CoCo-MILP: Inter-Variable Contrastive and Intra-Constraint Competitive MILP Solution Prediction

Tianle Pu[1,2*], **Jianing Li**[1,2*], **Yingying Gao**[1*], **Shixuan Liu**[3,1,2], **Zijie Geng**[4], **Haoyang Liu**[4], **Chao Chen**[1,2], **Changjun Fan**[1,2†]

[1]College of Systems Engineering, National University of Defense Technology
[2]Laboratory for Big Data and Decision, National University of Defense Technology
[3]College of Computer Science and Technology, National University of Defense Technology
[4]MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China
{putl22, lijianing24, gaoyingying21, liushixuan, chenc1997, fanchangjun}@nudt.edu.cn,
{zijiegeng, dgyoung}@mail.ustc.edu.cn

Abstract

Mixed-Integer Linear Programming (MILP) is a cornerstone of combinatorial optimization, yet solving large-scale instances remains a significant computational challenge. Recently, Graph Neural Networks (GNNs) have shown promise in accelerating MILP solvers by predicting high-quality solutions. However, we identify that existing methods misalign with the intrinsic structure of MILP problems at two levels. At the learning objective level, the Binary Cross-Entropy (BCE) loss treats variables independently, neglecting their relative priority and yielding plausible logits. At the model architecture level, standard GNN message passing inherently smooths the representations across variables, missing the natural competitive relationships within constraints. To address these challenges, we propose **CoCo-MILP**, which explicitly models inter-variable **Co**ntrast and intra-constraint **Co**mpetition for advanced MILP solution prediction. At the objective level, CoCo-MILP introduces the Inter-Variable Contrastive Loss (VCL), which explicitly maximizes the embedding margin between variables assigned one versus zero. At the architectural level, we design an Intra-Constraint Competitive GNN layer that, instead of homogenizing features, learns to differentiate representations of competing variables within a constraint, capturing their exclusionary nature. Experimental results on standard benchmarks demonstrate that CoCo-MILP significantly outperforms existing learning-based approaches, reducing the solution gap by up to 68.12% compared to traditional solvers.

1 Introduction

Mixed-Integer Linear Programming (MILP) is a cornerstone of combinatorial optimization, with diverse applications in the world (Fan et al. 2020, 2023; Liu et al. 2024b; Pu et al. 2024a,b; Liu et al. 2025c; Wang et al. 2025, 2024). Despite its expressive power, solving MILP instances is fundamentally challenging due to their NP-hard nature. To tackle this, extensive research has led to the development of sophisticated solvers like SCIP (Achterberg 2009) and Gurobi (Gurobi Optimization 2021), which are primarily based on Branch-and-Bound (B&B) and Branch-and-Cut (B&C) algorithms (Land and Doig 2010; Mitchell 2002). Although these solvers are meticulously enhanced with various heuristics, the significant computational burden for large-scale instances remains, motivating the search for new paradigms to accelerate the discovery of high-quality solutions.

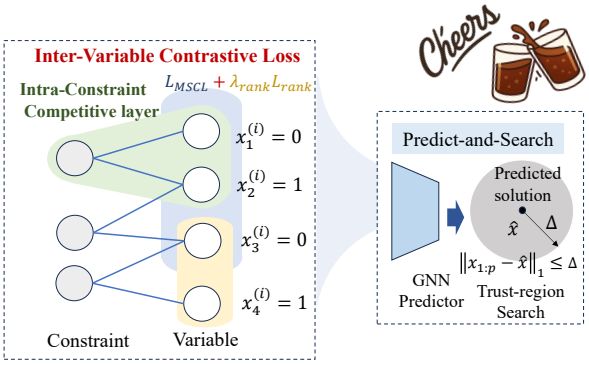

Figure 1: Illustration of CoCo-MILP. We aim to improve the quality of the predicted variables for MILP. The main contribution of our work is the proposed inter-variable contrastive loss and intra-constraint competitive layer for the GNN predictor in the predict-and-search framework.

A promising paradigm has recently emerged to directly predict high-quality primal solutions using machine learning, particularly Graph Neural Networks (GNNs) (Gasse et al. 2019; Wang et al. 2023). These methods typically frame the task in a supervised learning context: by representing MILP instances as graphs, a GNN is trained to map problem structures to variable assignments, using near-optimal solutions generated by traditional solvers as training labels (Han et al. 2023; Huang et al. 2024; Liu et al. 2025b). This data-driven approach allows the model to learn and exploit structural patterns from past instances to make rapid predictions. Pioneering works, such as Neural Diving (Nair et al. 2020a) and the broader predict-and-search strategies (Han

*These authors contributed equally.
†Corresponding author.

et al. 2023), have shown that leveraging these predictions can significantly accelerate the solving process.

However, despite this considerable potential, the efficacy of these learning-based heuristics is fundamentally limited by their underlying design choices. We identify that the standard GNN architectures and learning objectives are not fully aligned with the combinatorial nature of MILP, which manifests as two critical misalignments. First, at the learning objective level, the prevalent use of the Binary Cross-Entropy (BCE) loss function frames the task as a set of independent binary classifications, one for each variable. This formulation inherently neglects the relative priority among variables, which is critical for constructing a globally optimal solution, and consequently yields plausible but poorly separated logits. Second, at the model architecture level, standard GNN message-passing mechanisms inherently smooth representations, making connected nodes more similar. In the context of MILP, however, variables sharing a constraint (e.g., in $\sum_i x_i \leq 1$) are often mutually competitive (Chen et al. 2022, 2024a,b; Zhang et al. 2024). The smoothing nature of standard GNNs thus works directly against the goal of capturing these crucial competitive relationships, effectively masking the problem's underlying structure.

To address these challenges, we propose CoCo-MILP, a novel framework designed to explicitly model inter-variable Contrast and intra-constraint Competition. To rectify the objective-level misalignment, CoCo-MILP introduces an Inter-Variable Contrastive Loss (VCL), which moves beyond pointwise accuracy to directly optimize the relative ordering of variables by maximizing the embedding margin between variables assigned one versus zero. In parallel, to address the architectural misalignment, we design an Intra-Constraint Competitive GNN layer. This layer is engineered to differentiate, rather than homogenize, the representations of competing variables within a constraint, enabling the model to effectively learn their exclusionary nature.

