CAMBRANCH: CONTRASTIVE LEARNING WITH AUGMENTED MILPS FOR BRANCHING

Jiacheng Lin[1,†*] **Meng Xu**[2†] **Zhihua Xiong**[2] **Huangang Wang**[2‡]

[1]University of Illinois Urbana-Champaign [2]Tsinghua University

ABSTRACT

Recent advancements have introduced machine learning frameworks to enhance the Branch and Bound (B&B) branching policies for solving Mixed Integer Linear Programming (MILP). These methods, primarily relying on imitation learning of Strong Branching, have shown superior performance. However, collecting expert samples for imitation learning, particularly for Strong Branching, is a time-consuming endeavor. To address this challenge, we propose **C**ontrastive Learning with **A**ugmented **M**ILPs for **Branch**ing (CAMBranch), a framework that generates Augmented MILPs (AMILPs) by applying variable shifting to limited expert data from their original MILPs. This approach enables the acquisition of a considerable number of labeled expert samples. CAMBranch leverages both MILPs and AMILPs for imitation learning and employs contrastive learning to enhance the model's ability to capture MILP features, thereby improving the quality of branching decisions. Experimental results demonstrate that CAMBranch, trained with only 10% of the complete dataset, exhibits superior performance. Ablation studies further validate the effectiveness of our method.

1 INTRODUCTION

Mixed Integer Linear Programming (MILP) is a versatile tool for solving combinatorial optimization problems, with applications across various fields (Bao & Wang, 2017; Soylu et al., 2006; Godart et al., 2018; Almeida et al., 2006; Hait & Artigues, 2011). A prominent approach for solving MILPs is the Branch-and-Bound (B&B) algorithm (Land & Doig, 1960). This algorithm adopts a divide-and-conquer approach, iteratively resolving sub-problems and progressively reducing the search space. Within the execution of the algorithm, one pivotal decision comes to the fore: variable selection, also known as branching. Traditionally, variable selection relies heavily on expert-crafted rules rooted in substantial domain knowledge. However, recent developments have seen a shift of focus towards the integration of machine learning based frameworks, aiming to enhance the B&B algorithm by replacing conventional, hand-coded heuristics (Gasse et al., 2019; Zarpellon et al., 2021; Nair et al., 2020; Lin et al., 2022). This transition marks a notable advancement in the field, leveraging the power of machine learning and data-driven approaches to tackle complex problems more effectively. For a comprehensive overview of the notable developments in this emerging field, refer to the survey provided in Bengio et al. (2021).

The performance of the B&B algorithm hinges significantly upon its branching strategy, and sub-optimal branching decisions can exponentially escalate the computational workload. This predicament attracts researchers to explore the integration of machine learning (ML) techniques to enhance branching strategies. Notably, Gasse et al. (2019) have trained branching policy models using imitation learning, specifically targeting Strong Branching (Applegate et al., 1995), a traditional strategy known for generating minimal branching search trees but with extremely low efficiency. By mapping a MILP into a bipartite, these branching policy models leverage Graph Convolution Neural Networks (GCNN) (Kipf & Welling, 2017) to extract variable features and make variable selec-

*This work was completed during the master program at Tsinghua University.
†Equal contributions.
‡Corresponding to hgwang@tsinghua.edu.cn.

tion decisions. This approach has demonstrated superior performance in solving MILPs, marking a significant milestone in the application of machine learning to MILP solving.

Despite making significant progress, a significant challenge arises with the imitation learning paradigm mentioned above. The collection of expert samples for imitation learning requires solving numerous MILP instances using Strong Branching, which is computationally intensive and time-consuming. From our experiments, collecting 100k expert samples for four combinatorial optimization problems (*Easy* level) evaluated in (Gasse et al., 2019), namely the Set Covering Problem (BALAS, 1980), Combinatorial Auction Problem (Leyton-Brown et al., 2000), Capacitated Facility Location Problem (Cornuejols et al., 1991), and Maximum Independent Set Problem (Cire & Augusto, 2015), took a substantial amount of time: 26.65 hours, 12.48 hours, 84.79 hours, and 53.45 hours, respectively. These results underscore the considerable effort and resources required for collecting a sufficient number of expert policy samples even on the *Easy* level instances. Importantly, as the complexity of MILPs scales up, the challenge of collecting an adequate number of samples for imitation learning within a reasonable timeframe becomes increasingly impractical.

To address this issue, we present a novel framework named **C**ontrastive Learning with **A**ugmented **M**ILPs for **Branch**ing (CAMBranch). Our approach begins with the development of a data augmentation technique for MILPs. This technique generates a set of Augmented MILPs (AMILPs) through variable shifting, wherein random shifts are applied to each variable within a MILP to produce a new instance. This augmentation strategy enables the acquisition of a substantial number of labeled expert samples, even when expert data is limited. It eliminates the need for extensive computational efforts associated with solving numerous MILP instances, thereby mitigating the challenges related to expert strategy sample collection. Next, building upon the work of Gasse et al. (2019), we transform a MILP into a bipartite graph. By providing theoretical foundations and proofs, we establish a clear correspondence between an augmented bipartite graph (derived from an AMILP) and its corresponding original bipartite graph. These bipartite graph representations are then fed into Graph Convolutional Neural Networks (GCNNs) to extract essential features and make branching decisions. Finally, we employ contrastive learning between MILPs and corresponding AMILPs to facilitate policy network imitation learning. This choice is motivated by the fact that MILPs and their AMILP counterparts share identical branching decisions, enabling a seamless integration of this learning approach. We evaluate our approach on four classical NP-hard combinatorial optimization problems, following the experimental setup described in Gasse et al. (2019). The experimental results demonstrate the superior performance of our proposed CAMBranch, even if CAMBranch leverages only 10% of the data used in Gasse et al. (2019).

