Generative Branching for Mixed-Integer Linear Programming

Ruobing Wang[1], **Xin Li**[1,2*], **Yangchuan Wang**[1], **Zijian Zhang**[1], **Mingzhong Wang**[3]

[1]Beijing Institute of Technology, Beijing, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, China
[3] University of the Sunshine Coast
3120245684@bit.edu.cn, xinli@bit.edu.cn, wyc2828@139.com, zhangzijian@bit.edu.cn, mwang@usc.edu.au

Abstract

Branch-and-bound (B&B) is a fundamental algorithmic framework for solving Mixed-Integer Linear Programming (MILP) problems, where branching decisions critically affect solver efficiency. Recent learning-based methods apply imitation learning to select branching variables, but their deterministic predictions limit exploration and generalization. In this paper, we propose a novel framework that formulates branching variable selection as a conditional generative process, exploring deep-level decision features. Our approach leverages diffusion models to enable diverse and exploratory branching score generation, while consistency modeling distills this process into efficient one-step inference conditioned on the B&B state. This mode allows our method to achieve both high-quality and fast branching decisions, significantly improving the overall performance of branch-and-bound solvers. Extensive experiments on challenging cross-scale and cross-category benchmarks demonstrate that our framework consistently outperforms state-of-the-art imitation learning baselines, delivering substantial improvements in solution quality, computational efficiency, and inference speed.

Introduction

Mixed-Integer Linear Programming (MILP) problems constitute a fundamental class of NP-hard optimization problems that involve both continuous and discrete variables (Cook 1971; Karp 1972), with widespread applications in scheduling (Yao et al. 2023; Zhao et al. 2025; Meng et al. 2023), routing (Reda et al. 2024), and resource allocation (Zelaschi et al. 2025). Branch-and-Bound (B&B) algorithms (Land and Doig 1960) are widely used to solve these problems by recursively partitioning the solution space and making critical decisions on node selection and variable branching at each step (Wolsey 2020; Nemhauser and Wolsey 1988). However, the exponential growth of the search tree renders large-scale instances computationally expensive, often requiring prohibitive solve times for time-sensitive applications (Morrison et al. 2016).

Traditional MILP solvers rely on hand-crafted branching heuristics. As an improvement, strong branching (Achterberg, Koch, and Martin 2005) provides high-quality variable selection by evaluating all candidates, which is efficient for medium-scale problems but intractable for large-scale ones. To address this, simpler heuristics, such as most-fractional branching, which selects variables with the highest branching score, are computationally efficient but may lead to suboptimal decisions. Recent advances in machine learning have enabled learning-to-branch approaches, such as those based on imitation learning, which aim to combine expert-level decision quality with computational efficiency (Bengio, Lodi, and Prouvost 2021; Khalil et al. 2016).

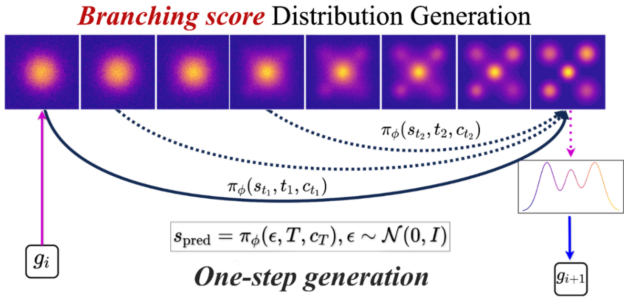

Figure 1: Illustration of Consistency Policy generating branching score distributions in a single step, conditioned on the branching state g_i, for controllable and efficient score generation.

Generally, exact solvers use branching scores of each candidate variable to guide branching decisions, with higher scores indicating greater single-step branching benefits (Bestuzheva et al. 2021). State-of-the-art imitation learning-based methods have demonstrated promising results in reducing the computational overhead of strong branching by predicting a single "best" variable to branch on. However, several challenges limit their broader applicability. First, learning single-point decisions may lead to incomplete characterization of the underlying decision process, failing to capture the true decision-making logic. Second, current methods primarily focus on replicating expert decisions at individual nodes, which may result in insufficient feature extraction for branching decisions (Bengio, Lodi, and Prouvost 2021). Finally, the deterministic nature of imitation learning-based methods limits their exploration capabilities, leading to poor generalization when transferring to different prob-

*Corresponding author.

lem scales or types outside the training distribution (Gasse et al. 2019; Nair et al. 2021).