2 Preliminaries

2.1 Mixed-Integer Linear Programming

A standard Mixed-Integer Linear Programming (MILP) instance $\mathcal{I}$ is defined as:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^p\times\mathbb{R}^{n-p}} \left\{\boldsymbol{c}^\top \boldsymbol{x} \;\middle|\; \boldsymbol{A}\boldsymbol{x} \leq \mathbf{b},\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\right\}, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denotes the decision variables, with the first p entries being integer and the remaining $n-p$ continuous. The vector $\boldsymbol{c} \in \mathbb{R}^n$ denotes the coefficients of the linear objective, the constraints are defined by the matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and the right-hand side vector $\boldsymbol{b} \in \mathbb{R}^m$, and the variable bounds are given by $\boldsymbol{l} \in (\mathbb{R} \cup \{-\infty\})^n$ and $\boldsymbol{u} = (\mathbb{R} \cup \{+\infty\})^n$. Without loss of generality, we focus on binary integer variables, i.e., we assume that $\boldsymbol{x} \in \{0,1\}^p \times \mathbb{R}^{n-p}$. General integer variables can be handled via standard preprocessing techniques (Nair et al. 2020b). Finding the optimal solution that optimizes the objective function is NP-hard, making large-scale instances computationally challenging for exact solvers.

2.2 Related Work

In recent years, machine learning techniques have seen widespread use in accelerating the solution of MILPs (Li et al. 2024). These research efforts primarily follow four key directions. One focus is on generating new data to support solver advancement (Geng et al. 2023; Liu et al. 2024a). Another critical direction involves enhancing key modules within solvers including variable selection (Gasse et al. 2019; Qu et al. 2022; Kuang et al. 2024), node selection (He, Daume III, and Eisner 2014; Labassi, Chetelat, and Lodi 2022; Zhang et al. 2025), cutting plane selection (Wang et al. 2023; Li et al. 2023; Ye et al. 2025), and large neighborhood search (Sonnerat et al. 2021; Huang et al. 2023; Song et al. 2020; Wu et al. 2021; Ye et al. 2023) A third area of exploration centers on predicting high-quality solutions to enable warm-starting of solvers, as demonstrated in studies such as (Nair et al. 2020b; Han et al. 2023; Huang et al. 2024; Geng et al. 2025; Liu et al. 2025b). Additionally, efforts have been directed at improving the generalization ability of learning-based models (Liu et al. 2023; Ye et al. 2023; Ye, Xu, and Wang 2024; Pu et al. 2025). We also observe that as Large Language Models (LLMs) grow in popularity, a range of LLM-based methods have appeared, and one notable application among these is LLM-driven optimization modeling (Jiang et al. 2024; Liu et al. 2025a). Finally, we provide details on related works in ML4CO and contrastive learning, which are included in Appendix B.

2.3 Predict-and-Search Framework for MILPs

We can encode each MILP instance as a bipartite graph $\mathcal{G} = (\mathcal{W} \cup \mathcal{V}, \mathcal{E})$, where $\mathcal{W}$ and $\mathcal{V}$ denote the sets of constraint and variable nodes, respectively, and the edge set $\mathcal{E}$ corresponds to non-zero entries in $\boldsymbol{A}$. Each node and edge is associated with a set of features derived from problem coefficients and structural attributes. Such a bipartite graph can completely describe a MILP instance, enabling GNNs to process the instances and predict their solutions.

We adopt the PS paradigm to approximate the solution distribution of a given MILP. Specifically, the distribution is defined via an energy function that assigns lower energy to high-quality feasible solutions and infinite energy to infeasible ones:

$$\begin{aligned} p(\boldsymbol{x} \mid \mathcal{I}) &= \frac{\exp(-E(\boldsymbol{x} \mid \mathcal{I}))}{\sum_{\boldsymbol{x}'} \exp(-E(\boldsymbol{x}' \mid \mathcal{I}))}, \\ \text{where} \quad E(\boldsymbol{x} \mid \mathcal{I}) &= \begin{cases} \boldsymbol{c}^\top \boldsymbol{x}, & \text{if } \boldsymbol{x} \text{ is feasible,} \\ +\infty, & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

Our goal is to learning distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{I})$ for a given instance $\mathcal{I}$. To make learning tractable, we assume a fully factorized solution distribution over the binary variables, i.e., $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{I}) = \prod_{i=1}^{p} p_{\boldsymbol{\theta}}(x_i|\mathcal{I})$, where $p_{\boldsymbol{\theta}}(x_i|\mathcal{I})$ denotes the predicted marginal probability for variable x_i. To do so, we use a GNN model to output a p-dimension vector $\hat{\boldsymbol{x}} = \boldsymbol{f}_{\boldsymbol{\theta}}(\mathcal{I}) = (\hat{x}_1, \cdots, \hat{x}_p)^\top \in [0,1]^p$, where $\hat{x}_j = p_{\boldsymbol{\theta}}(x_j = 1|\mathcal{I})$. To train the model, we use a weighted set of feasible solutions $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ as supervised signals, where each solution is assigned a weight $w_i \propto \exp(-\boldsymbol{c}^\top \boldsymbol{x}^{(i)})$. Let

$x_j^{(i)} = p_{\boldsymbol{\theta}}(x_j^{(i)} = 1|\mathcal{I})$. Then, the training loss function is a binary cross-entropy loss defined as:

$$\mathcal{L}_{\text{BCE}}(\boldsymbol{\theta}|\mathcal{I}) = \sum_{i=1}^{N} w_i \cdot \mathcal{L}_{\text{BCE}}(\boldsymbol{\theta}|\mathcal{I}, \boldsymbol{x}^{(i)})$$
$$= \sum_{i=1}^{N} w_i \cdot \sum_{j=1}^{p} -\left[x_j^{(i)} \log \hat{x}_j + (1 - x_j^{(i)}) \log(1 - \hat{x}_j)\right]. \tag{3}$$

At inference time, the GNN model outputs a predicted marginal $\hat{\boldsymbol{x}} \in [0,1]^p$. A standard MILP solver (e.g., Gurobi or SCIP) is then used to search for a feasible solution in a local neighborhood around $\hat{\boldsymbol{x}}$ by solving the following trust region problem:

$$\min_{\boldsymbol{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} \;\middle|\; \boldsymbol{A}\boldsymbol{x} \leq \mathbf{b},\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \boldsymbol{x}_{1:p} \in \mathcal{B}(\hat{\boldsymbol{x}}, \Delta) \right\}, \tag{4}$$

where the trust region $\mathcal{B}(\hat{\boldsymbol{x}}, \Delta) := \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}_{1:p} - \hat{\boldsymbol{x}}\|_1 \leq \Delta\}$ constrains the solver to remain close to the predicted binary configuration.