2 Preliminaries

2.1 Mixed Integer Linear Programming (MILP)

The general definition form of a MILP problem instance $\mathrm{MILP} = (\boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{\mathcal{I}})$ is shown below

$$\min_{\boldsymbol{x}} \ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b},\ \boldsymbol{l} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u},\ x_j \in \mathbb{Z},\ \forall j \in \boldsymbol{\mathcal{I}} \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is the constraint coefficient matrix in the constraint, $\boldsymbol{c} \in \mathbb{R}^n$ is the objective function coefficient vector, $\boldsymbol{b} \in \mathbb{R}^m$ represents the constraint right-hand side vector, while $\boldsymbol{l} \in (\mathbb{R} \cup \{-\infty\})^n$ and $\boldsymbol{u} \in (\mathbb{R} \cup \{+\infty\})^n$ represent the lower and upper bound vectors for each variable, respectively. The set $\boldsymbol{\mathcal{I}}$ is an integer set containing the indices of all integer variables.

In the realm of solving MILPs, the B&B algorithm serves as the cornerstone of contemporary optimization solvers. Within the framework of the B&B algorithm, the process involves branching, which entails a systematic division of the feasible solution space into progressively smaller subsets. Simultaneously, bounding occurs, which aims to establish either lower-bound or upper-bound targets for solutions within these subsets. Lower bounds are calculated through linear programming (LP) relaxations, while upper bounds are derived from feasible solutions to MILPs.

2.2 MILP Bipartite Graph Encoding

Following Gasse et al. (2019), we model the MILP corresponding to each node in the B&B tree with a bipartite graph, denoted by $(\mathcal{G}, \mathbf{C}, \mathbf{E}, \mathbf{V})$, where: (1) $\mathcal{G}$ represents the structure of the bipartite

Table 1: An overview of the features for constraints, edges, and variables in the bipartite graph $\mathbf{s}_t = (\mathcal{G}, \mathbf{C}, \mathbf{E}, \mathbf{V})$ following Gasse et al. (2019).

Type	Feature	Description
$\mathbf{C}$	obj_cos_sim	Cosine similarity between constraint coefficients and objective function coefficients.
	bias	Normalized right deviation term using constraint coefficients.
	is_tight	Indicator of tightness in the linear programming (LP) solution.
	dualsol_val	Normalized value of the dual solution.
	age	LP age, which refers to the number of solver iterations performed on the LP relaxation problem without finding a new integer solution.
$\mathbf{E}$	coef	Normalized constraint coefficient for each constraint.
$\mathbf{V}$	type	One-hot encoding representing the type (binary variables, integer variables, implicit integer variables, and continuous variables).
	coef	Normalized objective function coefficients.
	has_lb /_ub	Indicator for the lower/upper bound.
	sol_is_at_lb /_ub	The lower/upper bound is equal to the solution value.
	sol_frac	Fractionality of the solution value.
	basis_status	The state of variables in the simplex base is encoded using one-hot encodin (lower, basic, upper, zero).
	reduced_cost	Normalized reduced cost.
	age	Normalized LP age.
	sol_val	Value of the solution.
	inc_val /avg_inc_val	Value/Average value in the incumbent solutions.

graph, that is, if the variable i exists in the constraint j, then an edge $(i, j) \in \mathcal{E}$ is connected between node i and j within the bipartite graph. $\mathcal{E}$ represents the set of edges within the bipartite graph. (2) $\mathbf{C} \in \mathbb{R}^{|\mathbf{C}| \times d_1}$ stands for the features of the constraint nodes, with $|\mathbf{C}|$ denoting the number of constraint nodes, and d_1 representing the dimension of their features. (3) $\mathbf{V} \in \mathbb{R}^{|\mathbf{V}| \times d_2}$ refers to the features of the variable nodes, with $|\mathbf{V}|$ as the count of variable nodes, and d_2 as the dimension of their features. (4) $\mathbf{E} \in \mathbb{R}^{|\mathbf{C}| \times |\mathbf{V}| \times d_3}$ represents the features of the edges, with d_3 denoting the dimension of edge features. Details regarding these features can be found in Table 1.

Next, the input $\mathbf{s}_t = (\mathcal{G}, \mathbf{C}, \mathbf{E}, \mathbf{V})$ is then fed into GCNN which includes a bipartite graph convolutional layer. The bipartite graph convolution process involves information propagation from variable nodes to constraint nodes, and the constraint node features are updated by combining with variable node features. Similarly, the variable node updates its features by combining with constraint node features. For $\forall i \in \mathcal{C}, j \in \mathcal{V}$, the process of message passing can be represented as

$$\mathbf{c}_i' = \mathbf{f}_{\mathcal{C}}\left(\mathbf{c}_i, \sum_{j}^{(i,j)\in\mathcal{E}} \mathbf{g}_{\mathcal{C}}(\mathbf{c}_i, \mathbf{v}_j, \mathbf{e}_{i,j})\right) \qquad \mathbf{v}_j' = \mathbf{f}_{\mathcal{V}}\left(\mathbf{v}_j, \sum_{i}^{(i,j)\in\mathcal{E}} \mathbf{g}_{\mathcal{V}}(\mathbf{c}_i, \mathbf{v}_i, \mathbf{e}_{i,j})\right) \tag{2}$$

where $\mathbf{f}_{\mathcal{C}}$, $\mathbf{f}_{\mathcal{V}}$, $\mathbf{g}_{\mathcal{C}}$, and $\mathbf{g}_{\mathcal{V}}$ are Multi-Layer Perceptron (MLP) (Orbach, 1962) models with two activation layers that use the ReLU function (Agarap, 2018). After performing message passing (Gilmer et al., 2017), a bipartite graph with the same topology is obtained, where the feature values of variable nodes and constraint nodes have been updated. Subsequently, an MLP layer is used to score the variable nodes, and a masked softmax operation is applied to obtain the probability distribution of each variable being selected. The process mentioned above can be expressed as $\boldsymbol{P} = \mathrm{softmax}(\mathrm{MLP}(\mathbf{v}))$. Here, $\boldsymbol{P}$ is the probability distribution of the output variables. During the training phase, GCNN learns to imitate the Strong Branching strategy. Upon completion of the training process, the model becomes ready for solving MILPs.