To balance exploration capability and decision quality in the branch-and-bound framework, we propose a novel methodology for solving MILP problems in a B&B setting. Drawing inspiration from the strong exploration ability of diffusion models (Ho, Jain, and Abbeel 2020; Song et al. 2020; Chi et al. 2024), we adapt this paradigm to enable effective branching score generation via conditional diffusion. The branching score plays a critical role in guiding the B&B process by determining which variable to branch on at each decision node. We thus reformulate branching decisions as a conditional generation process, shifting from learning a single optimal branching variable to modeling the full distribution over branching scores.

To further enhance the generation/inference efficiency, we introduce a consistency modeling approach that distills knowledge from a pretrained diffusion model to achieve efficient inference through single-step generation(As illustrated in Figure 1). Our design enables fast, one-step score generation while preserving the rich decision-making patterns acquired during multi-step diffusion-based training. Overall, our approach significantly improves both generalization ability across diverse MILP problem types and scales, and exploration efficiency in the B&B solving process.

Our main contributions are as follows:

- **W**e reformulate branch decision-making as a conditional generative process, enhancing model exploration capabilities and deep decision-making paradigm by generating multi-modal branch score distributions.
- **T**his paper develope an optimized architecture that maintains exploration capabilities while enabling efficient one-step high-quality score generation by consistency modeling.
- **O**ur consistency policy achieves state-of-the-art performance in single-step prediction accuracy and solving efficiency across cross-problem and cross-size tasks, addressing the generalization limitations of learning-based B&B solvers.

Background

Mixed Integer Linear Programming is defined as:

$$\min \left\{ \mathbf{c}^\top \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b},\, \mathbf{x} \in X \right\} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the decision variable vector with integer variable indices $\mathcal{I} \subseteq \{1, \ldots, n\}$, X represents the mixed-integer feasible region, and $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$ define the constraint matrix, right-hand side vector, and objective coefficients, respectively.

Branch-and-Bound (B&B) is the predominant algorithmic framework for solving MILPs, which systematically partitions the feasible region through recursive branching on fractional integer variables. At each node of the B&B tree, the algorithm solves a linear programming (LP) relaxation to obtain a bound on the objective. If the solution to the LP relaxation contains fractional values for any integer variables, a branching decision must be made: selecting a fractional variable x_i and creating two child nodes with additional constraints $x_i \leq \lfloor x_i^* \rfloor$ and $x_i \geq \lceil x_i^* \rceil$, where x_i^* denotes the fractional value from the LP relaxation. The current **branching state** at each node is defined by the set of active constraints and variable bounds. This state is used to compute the bound and is updated whenever a branching decision is made.

Variable Scoring The efficacy of B&B critically depends on the variable selection strategy, which fundamentally relies on **branching scores** that quantify the potential impact of branching on each candidate variable(Linderoth and Savelsbergh 1999). **Strong Branching (SB)**, a cornerstone of modern branching rules(Achterberg, Koch, and Martin 2005), computes comprehensive scores by evaluating all fractional variables through tentative LP solutions for both potential child nodes. Strong branching compute a comprehensive score for each candidate variable by simulating the effect of branching on it, thereby providing high-quality guidance for branching decisions. Candidate variables with higher scores are expected to offer greater potential improvements for problem solving. While this method achieves superior decision quality, it incurs substantial computational overhead due to the requirement of solving multiple LP relaxation problems for each candidate variable. To address this, current learning-based approaches accelerate problem solving by predicting the highest-scoring variable $a_i = \text{argmax}(\mathbf{s}_i)$ using strong branching scores as supervision (Gasse et al. 2019).

Diffusion Models are generative probabilistic models that learn to reverse a gradual noising process, and have achieved remarkable success in high-fidelity image generation and sequential decision-making tasks (Song et al. 2020; Chi et al. 2024). Given data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ from the true data distribution, the forward diffusion process systematically adds Gaussian noise over T discrete timesteps:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{2}$$

where $\{\beta_t\}_{t=1}^T$ represents a predefined variance schedule controlling the noise magnitude at each step. The reverse generative process learns to iteratively denoise samples through a parameterized neural network:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I}) \tag{3}$$

Training optimizes the variational lower bound of the log-likelihood, which simplifies to a denoising objective that predicts the noise added at each step:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] \tag{4}$$

where $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ with $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and $\bar{\alpha}_t = \prod_{i=1}^t (1-\beta_i)$ representing the cumulative noise schedule. The network $\epsilon_\theta(\mathbf{x}_t, t)$ learns to predict the Gaussian noise ϵ that was added to the clean data $\mathbf{x}_0$ to produce the noisy observation $\mathbf{x}_t$ at timestep t. This noise prediction formulation is equivalent to learning a denoising autoencoder for each timestep, enabling stable training and high-quality generation.