3 Motivation

In this section, we motivate our approach by highlighting two fundamental limitations of conventional methods for predicting MILP solutions: (1) their tendency to produce ambiguous logit scores for variables, and (2) their failure to capture the competitive dynamics within constraints. We also provide an extensive motivation analysis from both operations research and application perspectives in Appendix A.

3.1 Inter-Variable Logits Ambiguity

A primary goal of a prediction model is to provide a clear signal to a solver about which variables to prioritize. However, we observe that standard models often fail to do so.

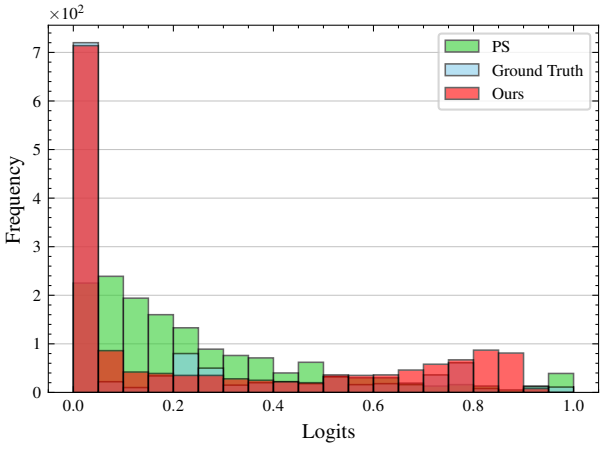

Figure 2: Distribution of Logits from different methods.

Logits Distribution Visualization. We begin by visualizing the logit distributions from a baseline GNN model trained with the standard Binary Cross-Entropy (BCE) loss. As shown in Figure 2, the logits produced by PS are poorly separated. The model assigns similar scores to many variables that should be distinguished, failing to establish a clear decision boundary for reliable variable ranking. In contrast, the logits produced by our method exhibit a much clearer separation, aligning better with the ground truth.

Quantifying Logit Separability. To quantify this ambiguity, we measure the model's ability to rank variables correctly. For a given instance, we sample a large number of positive-negative variable pairs (v_i, v_j) from a ground truth solution, where $x_i = 1$ and $x_j = 0$. We then compute the difference between the predicted scores, $\Delta_{ij} = \hat{x}_i - \hat{x}_j$, where $\hat{x}_i$ and $\hat{x}_j$ are the predictions for variables v_i and v_j, respectively. A positive difference signifies a correct relative ranking, while a negative one indicates an error. The statistics of these differences in Figure 3 reveal that for the baseline model, a substantial portion of the distribution falls into the negative region. This quantitatively confirms that the model is frequently uncertain or even incorrect about the relative importance of variable pairs, underscoring the need for better logit differentiation.

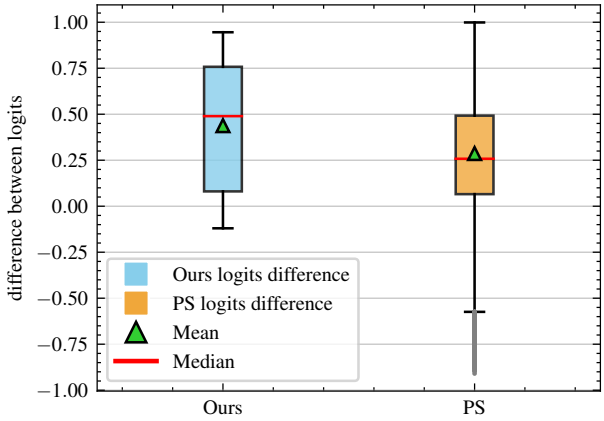

Figure 3: Distribution of the difference between logits corresponding to ground truth 1 and 0 from different methods.

Analysis. The standard training objective for MILP solution prediction is the Binary Cross-Entropy (BCE) loss. This objective encourages the model to independently match the marginal probability for each variable. However, a high-quality solution for a MILP problem depends not on individual accuracies, but on the correct *relative ordering* of variables. A solver does not need to know the absolute probability of $x_i = 1$; it needs to know whether x_i is a *better* candidate than x_j. A model trained with BCE can be "correct on average" yet fail to create a decisive separation between high- and low-priority variables. This results in ambiguous logits that offer a weak and indecisive signal to the solver, motivating our design of a new loss function that explicitly prioritizes comparative ranking.

3.2 Neglect of Intra-Constraint Competitions

A second critical limitation of existing methods is their neglect of the competitive relationships inherent in constraints.

Prevalence of Competition in Constraints. Many constraints, such as the set-packing constraint $\sum_{i \in S} x_i \leq 1$, enforce mutual exclusivity, where only one or a few variables can be simultaneously active (i.e., set to 1). To demonstrate the prevalence of this structure, we analyze the activation ratio of constraints across standard benchmark datasets, defined as the proportion of active variables within a constraint in a ground-truth solution. Figure 4 plots the distribution of these activation ratios. The plot reveals that for the vast majority of constraints, the activation ratio is extremely low. This confirms that a "winner-takes-few" or even "winner-takes-one" dynamic is a fundamental and widespread property of MILP instances.