3 METHODOLOGY

As previously mentioned, acquiring Strong Branching expert samples for imitation learning poses a non-trivial challenge. In this section, we introduce CAMBranch, a novel approach designed to address this issue. Our first step involves the generation of Augmented MILPs (AMILPs) labeled with

Strong Branching decisions, derived directly from the original MILPs. This augmentation process equips us with multiple expert samples essential for imitation learning, even when confronted with limited expert data. Subsequently, building upon the AMILPs, we proceed to create their augmented bipartite graphs. Finally, since MILPs and corresponding AMILPs share branching decisions, we view them as positive pairs. Leveraging the power of contrastive learning, we train our model to boost performance.

3.1 Augmented Mixed Integer Linear Programming (AMILP)

To obtain AMILPs, we adopt variable shift from $\boldsymbol{x}$ defined in Eq.(1) to $\hat{\boldsymbol{x}}$ using a shift vector $\boldsymbol{s}$, denoted as $\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{s}$, where $\boldsymbol{s}$ is a shift vector. Note that if $x_i \in \mathbb{Z}$, then $s_i \in \mathbb{Z}$; otherwise $s_i \in \mathbb{R}$. Based on this translation, we can derive a MILP from Eq.(1). To bring this model into standard form, we redefine the parameters as follows: $\hat{\boldsymbol{b}} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{b}$, $\hat{\boldsymbol{l}} = \boldsymbol{l} + \boldsymbol{s}$, and $\hat{\boldsymbol{u}} = \boldsymbol{u} + \boldsymbol{s}$. Consequently, the final expression for the AMILP model is represented as:

$$\min_{\hat{\boldsymbol{x}}} \ \boldsymbol{c}^{\mathrm{T}}\hat{\boldsymbol{x}} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{s} \quad \text{s.t. } \boldsymbol{A}\hat{\boldsymbol{x}} \leq \hat{\boldsymbol{b}}, \hat{\boldsymbol{l}} \leq \hat{\boldsymbol{x}} \leq \hat{\boldsymbol{u}}, \ \hat{x}_j \in \mathbb{Z}, \ \forall j \in \mathcal{I} \tag{3}$$

Through this data augmentation technique, a single MILP has the capacity to generate multiple AMILPs. It's worth noting that each MILP, along with its corresponding AMILPs, share identical variable selection decisions of the Strong Branching. We next present a theorem to demonstrate this characteristic. To illustrate this distinctive characteristic, we begin by introducing a lemma that elucidates the relationship between MILPs and AMILPs.

Lemma 3.1. *For MILPs Eq.(1) and their corresponding AMILPs Eq.(3), let the optimal solutions of the LP relaxation be denoted as $\boldsymbol{x}^*$ and $\hat{\boldsymbol{x}}^*$, respectively. A direct correspondence exists between these solutions, demonstrating that $\hat{\boldsymbol{x}}^* = \boldsymbol{x}^* + \boldsymbol{s}$.*

The proof of this lemma is provided in the Appendix. Building upon this lemma, we can initially establish the relationship between the optimal values of a MILP and its corresponding AMILP, denoted as $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^* = \boldsymbol{c}^{\mathrm{T}}\hat{\boldsymbol{x}}^* - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{s}$. This equation signifies the equivalence of their optimal values. With the above information in mind, we proceed to introduce the following theorem.

Theorem 3.1. *Suppose that an AMILP instance is derived by shifting variables from its original MILP. When employing Strong Branching to solve these instances, it becomes evident that both the MILP and AMILP consistently produce identical variable selection decisions at each branching step within B&B.*

Proof. In the context of solving a MILP $\mathcal{M}$ using Strong Branching, the process involves pre-branching all candidate variables at each branching step, resulting in sub-MILPs. Solving the linear programing (LP) relaxations of these sub-MILPs provides the optimal values, which act as potential lower bounds for $\mathcal{M}$. The Strong Branching strategy chooses the candidate variable that offers the most substantial lower bound improvement as the branching variable for that step. Thus, the goal of this proof is to demonstrate that the lower bound increments after each branching step are equal when applying Strong Branching to solve both a MILP and its corresponding AMILP.

Given a MILP's branching variable x_i and its corresponding shifted variable of AMILP $\hat{x}_i = x_i + s_i$, we perform branching operations on both variables. Firstly, we branch on x_i to produce two subproblems for MILP, which are formulated as follows:

$$\arg\min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}, \boldsymbol{l} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}, x_i \leqslant \lfloor x_i^* \rfloor, x_j \in \mathbb{Z}, \forall j \in \boldsymbol{\mathcal{I}} \right\} \tag{4}$$

$$\arg\min_{\boldsymbol{x}} \left\{ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}, \boldsymbol{l} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}, x_i \geqslant \lceil x_i^* \rceil, x_j \in \mathbb{Z}, \forall j \in \boldsymbol{\mathcal{I}} \right\} \tag{5}$$

where x_i^* represents the value of variable x_i in the optimal solution corresponding to the MILP LP relaxation. Likewise, we branch on the shifted variable $\hat{x}_i$, which generates two sub-problems for AMILP, as represented by the following mathematical expressions:

$$\arg\min_{\hat{\boldsymbol{x}}} \{ \boldsymbol{c}^{\mathrm{T}}\hat{\boldsymbol{x}} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{s} | \boldsymbol{A}\hat{\boldsymbol{x}} \leq \hat{\boldsymbol{b}}, \hat{\boldsymbol{l}} \leq \hat{\boldsymbol{x}} \leq \hat{\boldsymbol{u}}, \hat{x}_i \leq \lfloor \hat{x}_i^* \rfloor, \hat{x}_j \in \mathbb{Z}, \forall j \in \mathcal{I} \} \tag{6}$$

$$\arg\min_{\hat{\boldsymbol{x}}} \{ \boldsymbol{c}^{\mathrm{T}}\hat{\boldsymbol{x}} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{s} | \boldsymbol{A}\hat{\boldsymbol{x}} \leq \hat{\boldsymbol{b}}, \hat{\boldsymbol{l}} \leq \hat{\boldsymbol{x}} \leq \hat{\boldsymbol{u}}, \hat{x}_i \geq \lceil \hat{x}_i^* \rceil, \hat{x}_j \in \mathbb{Z}, \forall j \in \mathcal{I} \} \tag{7}$$

where $\hat{x_i}^*$ represents the value of variable $\hat{x_i}$ in the optimal solution corresponding to the AMILP LP relaxation.