Our Proposed Method

Current imitation learning-based solvers learn direct mappings from node states to branching decisions, suffering from insufficient exploration, inadequate feature learning, and limited generalization. We address these limitations by reformulating branching as a conditional generative task, leveraging the exploration capabilities of diffusion models to enhance performance by using parameterization decomposition. To meet real-time requirements, we further employ consistency models to simplify multi-step generation to single-step inference while maintaining solution quality.

Branching Reformulation as Conditional Probabilistic Generation

At each node g_i from B&B, we model branching decisions by learning a conditional policy distribution $\pi_\theta(\mathbf{s}_i|G_i)$ over candidate scores $\mathbf{s}_i$ given by the current branching state G_i. Our objective is to maximize the likelihood of expert decisions, i.e., high-quality branching scores $\mathbf{s}_i^*$, conditioned on the corresponding branching states, based on an expert dataset $\mathcal{D} = \{(G_i, \mathbf{s}_i^*)_j\}_{j=1}^N$:

$$\mathcal{L}(\theta) = \mathbb{E}_{(G_i,\mathbf{s}_i^*)\sim\mathcal{D}} \left[-\log \pi_\theta(\mathbf{s}_i^*|G_i)\right] \tag{5}$$

To better capture factors beyond the observed branching state, we extend the above formulation by introducing a latent variable model:

$$\pi_\theta(\mathbf{s}_i|G_i) = \int \pi_\theta(\mathbf{s}_i|\mathbf{z}_i, G_i)\pi_\theta(\mathbf{z}_i|G_i)d\mathbf{z}_i \tag{6}$$

where $\mathbf{z}$ denotes latent factors influencing the branching decision. To model complex score distributions, we parameterize π_θ using a diffusion-based generative model, which naturally captures the stochastic exploration required for effective branching decisions. For brevity, we denote the branching state G_i, latent features $\mathbf{z}_i$, and scores $\mathbf{s}_i$ at the i-th branching decision step as G, $\mathbf{z}$, and $\mathbf{s}$, respectively.

Conditional Input Encoding

To enable conditional generation guided by the branching state G, we design a neural encoding architecture that effectively integrates graph-structured information into the diffusion process. Specifically, we learn a noise prediction function $\epsilon_\theta(\mathbf{s}_t, t, G)$, along with a deterministic component $\mu_\theta(G)$, allowing the model to capture both the temporal dynamics of the generative process and the structural semantics of the MILP instance represented by G. Denote each branching state as a bipartite graph $\mathcal{G} = (C \cup V, E)$, where C and V are the sets of constraint and variable nodes, respectively, and E is the edge set encoding their interactions. The adjacency matrix of the graph is denoted as A.

Graph Information Encoding We use a Graph Neural Network (GNN) encoder to embed the branching state G into a high-dimensional vector(Scavuzzo et al. 2022). The encoder processes the bipartite graph through L layers of message passing:

$$\mathbf{h}^{(\ell)} = \sigma\left(\text{GCN}^{(\ell)}(\mathbf{h}^{(\ell-1)}, A, E)\right) + \mathbf{h}^{(\ell-1)} \tag{7}$$

where $\mathbf{h}^{(\ell)}$ represents the node embeddings at layer ℓ.

Context-Aware Representation To enable flexible and stable decisions, we fuse three types of information: the branching state, the current noisy score $\mathbf{s}_t$, and the diffusion timestep t. The fused representation for each candidate variable is given as:

$$\mathcal{F} = \mathbf{h}^{(L)} \oplus \text{ScoreEmb}(\mathbf{s}_t) \oplus \text{TimeEmb}(t) \tag{8}$$

We then apply self-attention across all candidate variables' fused representation to capture inter-variable dependencies at generation step t by residual style:

$$\mathbf{c}_t = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V + \mathcal{F} \tag{9}$$

where $Q = W_Q\mathcal{F}$, $K = W_K\mathcal{F}$, and $V = W_V\mathcal{F}$ with learnable projections $W_Q, W_K, W_V \in \mathbb{R}^{d\times d_k}$. The resulting $\mathbf{c}_t$ serves as the conditional input for noise prediction.

Accordingly, we reformulate the learning objective from modeling the original branching-state-conditional distribution $\pi_\theta(\mathbf{s}|G)$ to modeling the fused representation-conditional distribution $\pi_\theta(\mathbf{s}|\mathbf{c}_0)$. This enables the generated branching scores to be adaptively controlled by the current branching context, thereby enhancing the expressiveness and controllability of the generation process.

Diffusion Modeling for Conditional Branching

We implement the conditional probabilistic generation framework from Eq. 6 using a diffusion process, guided by the fused encoding state $\mathbf{c}$. This formulation enables the model to explore diverse and high-quality branching strategies while remaining strongly conditioned on the current encoding state $\mathbf{c}_t$.