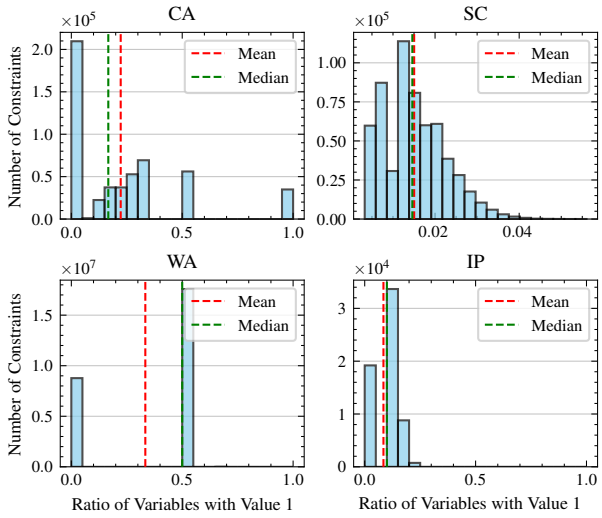

Figure 4: Ratio of variables with value 1 from different problems. For each problem category, we traverse through all constraints to count the number of variables taking the value 1 in the ground truth and plot the resulting histogram.

Insufficient Differentiation within Constraints. We next investigate whether baseline models can capture this competitive structure. An effective model should differentiate between variables within a constraint, assigning high scores to the few "winners" and low scores to the many "losers." We measure this differentiation by calculating the variance of logits among all variables sharing a constraint. Figure 5 compares the distribution of this intra-constraint logit variance for the baseline model against our method. The baseline model's variance is predominantly low, indicating that it assigns indiscriminately similar scores to competing variables. In contrast, our method produces significantly higher variance, demonstrating a superior ability to distinguish between winners and losers within a constraint.

Analysis. This failure stems from the core mechanism of the GNNs commonly used for prediction. Standard GNN message passing is designed to smooth and aggregate features among neighboring nodes. While effective for tasks like node classification, this mechanism is fundamentally ill-suited for modeling competition. By averaging information from neighbors within a constraint, GNNs tend to homogenize the representations of competing variables. This process inherently masks their exclusionary relationships, effectively treating them as collaborators rather than the competitors they are.

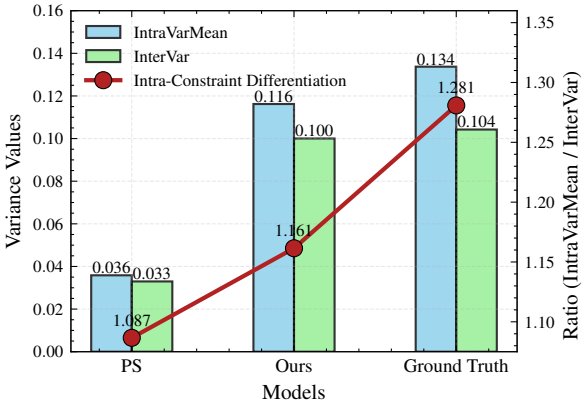

Figure 5: Distribution of logit variance per constraint. IntraVarMean first calculates the variance of logits within each constraint and then computes their average; InterVar calculates the variance across all logits; Ratio represents the ratio of IntraVarMean to InterVar.

4 Methodology

Motivated by the observations in Section 3, we propose **CoCo-MILP**, a novel framework that explicitly models inter-variable **Co**ntrast and intra-constraint **Co**mpetition for advanced MILP solution prediction.

4.1 Framework Overview

The CoCo-MILP framework is built upon the PS framework, as introduced in Section 2.3. In the framework, a MILP instance is represented by a variable-constraint bipartite graph, and a GNN is trained to predict the values of binary variables. Our analysis in Section 3 revealed two critical weaknesses in the standard PS approach: the ambiguous variable logits and honogenized representations. We attribute these issues to the inadequate learning objective and the GNN architecture neglecting variable competitions.

CoCo-MILP addresses the aforementioned issues with two core innovations. In Section 4.2, we introduce the Inter-Variable Contrastive Loss (VCL), which replaces the standard BCE loss to learn more discriminative variable logits by focusing on their relative contrast. Then, in Section 4.3, we introduce the Intra-Constraint Competitive (ICC) GNN layer. This layer explicitly models the competitive relationships among variables within shared constraints, thereby counteracting the feature-smoothing effect of standard message passing.

4.2 Inter-Variable Contrastive Loss (VCL)

The conventional Binary Cross-Entropy (BCE) loss function treats the prediction for each variable as an independent classification task. This approach disregards the crucial relational context between variables, leading to the ambiguous logits demonstrated in Section 3. To address this, we propose the Inter-Variable Contrastive Loss (VCL), which fundamentally shifts the learning objective from pointwise accuracy to learning discriminative logits. The goal of VCL is to structure the embedding space such that the logits of variables with a ground-truth value of one are systematically and confidently larger than those with a ground-truth value of zero.

Our VCL is a composite objective composed of two complementary loss functions: a global multi-sample contrastive loss and a pairwise ranking loss. For a given MILP instance $\mathcal{I}$, let our model with parameters $\boldsymbol{\theta}$ produce a vector of logits $\boldsymbol{z}$ and the corresponding prediction $\hat{\boldsymbol{x}}$. For a ground-truth solution $\boldsymbol{x}^{(i)}$, we partition the set of binary variables $\mathcal{V}_0$ into a positive set $\mathcal{V}_+ = \{v_i \mid x_i^{(i)} = 1\}$ and a negative set $\mathcal{V}_- = \{v_j \mid x_j^{(i)} = 0\}$.

First, we introduce the Multi-Sample Contrastive Loss (MSCL), which is inspired by InfoNCE, which adopts a global perspective. It encourages the logits of all positive samples to be collectively larger than the logits of all other variables in the instance. This is formulated as:

$$\mathcal{L}_{\text{MSCL}}(\boldsymbol{\theta} \mid \mathcal{I}, \mathbf{x}^{(i)}) = -\log \frac{\sum_{v_i \in \mathcal{V}_+} \exp(z_i/\tau)}{\sum_{v_k \in \mathcal{V}_0} \exp(z_k/\tau)}, \tag{5}$$

where τ is a temperature hyperparameter. This loss pushes the entire group of positive variables away from the entire pool of variables, promoting a coarse-grained separation.