According to Lemma 3.1, the LP relaxations of Eq.(4) and Eq.(6) have optimal solutions that can be obtained through variable shifting and these two LP relaxations have equivalent optimal values. Similarly, the optimal values of the LP relaxations of Eq.(5) and Eq.(7) are also equal. Thus, for $\hat{x}_i$ and x_i, the lower bound improvements of the subproblems generated from MILP Eq.(1) and AMILP Eq.(3) are equivalent, demonstrating identical branching decisions. The proof is completed. $\square$

Based on Theorem 3.1, it is evident that the generated AMILPs are equipped with expert decision labels, making them readily suitable for imitation learning.

3.2 Augmented Bipartite Graph

After obtaining the AMILP, the subsequent task involves constructing the augmented bipartite graph using a modeling approach akin to the one introduced by Gasse et al. (2019). To achieve this, we leverage the above relationship between MILP and AMILP to derive the node features for the augmented bipartite graph from the corresponding node features of the original bipartite graph, as outlined in Table 1. For a detailed overview of the relationships between node features in the augmented and original bipartite graphs, please refer to the Appendix.

3.2.1 Constraint Node Features

It is worth noting that the AMILP is derived from a translation transformation of the MILP, resulting in certain invariant features: (1) cosine similarity between constraint coefficients and objective function coefficients; (2) tightness state of the LP relaxation solution within constraints; (3) LP age. Additionally, for the *bias* feature, representing the right-hand term, the transformed feature is $b_i + \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{s}$. To obtain this term, consider the i-th constraint node, which corresponds to the i-th constraint of $\boldsymbol{a_i}^{\mathrm{T}}\boldsymbol{x} \leqslant b_i$, After translation, this constraint can be represented as $\boldsymbol{a}_i\hat{\boldsymbol{x}} \leqslant b_i + \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{s}$, leading to the *bias* feature $b_i + \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{s}$. For *dualsol_val* feature, we consider proposing the following theorem for the explanation.

Theorem 3.2. *For MILPs Eq.(1) and AMILPs Eq.(3), let the optimal solution of the dual problem of the LP relaxations be denoted as $\boldsymbol{y}^*$ and $\hat{\boldsymbol{y}}^*$, respectively. Then, a direct correspondence exists between these solutions, indicating that $\boldsymbol{y}^* = \hat{\boldsymbol{y}}^*$.*

The proof can be found in the Appendix. From Theorem 3.2, we can conclude that the *dualsol_val* feature of the augmented bipartite graph remains unchanged compared to the original bipartite graph. Thus, we have successfully determined the constraint node features for the AMILP's bipartite graph through the preceding analysis.

3.2.2 Edge Features

Given that an AMILP is generated from the original MIP through variable shifting, the coefficients of the constraints remain invariant throughout this transformation. Consequently, the values of the edge features in the bipartite graph, which directly reflect the coefficients connecting variable nodes and constraint nodes, also remain unchanged.

3.2.3 Variable Node Features

Similarly to constraint node features, several variable node features also remain unaltered during the transformation. These include (1) the variable type (i.e., integer or continuous); (2) the coefficients of variables corresponding to the objective function; (3) whether the variable has upper and lower bounds; (4) whether the solution value of a variable is within the bounds; (5) whether the solution value of a variable has a decimal part; (6) the status of the corresponding basic vector; (7) the LP age. For *reduced_cost* features, we consider the following theorem for clarification.

Theorem 3.3. *For MILPs Eq.(1) and their corresponding AMILPs Eq.(3), consider the reduced cost corresponding to LP relaxations for a MILP, denoted as σ_i, and for an AMILP, denoted as $\hat{\sigma}_i$. Then, a direct correspondence exists between these reduced costs, implying that $\sigma_i = \hat{\sigma}_i$.*

The proof is provided in the Appendix. Moreover, the features *sol_val*, *inc_val*, and *avg_inc_val* all exhibit shifts in their values corresponding to the shift vector s. With all the above, we have successfully acquired all the features of the augmented bipartite graph.

3.3 CONTRASTIVE LEARNING

Contrastive learning has been widely adopted in various domains (He et al., 2020; Chen et al., 2020a; P. et al., 2020; Xu et al., 2022; Iter et al., 2020; Giorgi et al., 2021; Yu et al., 2022). The fundamental idea behind contrastive learning is to pull similar data points (positives) closer together in the feature space while pushing dissimilar ones (negatives) apart. Within our proposed CAMBranch, we leverage this principle by viewing a MILP and its corresponding AMILP as positive pairs while considering the MILP and other AMILPs within the same batch as negative pairs. This enables us to harness the power of contrastive learning to enhance our training process.

We initiate the process with a MILP bipartite graph $(\mathcal{G}_{\text{ori}}, \mathbf{C}_{\text{ori}}, \mathbf{E}_{\text{ori}}, \mathbf{V}_{\text{ori}})$ and its augmented counterpart $(\mathcal{G}_{\text{aug}}, \mathbf{C}_{\text{aug}}, \mathbf{E}_{\text{aug}}, \mathbf{V}_{\text{aug}})$. These graphs undergo processing with a GCNN, following the message passing illustrated in Eq.(2). This results in updated constraint and variable node features $\mathbf{C}'_{\text{ori}}$ and $\mathbf{V}'_{\text{ori}}$ for the MILP, along with $\mathbf{C}'_{\text{aug}}$ and $\mathbf{V}'_{\text{aug}}$ for the AMILP. Subsequently, we generate graph-level representations for both bipartite graphs. To achieve this, we conduct max and average pooling on the constraint nodes and variable nodes, respectively. Merging these embeddings using an MLP yields pooled embeddings for constraint and variable nodes, denoted as $\boldsymbol{c}_{\text{ori}}^{\mathcal{G}}, \boldsymbol{v}_{\text{ori}}^{\mathcal{G}}$ for the MILP and $\boldsymbol{c}_{\text{ori}}^{\mathcal{G}}, \boldsymbol{v}_{\text{ori}}^{\mathcal{G}}$ for the AMILP. These embeddings serve as inputs to another MLP, resulting in graph-level embeddings $\boldsymbol{g}_{\text{ori}}$ and $\boldsymbol{g}_{\text{aug}}$ for MILP and AMILP bipartite graphs, respectively. To train our model using contrastive learning, we treat $\boldsymbol{g}_{\text{ori}}$ and its corresponding $\boldsymbol{g}_{\text{aug}}$ as positive pairs, while considering other AMILPs in the same batch as negative samples. By applying infoNCE (van den Oord et al., 2018) loss, we have

