial Optimization with Graph Convolutional Neural Networks. In *Neural Information Processing Systems*.

Gomes, C. P.; van Hoeve, W.-J.; and Sabharwal, A. 2008. Connections in Networks: A Hybrid Approach. In Perron, L.; and Trick, M. A., eds., *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 303–307. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-68155-7.

Gomory, R. 1958. Outline of an Algorithm for Integer Solutions to Linear Programs. *Bulletin of the American Mathematical Society*, 64: 275–278.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gupta, P.; Gasse, M.; Khalil, E.; Mudigonda, P.; Lodi, A.; and Bengio, Y. 2020. Hybrid Models for Learning to Branch. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 18087–18097. Curran Associates, Inc.

Gupta, P.; Khalil, E. B.; Chetélat, D.; Gasse, M.; Bengio, Y.; Lodi, A.; and Kumar, M. P. 2022. Lookback for learning to branch. *arXiv preprint arXiv:2206.14987*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 1861–1870. PMLR.

Hartmanis, J. 1982. Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson). *SIAM Review*, 24(1): 90–91.

Huang, Z.; Wang, K.; Liu, F.; Zhen, H.-L.; Zhang, W.; Yuan, M.; Hao, J.; Yu, Y.; and Wang, J. 2022. Learning to se-

lect cuts for efficient mixed-integer programming. *Pattern Recognition*, 123: 108353.

Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *Knapsack Problems*. ISBN 978-3-540-40286-2.

Kingma, D.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Kuang, Y.; Lu, M.; Wang, J.; Zhou, Q.; Li, B.; and Li, H. 2022. Learning Robust Policy against Disturbance in Transition Dynamics via State-Conservative Policy Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36: 7247–7254.

Laporte, G. 2009. Fifty Years of Vehicle Routing. *Transp. Sci.*, 43: 408–416.

Liu, Q.; Zhou, Q.; Yang, R.; and Wang, J. 2023. Robust Representation Learning by Clustering with Bisimulation Metrics for Visual Reinforcement Learning with Distractions. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press. ISBN 978-1-57735-880-0.

Nair, V.; Bartunov, S.; Gimeno, F.; von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; Addanki, R.; Hapuarachchi, T.; Keck, T.; Keeling, J.; Kohli, P.; Ktena, I.; Li, Y.; Vinyals, O.; and Zwols, Y. 2021. Solving Mixed Integer Programs Using Neural Networks. arXiv:2012.13349.

Nemhauser, G. L.; and Wolsey, L. A. 2020. Integer Programming.

Padberg, M.; and Rinaldi, G. 1991. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, 33(1): 60–100.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Neural Information Processing Systems*.

Paulus, M. B.; Zarpellon, G.; Krause, A.; Charlin, L.; and Maddison, C. 2022. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In *International conference on machine learning*, 17584–17600. PMLR.

Pochet, Y.; and Wolsey, L. A. 2010. Production Planning by Mixed Integer Programming.

Prouvost, A.; Dumouchelle, J.; Scavuzzo, L.; Gasse, M.; Chételat, D.; and Lodi, A. 2020. Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*.

Rechenberg, I. 1973. Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.

Rosenblatt, F. 1963. PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS. *American Journal of Psychology*, 76: 705.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088): 533–536.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; and Sutskever, I. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. arXiv:1703.03864.

Sutton, R.; and Barto, A. 1998. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 9(5): 1054–1054.

Tang, Y.; Agrawal, S.; and Faenza, Y. 2020. Reinforcement Learning for Integer Programming: Learning to Cut. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 9367–9376. PMLR.

Turner, M.; Koch, T.; Serrano, F.; and Winkler, M. 2022. Adaptive cut selection in mixed-integer linear programming. *arXiv preprint arXiv:2202.10962*.

Wang, Z.; Li, X.; Wang, J.; Kuang, Y.; Yuan, M.; Zeng, J.; Zhang, Y.; and Wu, F. 2023a. Learning Cut Selection for Mixed-Integer Linear Programming via Hierarchical Sequence Model. In *The Eleventh International Conference on Learning Representations*.

Wang, Z.; Pan, T.; Zhou, Q.; and Wang, J. 2023b. Efficient Exploration in Resource-Restricted Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8): 10279–10287.

Wang, Z.; Wang, J.; Zhou, Q.; Li, B.; and Li, H. 2022. Sample-Efficient Reinforcement Learning via Conservative Model-Based Actor-Critic. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8): 8612–8620.

Wesselmann, F.; and Stuhl, U. 2012. Implementing cutting plane management and selection techniques. *University of Paderborn, Tech. Rep*.

Yang, R.; Wang, J.; Geng, Z.; Ye, M.; Ji, S.; Li, B.; and Wu, F. 2022. Learning Task-Relevant Representations for Generalization via Characteristic Functions of Reward Sequence Distributions. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, 2242–2252. New York, NY, USA: Association for Computing Machinery. ISBN 9781450393850.

Zhou, Q.; Kuang, Y.; Qiu, Z.; Li, H.; and Wang, J. 2020. Promoting Stochasticity for Expressive Policies via a Simple and Efficient Regularization Method. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 13504–13514. Curran Associates, Inc.

Zhou, Q.; Li, H.; and Wang, J. 2020. Deep Model-Based Reinforcement Learning via Estimated Uncertainty and Conservative Policy Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34: 6941–6948.