Second, to enforce a more fine-grained separation, we employ a Pairwise Ranking Loss. This loss focuses on the local relationship between every individual positive-negative pair, ensuring a strict margin between their logits. It is defined as:

$$\begin{aligned}&\mathcal{L}_{\text{rank}}(\boldsymbol{\theta} \mid \mathcal{I}, \mathbf{x}^{(i)}) \\ =&\frac{1}{|\mathcal{V}_+||\mathcal{V}_-|} \sum_{v_i \in \mathcal{V}_+, v_j \in \mathcal{V}_-} \max(0, \gamma - (z_i - z_j)),\end{aligned} \tag{6}$$

where $\gamma > 0$ is a predefined margin. This loss penalizes any pair (v_i, v_j) where the logit of the positive variable z_i does not exceed the logit of the negative variable z_j by at least γ.

The final Inter-Variable Contrastive Loss (VCL) is the weighted sum of these two components:

$$\begin{aligned}&\mathcal{L}_{\text{VCL}}(\boldsymbol{\theta} \mid \mathcal{I}) = \sum_{i=1}^{N} w_i \cdot \mathcal{L}_{\text{VCL}}(\boldsymbol{\theta} \mid \mathcal{I}, \mathbf{x}^{(i)}) \\ =&\sum_{i=1}^{N} w_i \cdot \left[\mathcal{L}_{\text{MSCL}}(\boldsymbol{\theta} \mid \mathcal{I}, \mathbf{x}^{(i)}) + \lambda_{\text{rank}} \cdot \mathcal{L}_{\text{rank}}(\boldsymbol{\theta} \mid \mathcal{I}, \mathbf{x}^{(i)})\right],\end{aligned} \tag{7}$$

where w_i are weights for each solution and λ_{rank} is a coefficient balancing the two objectives. Unlike BCE, which evaluates each variable in isolation, VCL is relational; its gradient for any single variable depends on the logits of all other variables. By combining a global contrastive push with fine-grained pairwise ranking, VCL produces a much clearer and more robust ranking signal, which is better aligned with the needs of downstream heuristics and solvers.

4.3 Intra-Constraint Competitive GNN

Standard GNN message passing tends to smooth features across connected nodes, an effect that is counterproductive in the MILP context. As discussed in Section 3, variables sharing a constraint are not collaborators but competitors. An effective GNN architecture should therefore not homogenize its representations but instead *differentiate* them to highlight the most promising candidates for inclusion in a solution.

To achieve this, we propose that a variable's representation should be contextualized by its local competition. Instead of learning an absolute embedding, we aim to learn how each variable's features *deviate* from the average features of its direct competitors within each constraint. This relative representation is inherently more discriminative.

We implement this principle with our Intra-Constraint Competitive (ICC) layer. The ICC layer achieves differentiation through a simple yet effective normalization process that follows each standard GNN message-passing update. Let $\boldsymbol{h}_j^{(l)}$ be the embedding of a variable v_j after the l^{th} message-passing layer. The ICC mechanism proceeds in three steps.

Aggregate Competitor Features. For each constraint c_k, we compute an aggregated message by averaging the embeddings of all variables participating in it:

$$\overline{\boldsymbol{h}}_k^{(l)} \leftarrow \frac{1}{|\mathcal{N}(c_k)|} \sum_{v_j \in \mathcal{N}(c_k)} \boldsymbol{h}_j^{(l)}, \tag{8}$$

where $\mathcal{N}(c_k)$ denotes the set of variable nodes connected to constraint c_k. The vector $\overline{\boldsymbol{h}}_k^{(l)}$ represents the "average competitor" within that constraint's neighborhood.

Propagate Competitive Context. The aggregated information is propagated back to the variables. Each variable v_j aggregates the "average competitor" representations from all constraints it participates in:

$$\overline{\boldsymbol{h}}_j^{(l)} \leftarrow \frac{1}{|\mathcal{N}(v_j)|} \sum_{c_k \in \mathcal{N}(v_j)} \overline{\boldsymbol{h}}_k^{(l)}, \tag{9}$$

where $\mathcal{N}(v_j)$ is the set of constraint nodes connected to variable v_j. The resulting vector $\overline{\boldsymbol{h}}_j$ represents the aggregated features of the average peer group against which variable v_j competes.

Compute Competitive Deviation The final update is then performed by subtracting the peer representation from the variable's embedding, scaled by a learnable parameter β:

$$\boldsymbol{h}_j^{(l)} \leftarrow \boldsymbol{h}_j^{(l)} - \beta \cdot \overline{\boldsymbol{h}}_j^{(l)} \tag{10}$$

This mechanism explicitly calculates the deviation of a variable's feature from its competitive baselines. A variable whose embedding is highly distinct from its peers will

produce a resultant vector with a large magnitude, signaling it as a strong, standout candidate. Conversely, a variable that is feature-wise similar to its competitors will have its representation pushed towards the zero vector, diminishing its salience. This process directly counteracts the feature smoothing of standard GNNs and forces the model to learn embeddings that highlight the "winners" within each local competition, producing representations that are far more differentiated and informative for the final prediction task.

5 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of CoCo-MILP. Our approach achieves significant improvements in solving performance on both the synthetic (Section 5.2) and real-world (Section 5.3) benchmarks, as well as generalization ability (Appendix E.4).

Code — https://github.com/happypu326/CoCo-MILP

5.1 Experimental Setup

Datasets We evaluate our framework on four well-established MILP benchmarks frequently used in ML4CO research: Set Covering (SC), Item Placement (IP), Combinatorial Auctions (CA), and Workload Appointment (WA). These benchmarks cover a wide range of combinatorial structures, constraint types, and objective complexities. For SC and CA, we follow standard instance generation procedures from current works (Gasse et al. 2019; Wang et al. 2023; Han et al. 2023; Huang et al. 2024), while IP and WA are taken from the more challenging tracks of the NeurIPS 2021 ML4CO competition (Gasse et al. 2022). Following the experimental setting in (Han et al. 2023), we use 240 instances for training, 60 for validation, and 100 for testing. Please refer to Appendix D.1 for further benchmark details.