$$\mathcal{L}^{(\text{infoNCE})} = -\sum_{i=1}^{n_{\text{batch}}} \log\left(\frac{\exp\left(\tilde{\boldsymbol{g}}_{\text{ori}}^{\text{T}}(i)\cdot\tilde{\boldsymbol{g}}_{\text{aug}}(i)\right)}{\sum_{j=1}^{n_{\text{batch}}}\exp\left(\tilde{\boldsymbol{g}}_{\text{ori}}^{\text{T}}(i)\cdot\tilde{\boldsymbol{g}}_{\text{aug}}(j)\right)}\right) \tag{8}$$

where n_{batch} represents the number of samples in a training batch. $\tilde{\boldsymbol{g}}_{\text{ori}}$ and $\tilde{\boldsymbol{g}}_{\text{aug}}$ denote the normalized vectors of $\boldsymbol{g}_{\text{ori}}$ and $\boldsymbol{g}_{\text{aug}}$, respectively. This contrastive learning approach enhances our model's ability to capture representations for MILPs, which further benefits the imitation learning process. The imitation learning process follows Gasse et al. (2019). More details can be found in the Appendix C.3.

4 EXPERIMENT

We evaluate our proposed CAMBranch by fully following the settings in Gasse et al. (2019). Due to the space limit, we briefly introduce the experimental setup and results. More details are provided in the Appendix D.

4.1 SETUP

Benchmarks. Following Gasse et al. (2019), we assess our method on four NP-hard problems, i.e., Set Covering (BALAS, 1980), Combinatorial Auction (Leyton-Brown et al., 2000), Capacitated Facility Location (Cornuejols et al., 1991), and Maximum Independent Set (Cire & Augusto, 2015). Each problem has three levels of difficulty, that is, *Easy*, *Medium*, and *Hard*. We train and test models on each benchmark separately. Through the experiments, we leverage SCIP 6.0.1 (Gleixner et al., 2018) as the backend solver and set the time limit as 1 hour. See more details in the supplementary materials.

Baselines. We compare CAMBranch with the following branching strategies: (1) Reliability Pseudocost Branching (`RPB`) (Achterberg et al., 2005), a state-of-the-art human-designed branching policy and the default branching rule of the SCIP solver; (2) GCNN (Gasse et al., 2019), a state-of-the-art neural branching policy; (3) GCNN (10%), which uses only 10% of the training data from Gasse et al. (2019). This is done to ensure a complete comparison since CAMBranch also utilizes 10% of the data.

Data Collection and Split. The expert samples for imitation learning are collected from SCIP rollout with Strong Branching on the *Easy* level instances. Following Gasse et al. (2019), we train

GCNN with 100k expert samples, while CAMBranch is trained with 10% of these samples. Trained with *Easy* level samples, the models are tested on all three level instances. Each level contains 20 new instances for evaluation using five different seeds, resulting in 100 solving attempts for each difficulty level.

Metrics. Following Gasse et al. (2019), our metrics are standard for MILP benchmarking, including solving time, number of nodes in the branch and bound search tree, and number of times each method achieves the best solving time among all methods (number of wins). For the first two metrics, smaller values are indicative of better performance, while for the latter, higher values are preferable.

4.2 Experimental Results

To assess the effectiveness of our proposed CAMBranch, we conducted the evaluation from two aspects: imitation learning accuracy and MILP instance-solving performance. The former measures the model's ability to imitate the expert strategy, i.e., the Strong Branching strategy. Meanwhile, MILP instance-solving performance evaluates the quality and efficiency of the policy network's decisions, emphasizing critical metrics such as MILP solving time and the size of the B&B tree (i.e., the number of nodes) generated during the solving process.

4.2.1 Imitation Learning Accuracy

First, we initiated our evaluation by comparing CAMBranch with baseline methods in terms of imitation learning accuracy. Following Gasse et al. (2019), we curated a test set comprising 20k expert samples for each problem. The results are depicted in Figure 1. Notably, CAMBranch, trained with 10% of the full training data, outperforms GCNN (10%), demonstrating that our proposed data augmentation and contrastive learning framework benefit the imitation learning process. Moreover, CAMBranch's performance, while exhibiting a slight lag compared to GCNN (Gasse et al., 2019) trained on the entire dataset, aligns with expectations, considering the substantial difference in the size of the training data. CAMBranch delivers comparable performance across three of the four problems, with the exception being the Set Covering problem.

4.2.2 Instance Solving Evaluation

Next, we sought to evaluate our proposed CAMBranch on MILP instance solving focusing on three key metrics: solving time, the number of B&B nodes, and the number of wins, as illustrated by Table 2.

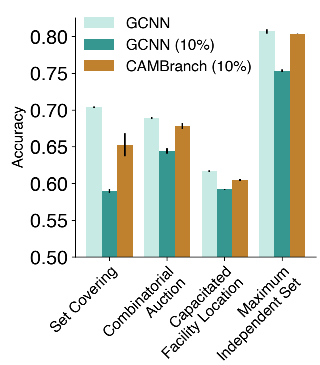

Figure 1: Imitation learning accuracy on the test sets of expert samples.

Solving time reflects the efficiency of each model in solving MILP instances. As evident in Table 2, we observed that as problem complexity increases, the solving time also rises substantially, with an obvious gap between *Easy* and *Hard* levels. In the *Easy* level, all strategies exhibit similar solving time, with only a maximum gap of about 5 seconds. However, for the *Medium* and *Hard* levels, differences become more significant. Notably, delving into each strategy, we found that, neural network-based policies consistently outperform the traditional `RPB`, demonstrating the potential of replacing heuristic methods with machine learning-based approaches. Moreover, CAMBranch exhibits the fastest solving process in most cases, particularly in challenging instances. For example, in the hard-level Capacitated Facility Location problem, CAMBranch achieved a solving time of 470.83 seconds, nearly 200 seconds faster than GCNN. Furthermore, CAMBranch outperforms GCNN (10%) across various instances, reaffirming the effectiveness of our CAMBranch framework.

The number of B&B nodes serves as a measure of branching decision quality, with fewer nodes indicating better decision quality. Table 2 presents similar observations to solving time. CAMBranch consistently outperforms GCNN in most cases, especially in challenging scenarios like the hard-level Maximum Independent Set problem. On average, CAMBranch generates fewer nodes than GCNN (10%), highlighting the efficacy of our data augmentation and contrastive learning network. However, it's worth noting that in some cases, `RPB` generates the fewest nodes, particularly in the Capacitated Facility Location problem. Nevertheless, this doesn't translate into shorter solving times, as certain `RPB` decisions are time-consuming.

The number of wins quantifies the instances in which the model achieves the shortest solving time. Higher win counts indicate better performance. With this metric, we examined models at the instance level. From Table 2, we found that GCNN obtains the most times of getting the fastest solving process in the Set Covering problem and the Combinatorial Auction problem (*Easy* and *Medium*). However, for the remaining problems, CAMBranch leads in this metric. Additionally, CAMBranch tends to optimally solve the highest number of instances, except in the case of Set Covering. These results underline the promise of our proposed method, especially in scenarios with limited training data. Collectively, CAMBranch's prominent performance across these three metrics underscores the importance of MILP augmentation and the effectiveness of our contrastive learning framework.

Table 2: Policy evaluation in terms of solving time, number of B&B nodes, and number of wins over number of solved instances on four combinatorial optimization problems. Each level contains 20 instances for evaluation using five different seeds, resulting in 100 solving attempts for each difficulty level. Bold CAMBranch numbers denote column-best results among neural policies.

	Easy			Medium			Hard		
Model	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
`FSB`	17.98	0/100	27	345.91	0/90	223	3600.00	-	-
`RPB`	7.82	4/100	103	64.77	10/100	2587	1210.18	32 / 63	80599
GCNN	6.03	57/100	167	50.11	81/100	1999	1344.59	36/68	56252
GCNN (10%)	6.34	39/100	230	98.20	5/96	5062	2385.23	0/6	113344
CAMBranch (10%)	6.79	0/100	188	61.00	4/100	2339	1427.02	0/55	66943
Set Covering									
`FSB`	4.71	0/100	10	97.6	0/100	90	1396.62	0/64	381
`RPB`	2.61	1/100	21	19.68	2/100	713	142.52	29/100	8971
GCNN	1.96	43/100	87	11.30	74/100	695	158.81	19/94	12089
GCNN (10%)	1.99	44/100	102	12.38	16/100	787	144.40	10/100	10031
CAMBranch (10%)	2.03	12/100	91	12.68	8/100	758	**131.79**	**42/100**	**9074**
Combinatorial Auction									
`FSB`	34.94	0/100	54	242.51	0/100	114	995.40	0/82	84
`RPB`	30.63	9/100	79	177.25	2/100	196	830.90	2/93	178
GCNN	24.72	25/100	169	145.17	13/100	405	680.78	5/95	449
GCNN (10%)	26.30	15/100	180	124.49	48/100	406	672.88	11/95	423
CAMBranch (10%)	24.91	**50/100**	183	**124.36**	37/100	**390**	**470.83**	**77/95**	428
Capacitated Facility Location									
`FSB`	28.85	10/100	19	1219.15	0/62	81	3600.00	-	-
`RPB`	10.73	11/100	78	133.30	5/100	2917	965.67	10/40	17019
GCNN	7.17	11/100	90	164.51	4/99	5041	1020.58	0/17	21925
GCNN (10%)	7.18	26/100	103	122.65	8/89	3711	695.96	2/20	17034
CAMBranch (10%)	**6.92**	**42/100**	**90**	**61.51**	**83/100**	**1479**	**496.86**	**33/40**	**10828**
Maximum Independent Set									

4.2.3 Evaluation of Data Collection Efficiency

In this part, we compared the efficiency of expert sample collection for GCNN in Gasse et al. (2019) and our proposed CAMBranch. We focused on the Capacitated Facility Location Problem, which exhibits the lowest collection efficiency among the four MILP benchmarks and thus closely simulates real-world MILPs by low data collection efficiency. Generating 100k expert samples using Strong Branching to solve the instances takes 84.79 hours. In contrast, if obtaining the same quantity of expert samples, CAMBranch requires 8.48 hours (collecting 10k samples initially) plus 0.28 hours (generating the remaining 90k samples based on the initial 10k), totaling 8.76 hours—an 89.67% time savings. This underscores the superiority of CAMBranch in data collection efficiency.