Baselines We mainly evaluate our method through two standard categories. First, we measure the performance gains by integrating the heuristics learned by CoCo-MILP into two typical solvers, Gurobi (Gurobi Optimization 2021) and SCIP (Achterberg 2009). Second, we compare CoCo-MILP against two representative learning-based baselines: Predict-and-Search (PS) (Han et al. 2023) and Contrastive Predict-and-Search (ConPS) (Huang et al. 2024). Both PS and ConPS follow a two-stage paradigm which first predicts partial solutions and then finds better primal solutions based on the Gurobi or SCIP solvers. ConPS extends PS by incorporating contrastive learning to enhance predictive quality. Neural Diving (ND) (Nair et al. 2020a) is another related baseline with a similar structure to PS, but we do not include it in our comparison due to its weaker performance reported in PS. Most importantly, we are aware of recent stronger baselines, notably Apollo-MILP (Liu et al. 2025b), which focuses on improving search efficiency without modifying the training process of PS. Therefore, to demonstrate the compatibility of our method, we also integrate CoCo-MILP into Apollo-MILP as a plug-in component and report the results in Appendix E.5. More experimental details can be found in Appendix D.

Metrics To evaluate solution quality, we compare the best objective values (OBJ) obtained by each method within a 1000-second time limit on each test instance. To approximate the optimal value, we adopt the setting from Han et al. (2023), where a single-threaded Gurobi is executed for 3600 seconds and its best result is recorded as the best-known solution (BKS). We then compute the absolute primal gap as $\text{gap}_{\text{abs}} := |\text{OBJ} - \text{BKS}|$, which quantifies how far a solution is from the BKS. A smaller gapabs reflects higher solution quality under the same time budget.

Training and Inference Following the setup in PS, all methods are trained for 1000 epochs on each dataset, with the best model selected based on validation prediction loss. To enhance traditional solvers, PS introduces additional constraints to confine the solution space within a trust region, which is controlled by several key hyperparameters. However, since our benchmarks are more challenging and complex, the hyperparameter settings reported in the original papers result in poor performance. Therefore, we re-tune the hyperparameters for all baselines on our benchmarks. The final hyperparameters are detailed in Appendix D.3. And the extensive analysis of these parameters is presented in Appendix E.2.

5.2 Main Evaluation

Solving Performance Table 1 presents comprehensive results demonstrating that our method consistently achieves state-of-the-art performance across all benchmarks. On the SC benchmark, our method exactly matches the BKS with a near-zero primal gap, yielding a 93.8% improvement over vanilla Gurobi. For the industrial-scale IP and WA benchmarks, our method achieves perfect optimality, fully matching BKS values. Even on the more complex CA benchmark, our approach narrows the absolute gap by 37.1% compared to Gurobi. In comparison with learning-based methods, our model reduces the primal gap by 78.6% against PS and 63.0% against ConPS, underscoring the effectiveness of our architecture in enhancing traditional solvers.

Primal Gap as a Function of Runtime As shown in Figure 6, our method demonstrates strong convergence across all benchmarks. Although the initial gap reduction appears less sharp, this is primarily due to the scaling of the x-axis. In practice, CoCo-MILP is able to reach high-quality solutions within the first 100 seconds. This behavior is attributed to the more accurate variable selection in our method. The fixed variables predicted by CoCo-MILP provide a better starting point for the solver, enabling more efficient local search. In contrast, baseline methods often make early decisions based on pointwise training losses, which can misidentify key variables and lead to premature convergence toward suboptimal solutions. The consistently better final objective values achieved by CoCo-MILP further support its advantage in both convergence speed and solution quality. In addition, we provide the results obtained with the SCIP solver for comparison, as presented in Appendix E.3.

	CA (BKS 97524.37)		SC (BKS 125.05)		IP (BKS 11.16)		WA (BKS 703.05)	
	Obj ↑	gap_{abs} ↓	Obj ↓	gap_{abs} ↓	Obj ↓	gap_{abs} ↓	Obj ↓	gap_{abs} ↓
Gurobi	97228.93	295.44	125.21	0.16	11.43	0.27	703.47	0.42
PS+Gurobi	97286.29	238.08	125.17	0.12	11.40	0.24	703.47	0.42
ConPS+Gurobi	97315.83	208.54	125.18	0.13	11.36	0.20	703.47	0.42
CoCo-MILP+Gurobi	**97338.64**	**185.73**	**125.06**	**0.01**	**11.26**	**0.10**	**703.14**	**0.09**
Improvement		37.1%		93.8%		63.0%		78.6%

Table 1. Performance comparison on four MILP benchmarks (CA, SC, IP, and WA) under a 1000-second time limit. '↑' indicates higher is better; '↓' indicates lower is better. **Bold** denotes the best result. All improvements are reported as gap_{abs} relative to Gurobi.

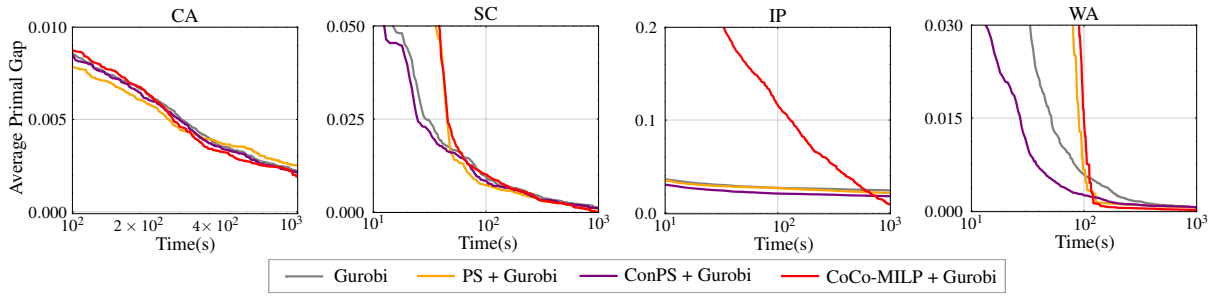

Figure 6: The average primal gap of each method over 100 test instances as a function of solving time. All methods are implemented using Gurobi, with a time limit of 1000 seconds.

	BKS	Gurobi	PS	ConPS	CoCo-MILP
ex1010-pi	233.00	239.00	241.00	239.00	**237.00**
fast0507	174.00	174.00	179.00	179.00	**174.00**
ramos3	186.00	233.00	225.00	225.00	**224.00**
scpj4scip	128.00	132.00	133.00	133.00	**131.00**
scpl4	259.00	277.00	275.00	275.00	**273.00**

Table 2. The best objectives found by the approaches on each test instance in MIPLIB. We obtain the *BKS* from the website of MIPLIB. The ML approaches are implemented using Gurobi with a time limit set to 1000 seconds.

	SC (BKS 125.05)↓	CA (BKS 97524.37)↑
CoCo-MILP w.i. $\mathcal{L}_{\text{BCE}}$	125.25	97240.00
CoCo-MILP w/o $\mathcal{L}_{\text{rank}}$	125.21	97272.41
CoCo-MILP w/o $\mathcal{L}_{\text{MSCL}}$	125.19	97217.40
CoCo-MILP w/o $\mathcal{L}_{\text{ICC layer}}$	125.18	97315.96
CoCo-MILP	**125.06**	**97338.64**

Table 3. Ablation studies. The ML approaches are implemented using Gurobi with a time limit set to 1000 seconds. '↑' indicates that higher is better, and '↓' indicates that lower is better. We mark the **best values** in bold.

5.3 Evaluation on Real-world Benchmark

To further evaluate CoCo-MILP's applicability, we conduct experiments on the MIPLIB dataset (Gleixner et al. 2021). Following (Wang et al. 2023; Liu et al. 2025b), we focus on the IIS subset of MIPLIB, which contains six training instances and five testing instances. The complete information about the MIPLIB benchmark, please refer to Appendix D.1. All methods are trained identically, and their performance is reported in Table 2. We also evaluate CoCo-MILP on additional challenging subsets of MIPLIB and report the results in Appendix E.1.

5.4 Ablation Study

To evaluate the contribution of each component in CoCo-MILP, we compare the full model with four ablated variants: (i) Replacing the VCL loss with $\mathcal{L}_{\text{BCE}}$; (ii) Removing the ranking terms; (iii) Removing the MCSL terms; (iv) Removing the ICC layers in the GNN. As shown in Table 3, omitting either the VCL loss or ICC layers significantly degrades performance, confirming that these design elements play a critical role in CoCo-MILP's effectiveness.

6 Conclusion

In this paper, we identifies key shortcomings in existing learning-based methods for MILP solvers and introduces CoCo-MILP as a solution. Our approach directly addresses these limitations by incorporating explicit mechanisms for inter-variable contrast and intra-constraint competition. Empirical evaluations show that CoCo-MILP surpasses other ML-based baselines in performance, demonstrating not only higher accuracy but also strong potential for practical, real-world deployment.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC, 72571284, 72421002, 62206303, 62273352), the Hunan Provincial Fund for Distinguished Young Scholars (2025JJ20073), the Science and Technology Innovation Program of Hunan Province (2023RC3009), the Foundation Fund Program of National University of Defense Technology (JS24-05), the Major Science and Technology Projects in Changsha, China (kq2301008), and the Hunan Provincial Innovation Foundation for Postgraduate (XJQY2025044).

References

Achterberg, T. 2009. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1: 1–41.

Chen, Z.; Chen, X.; Liu, J.; Wang, X.; and Yin, W. 2024a. Expressive power of graph neural networks for (mixed-integer) quadratic programs. *arXiv preprint arXiv:2406.05938*.

Chen, Z.; Liu, J.; Chen, X.; Wang, W.; and Yin, W. 2024b. Rethinking the capacity of graph neural networks for branching strategy. *Advances in Neural Information Processing Systems*, 37: 123991–124024.

Chen, Z.; Liu, J.; Wang, X.; Lu, J.; and Yin, W. 2022. On representing linear programs by graph neural networks. *arXiv preprint arXiv:2209.12288*.

Fan, C.; Shen, M.; Nussinov, Z.; Liu, Z.; Sun, Y.; and Liu, Y.-Y. 2023. Searching for spin glass ground states through deep reinforcement learning. *Nature communications*, 14(1): 725.

Fan, C.; Zeng, L.; Sun, Y.; and Liu, Y.-Y. 2020. Finding key players in complex networks through deep reinforcement learning. *Nature machine intelligence*, 2(6): 317–324.

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. *Advances in neural information processing systems*, 32.

Geng, Z.; Li, X.; Wang, J.; Li, X.; Zhang, Y.; and Wu, F. 2023. A Deep Instance Generative Framework for MILP Solvers Under Limited Data Availability. In *Advances in Neural Information Processing Systems*.

Geng, Z.; Wang, J.; Li, X.; Zhu, F.; HAO, J.; Li, B.; and Wu, F. 2025. Differentiable Integer Linear Programming. In *The Thirteenth International Conference on Learning Representations*.

Gleixner, A.; Hendel, G.; Gamrath, G.; Achterberg, T.; Bastubbe, M.; Berthold, T.; Christophel, P.; Jarck, K.; Koch, T.; Linderoth, J.; et al. 2021. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3): 443–490.

Gurobi Optimization, L. 2021. Gurobi optimizer. *URL http://www. gurobi. com*.

Han, Q.; Yang, L.; Chen, Q.; Zhou, X.; Zhang, D.; Wang, A.; Sun, R.; and Luo, X. 2023. A gnn-guided predict-and-search framework for mixed-integer linear programming. In *The Eleventh International Conference on Learning Representations*.

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.

Huang, T.; Ferber, A.; Tian, Y.; Dilkina, B. N.; and Steiner, B. 2023. Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning. In *International Conference on Machine Learning*.

Huang, T.; Ferber, A. M.; Zharmagambetov, A.; Tian, Y.; and Dilkina, B. 2024. Contrastive Predict-and-Search for Mixed Integer Linear Programs. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 19757–19771. PMLR.

Jiang, C.; Shu, X.; Qian, H.; Lu, X.; Zhou, J.; Zhou, A.; and Yu, Y. 2024. LLMOPT: Learning to Define and Solve General Optimization Problems from Scratch. *arXiv preprint arXiv:2410.13213*.

Kuang, Y.; Wang, J.; Liu, H.; Zhu, F.; Li, X.; Zeng, J.; Jianye, H.; Li, B.; and Wu, F. 2024. Rethinking Branching on Exact Combinatorial Optimization Solver: The First Deep Symbolic Discovery Framework. In *The Twelfth International Conference on Learning Representations*.