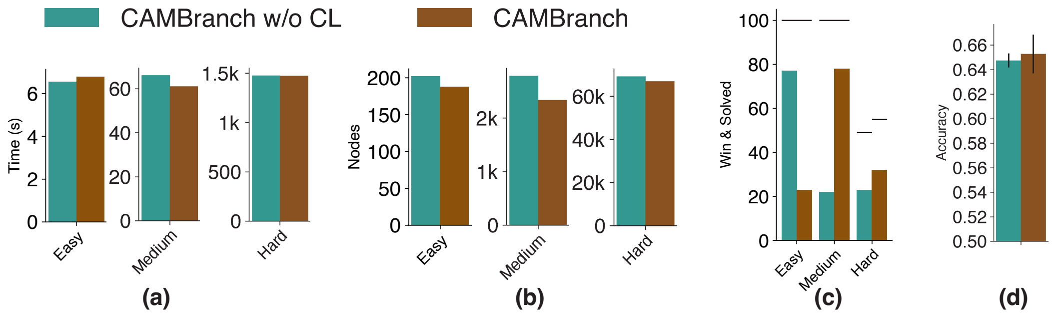

Figure 2: Ablation experiment results on CAMBranch (10%) for instance solving evaluation, including solving time (a), number of nodes (b), and number of wins (c), in addition to imitation learning accuracy (d) for the Set Covering problem.

4.3 Ablation Studies

To further validate the effectiveness of contrastive learning, we conducted ablation studies. Specifically, we compared the performance of CAMBranch with contrastive learning to CAMBranch without contrastive learning but with data augmentation, denoted as CAMBranch w/o CL. These experiments were conducted on the Set Cover problem, and the results are displayed in Figure 2. It is evident from the results that integrating contrastive learning significantly enhances CAMBranch's performance, providing compelling evidence of the efficacy of this integration within CAMBranch.

4.4 Evaluating CAMBranch on Full Datasets

Previous experiments have showcased CAMBranch's superiority in data-scarce scenarios. To further explore CAMBranch's potential, we conducted evaluations on complete datasets to assess its performance with the entire training data. Table 3 presents the results of instance-solving evaluations for the Combinatorial Auction problem. The outcomes reveal that when trained with the full dataset, CAMBranch (100%) surpasses GCNN (10%), CAMBranch (10%) and even GCNN (100%). Notably, CAMBranch exhibits the fastest solving time for nearly 90% of instances, underscoring its effectiveness. For *Hard* instances, CAMBranch (100%) demonstrates significant improvements across all instance-solving evaluation metrics. These findings affirm that our plug-and-play CAMBranch is versatile, excelling not only in data-limited scenarios but also serving as a valuable tool for data augmentation to enhance performance with complete datasets.

Table 3: The results of evaluating the instance-solving performance for the Combinatorial Auction problem by utilizing the complete training dataset. Bold numbers denote the best results.

Model	Easy			Medium			Hard		
	Time	Wins	Nodes	Time	Wins	Nodes	Time	Wins	Nodes
GCNN (10%)	1.99	2/100	102	12.38	3/100	787	144.40	2/100	10031
GCNN (100%)	1.96	4/100	**87**	11.30	7/100	695	158.81	4/94	12089
CAMBranch (10%)	2.03	1/100	91	12.68	2/100	758	131.79	11/100	9074
CAMBranch (100%)	**1.73**	**93/100**	88	**10.04**	**88/100**	**690**	**109.96**	**83/100**	**8260**

5 Conclusion

In this paper, we have introduced CAMBranch, a novel framework designed to address the challenge of collecting expert strategy samples for imitation learning when applying machine learning techniques to solve MILPs. By introducing variable shifting, CAMBranch generates AMILPs from the original MILPs, harnessing the collective power of both to enhance imitation learning. Our utilization of contrastive learning enhances the model's capability to capture MILP features, resulting in more effective branching decisions. We have evaluated our method on four representative combinatorial optimization problems and observed that CAMBranch exhibits superior performance, even when trained on only 10% of the complete dataset. This underscores the potential of CAMBranch, especially in scenarios with limited training data.

ACKNOWLEDGEMENT

This work is supported by National Key R&D Program of China (Grant No.2021YFC2902701).

REFERENCES

Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Oper. Res. Lett.*, 33(1):42–54, 2005. doi: 10.1016/j.orl.2004.04.002. URL https://doi.org/10.1016/j.orl.2004.04.002.

Abien Fred Agarap. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. URL http://arxiv.org/abs/1803.08375.

Renato T. R. Almeida, Luiz C. de Calmon, Elias Olivieira, and Marcelo Eduardo Vieira Segatto. Design of virtual topologies for large optical networks through an efficient MILP formulation. *Opt. Switch. Netw.*, 3(1):2–10, 2006. doi: 10.1016/j.osn.2005.10.002. URL https://doi.org/10.1016/j.osn.2005.10.002.

Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. A machine learning-based approximation of strong branching. *INFORMS J. Comput.*, 29(1):185–195, 2017. doi: 10.1287/ijoc.2016.0723. URL https://doi.org/10.1287/ijoc.2016.0723.

David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. *Finding cuts in the TSP (A preliminary report)*, volume 95. Citeseer, 1995.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

E BALAS. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study. *Mathematical Programming Study*, 12:37–60, 1980.

Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 353–362. PMLR, 2018. URL http://proceedings.mlr.press/v80/balcan18a.html.

Tianpeng Bao and Huangang Wang. Cyclic scheduling of multi-cluster tools based on mixed integer programming. *IEEE Transactions on Semiconductor Manufacturing*, 30(4):515–525, 2017.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.*, 290(2):405–421, 2021. doi: 10.1016/j.ejor.2020.07.063. URL https://doi.org/10.1016/j.ejor.2020.07.063.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/70feb62b69f16e0238f741fab228fec2-Abstract.html.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020a. URL http://proceedings.mlr.press/v119/chen20j.html.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien

Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/fcbc95ccdd551da181207c0c1400c655-Abstract.html.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 15750–15758. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01549. URL https://openaccess.thecvf.com/content/CVPR2021/html/Chen_Exploring_Simple_Siamese_Representation_Learning_CVPR_2021_paper.html.

Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *CoRR*, abs/2003.04297, 2020c. URL https://arxiv.org/abs/2003.04297.

Cire and Andre Augusto. Decision diagrams for optimization. *Constraints*, 20(4):494–495, 2015.

G. Cornuejols, R. Sridharan, and J. M. Thizy. A comparison of heuristics and relaxations for the capacitated plant location problem. *European Journal of Operational Research*, 50(3):280–297, 1991.