Labassi, A. G.; Chetelat, D.; and Lodi, A. 2022. Learning to Compare Nodes in Branch and Bound with Graph Neural Networks. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 32000–32010. Curran Associates, Inc.

Land, A. H.; and Doig, A. G. 2010. *An automatic method for solving discrete programming problems*. Springer.

Li, S.; Ouyang, W.; Paulus, M. B.; and Wu, C. 2023. Learning to Configure Separators in Branch-and-Cut. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Li, X.; Zhu, F.; Zhen, H.-L.; Luo, W.; Lu, M.; Huang, Y.; Fan, Z.; Zhou, Z.; Kuang, Y.; Wang, Z.; et al. 2024. Machine learning insides optverse ai solver: Design principles and applications. *arXiv preprint arXiv:2401.05960*.

Liu, H.; Kuang, Y.; Wang, J.; Li, X.; Zhang, Y.; and Wu, F. 2023. Promoting Generalization for Exact Solvers via Adversarial Instance Augmentation. arXiv:2310.14161.

Liu, H.; Wang, J.; Cai, Y.; Han, X.; Kuang, Y.; and HAO, J. 2025a. OptiTree: Hierarchical Thoughts Generation with Tree Search for LLM Optimization Modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Liu, H.; Wang, J.; Geng, Z.; Li, X.; Zong, Y.; Zhu, F.; HAO, J.; and Wu, F. 2025b. Apollo-MILP: An Alternating Prediction-Correction Neural Solving Framework for Mixed-Integer Linear Programming. In *The Thirteenth International Conference on Learning Representations*.

Liu, H.; Wang, J.; Zhang, W.; Geng, Z.; Kuang, Y.; Li, X.; Li, B.; Zhang, Y.; and Wu, F. 2024a. MILP-StuDio: MILP Instance Generation via Block Structure Decomposition. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Liu, S.; Fan, C.; Cheng, K.; Wang, Y.; Cui, P.; Sun, Y.; and Liu, Z. 2024b. Inductive meta-path learning for schema-complex heterogeneous information networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liu, S.; He, Y.; Wang, H.; Yang, W.; Wang, Y.; Cui, P.; and Liu, Z. 2025c. Environment Inference for Learning Generalizable Dynamical Systems. In *Advances in Neural Information Processing Systems*, volume 39.

Mitchell, J. E. 2002. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, 1(1): 65–77.

Nair, V.; Bartunov, S.; Gimeno, F.; Von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. 2020a. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*.

Nair, V.; Bartunov, S.; Gimeno, F.; Von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. 2020b. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*.

Pu, T.; Chen, C.; Zeng, L.; Liu, S.; Sun, R.; and Fan, C. 2024a. Solving Combinatorial Optimization Problem Over Graph Through QUBO Transformation and Deep Reinforcement Learning. In *2024 IEEE International Conference on Data Mining (ICDM)*, 390–399. IEEE.

Pu, T.; Fan, C.; Shen, M.; Lu, Y.; Zeng, L.; Nussinov, Z.; Chen, C.; and Liu, Z. 2024b. Exploratory Combinatorial Optimization Problem Solving via Gauge Transformation. In *2024 IEEE International Conference on Data Mining (ICDM)*, 821–826. IEEE.

Pu, T.; Geng, Z.; Liu, H.; Liu, S.; Wang, J.; Zeng, L.; Chen, C.; and Fan, C. 2025. RoME: Domain-Robust Mixture-of-Experts for MILP Solution Prediction across Domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Qu, Q.; Li, X.; Zhou, Y.; Zeng, J.; Yuan, M.; Wang, J.; Lv, J.; Liu, K.; and Mao, K. 2022. An improved reinforcement learning algorithm for learning to branch. *arXiv preprint arXiv:2201.06213*.

Song, J.; lanka, r.; Yue, Y.; and Dilkina, B. 2020. A General Large Neighborhood Search Framework for Solving Integer Linear Programs. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 20012–20023. Curran Associates, Inc.

Sonnerat, N.; Wang, P.; Ktena, I.; Bartunov, S.; and Nair, V. 2021. Learning a Large Neighborhood Search Algorithm for Mixed Integer Programs. *ArXiv*, abs/2107.10201.

Wang, H.; Liu, H.; Luo, J.; Wang, J.; et al. 2024. Accelerating PDE Data Generation via Differential Operator Action in Solution Space. In *Forty-first International Conference on Machine Learning*.

Wang, H.; Wang, J.; Ma, M.; Shao, H.; and Liu, H. 2025. SymMaP: Improving Computational Efficiency in Linear Solvers through Symbolic Preconditioning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

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

Wu, Y.; Song, W.; Cao, Z.; and Zhang, J. 2021. Learning Large Neighborhood Search Policy for Integer Programming. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 30075–30087. Curran Associates, Inc.

Ye, H.; Xu, H.; and Wang, H. 2024. Light-MILPopt: Solving Large-scale Mixed Integer Linear Programs with Lightweight Optimizer and Small-scale Training Dataset. In *The Twelfth International Conference on Learning Representations*.

Ye, H.; Xu, H.; Wang, H.; Wang, C.; and Jiang, Y. 2023. GNN&GBDT-Guided Fast Optimizing Framework for Large-scale Integer Programming. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 39864–39878. PMLR.

Ye, M.; Wang, J.; Zhu, F.; Wang, Z.; Kuang, Y.; Li, X.; Luo, W.; Hao, J.; and Wu, F. 2025. Dynamic Configuration for Cutting Plane Separators via Reinforcement Learning on Incremental Graph. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zhang, B.; Fan, C.; Liu, S.; Huang, K.; Zhao, X.; Huang, J.; and Liu, Z. 2024. The Expressive Power of Graph Neural Networks: A Survey. *IEEE Transactions on Knowledge & Data Engineering*, (01): 1–20.

Zhang, S.; Zeng, S.; Li, S.; Wu, F.; and Li, X. 2025. Learning to Select Nodes in Branch and Bound with Sufficient Tree Representation. In *The Thirteenth International Conference on Learning Representations*.