Santanu S Dey, Yatharth Dubey, Marco Molinaro, and Prachi Shah. A theoretical and computational analysis of full strong-branching. *Mathematical Programming*, pp. 1–34, 2023.

William Falcon and Kyunghyun Cho. A framework for contrastive self-supervised learning and designing A new approach. *CoRR*, abs/2009.00104, 2020. URL https://arxiv.org/abs/2009.00104.

Hongchao Fang and Pengtao Xie. CERT: contrastive self-supervised learning for language understanding. *CoRR*, abs/2005.12766, 2020. URL https://arxiv.org/abs/2005.12766.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019. URL http://arxiv.org/abs/1903.02428.

Gerald Gamrath, Timo Berthold, and Domenico Salvagnin. An exploratory computational analysis of dual degeneracy in mixed-integer programming. *EURO J. Comput. Optim.*, 8(3):241–261, 2020. doi: 10.1007/s13675-020-00130-z. URL https://doi.org/10.1007/s13675-020-00130-z.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15554–15566, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d14c2267d848abeb81fd590f371d39bd-Abstract.html.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017. URL http://proceedings.mlr.press/v70/gilmer17a.html.

John M. Giorgi, Osvald Nitski, Bo Wang, and Gary D. Bader. Declutr: Deep contrastive learning for unsupervised textual representations. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 879–895. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.72. URL https://doi.org/10.18653/v1/2021.acl-long.72.

Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. Technical report, Optimization Online, July 2018. URL http://www.optimization-online.org/DB_HTML/2018/07/6692.html.

Alexis Godart, Hervé Manier, Christelle Bloch, and Marie-Ange Manier. Milp for a variant of pickup & delivery problem for both passengers and goods transportation. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2692–2698. IEEE, 2018.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

Prateek Gupta, Maxime Gasse, Elias B. Khalil, Pawan Kumar Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d1e946f4e67db4b362ad23818a6fb78a-Abstract.html.

Alain Hait and Christian Artigues. A hybrid cp/milp method for scheduling with energy costs. *European Journal of Industrial Engineering*, 5(4):471–489, 2011.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 9726–9735. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00975. URL https://doi.org/10.1109/CVPR42600.2020.00975.

Dan Iter, Kelvin Guu, Larry Lansing, and Dan Jurafsky. Pretraining with contrastive sentence objectives improves discourse performance of language models. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 4859–4870. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.439. URL https://doi.org/10.18653/v1/2020.acl-main.439.

Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6348–6358, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html.

Elias Boutros Khalil, Pierre Le Bodic, Le Song, George L. Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 724–731. AAAI Press, 2016. doi: 10.1609/aaai.v30i1.10080. URL https://doi.org/10.1609/aaai.v30i1.10080.

Fatma Kilinç-Karzan, George L. Nemhauser, and Martin W. P. Savelsbergh. Information-based branching schemes for binary linear mixed integer problems. *Math. Program. Comput.*, 1(4):

249–293, 2009. doi: 10.1007/s12532-009-0009-1. URL https://doi.org/10.1007/s12532-009-0009-1.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

AH Land and AG Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In Anant Jhingran, Jeff MacKie-Mason, and Doug J. Tygar (eds.), *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC-00), Minneapolis, MN, USA, October 17-20, 2000*, pp. 66–76. ACM, 2000. doi: 10.1145/352871.352879. URL https://doi.org/10.1145/352871.352879.

Jiacheng Lin, Jialin Zhu, Huangang Wang, and Tao Zhang. Learning to branch with tree-aware branching transformers. *Knowl. Based Syst.*, 252:109455, 2022. doi: 10.1016/j.knosys.2022.109455. URL https://doi.org/10.1016/j.knosys.2022.109455.

Michele Monaci Matteo Fischetti. Branching on nonchimerical fractionalities. *Operations Research Letters*, 40(3):159–164, 2012.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 3111–3119, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. Solving mixed integer programs using neural networks. *CoRR*, abs/2012.13349, 2020. URL https://arxiv.org/abs/2012.13349.

J Orbach. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *Archives of General Psychiatry*, 7(3):218–219, 1962.

Khosla P., Teterwak P., and Wang C. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Dean Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Comput.*, 3(1):88–97, 1991. doi: 10.1162/neco.1991.3.1.88. URL https://doi.org/10.1162/neco.1991.3.1.88.

Lara Scavuzzo, Feng Yang Chen, Didier Chételat, Maxime Gasse, Andrea Lodi, Neil Yorke-Smith, and Karen I. Aardal. Learning to branch with tree mdps. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/756d74cd58592849c904421e3b2ec7a4-Abstract-Conference.html.

Ahu Soylu, Cihan Oruç, Metin Turkay, Kaoru Fujita, and Tatsuyuki Asakura. Synergy analysis of collaborative supply chain management in energy systems using multi-period milp. *European Journal of Operational Research*, 174(1):387–403, 2006.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Hanwen Xu, Jiacheng Lin, Addie Woicik, Zixuan Liu, Jianzhu Ma, Sheng Zhang, Hoifung Poon, Liewei Wang, and Sheng Wang. Pisces: A multi-modal data augmentation approach for drug combination synergy prediction. *bioRxiv*, pp. 2022–11, 2022.

Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary?: Simple graph contrastive learning for recommendation. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (eds.), *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pp. 1294–1303. ACM, 2022. doi: 10.1145/3477495.3531937. URL https://doi.org/10.1145/3477495.3531937.

Giulia Zarpellon, Jason Jo, Andrea Lodi, and Yoshua Bengio. Parameterizing branch-and-bound search trees to learn branching policies. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 3931–3939. AAAI Press, 2021. doi: 10.1609/aaai.v35i5.16512. URL https://doi.org/10.1609/aaai.v35i5.16512.

Jiayi Zhang, Chang Liu, Xijun Li, Hui-Ling Zhen, Mingxuan Yuan, Yawen Li, and Junchi Yan. A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519: 205–217, 2023. doi: 10.1016/j.neucom.2022.11.024. URL https://doi.org/10.1016/j.neucom.2022.11.024.