Given a high-quality branching score $\mathbf{s}_0$ conditioned on $\mathbf{c}_0$, we define the diffusion process as follows:

- **Forward Process.** We progressively perturb the target scores by adding Gaussian noise at each timestep :

$$\begin{aligned} q(\mathbf{s}_t|\mathbf{s}_{t-1}) &= \mathcal{N}(\mathbf{s}_t; \sqrt{1-\beta_t}\mathbf{s}_{t-1}, \beta_t\mathbf{I}), \\ \mathbf{s}_t &= \sqrt{\bar{\alpha}_t}\mathbf{s}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \end{aligned} \tag{10}$$

  with marginal distribution $q(\mathbf{s}_t|\mathbf{s}_0) = \mathcal{N}(\mathbf{s}_t; \sqrt{\bar{\alpha}_t}\mathbf{s}_0, (1-\bar{\alpha}_t)\mathbf{I})$, where $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$.
- **Generation (Reverse) Process.** The key innovation is to condition the reverse diffusion process on the fused representation c_t:

$$\pi_\theta(\mathbf{s}_{t-1}|\mathbf{s}_t, \mathbf{c}_t) = \mathcal{N}(\mathbf{s}_{t-1}; \mu_\theta(\mathbf{s}_t, t, \mathbf{c}_t), \sigma_t^2\mathbf{I}) \tag{11}$$

In practice, we implement $\pi_\theta(\mathbf{s}|\mathbf{c}_0)$ as a diffusion process with latent variables $\mathbf{z}_{0:T}$:

$$\pi_\theta(\mathbf{s}, \mathbf{z}_{0:T}|\mathbf{c}_0) \simeq \pi_\theta(\mathbf{s}|\mathbf{z}_0, \mathbf{c}_0)p(\mathbf{z}_T)\prod_{t=1}^{T}\pi_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{c}_t) \tag{12}$$

where $z_1, .., z_T$ are latent variables defining the reverse denoising process from $\mathbf{z}_T \sim \mathcal{N}(0, \mathbf{I})$ to high-quality scores. Thus, the associated evidence lower bound (ELBO) is:

$$\mathbb{E}[-\log \pi_\theta(\mathbf{s}|\mathbf{c}_t)] \leq \mathbb{E}_q\left[D_{\text{KL}} - \log \pi_\theta(\mathbf{s}|\mathbf{z}_0, \mathbf{c}_t)\right] + C \tag{13}$$

where $D_{\text{KL}} = \sum_{t>1} D_{\text{KL}}[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0)\|\pi_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{c}_t)]$ measures generation process quality, and $-\log \pi_\theta(\mathbf{s}|\mathbf{z}_0, c_0)$ ensures score fidelity. Considering the complexity of the scores, we adopt a *structured parameterization decomposition*: $\pi_\theta(\mathbf{s}|\mathbf{z}, \mathbf{c}_t) = \mathcal{N}(\mathbf{s}; \mu_\theta(\mathbf{c}_t) + \alpha\mathbf{z}, \sigma^2\mathbf{I})$, which naturally emerges from the principle that branching scores consist of a controllable deterministic regularization component $\mu_\theta(G)$ derived from the graph structure and a stochastic component $\alpha\mathbf{z}$ capturing decision uncertainties. The reconstruction term becomes:

$$-\log \pi_\theta(\mathbf{s}|\mathbf{z}_0, \mathbf{c}_t) = \frac{1}{2\sigma^2}\|\mathbf{s} - \mu_\theta(\mathbf{c}_t) - \alpha\mathbf{z}_0\|^2 + C \quad (14)$$

C, α is constant. To avoid degeneracy and ensure meaningful learning of $\mu_\theta(\mathbf{c}_t)$ and enable high-quality branching, we reformulate the optimization using the reparameterization $\mathbf{z}_0 \propto \mathbf{s}^* - \mu_\theta(\mathbf{c}_t)$ and noise prediction objective.

Accordingly, training π_θ by optimizing the ELBO can be simplified to minimizing the following loss (with detailed proof in Appendix):

$$\mathcal{L}_\theta = \mathbb{E}\left[\|\epsilon - \epsilon_\theta(\mathbf{s}_t, t, \mathbf{c}_t)\|^2\right] + \mathbb{E}\left[\|\mathbf{s}^* - \mu_\theta(\mathbf{c}_0)\|^2\right] \quad (15)$$

where $\epsilon_\theta(\mathbf{s}_t, t, \mathbf{c}_t)$ is the noise prediction network that learns to predict the Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ added to clean scores $\mathbf{s}_0$ at timestep t, conditioned on the encoding state $\mathbf{c}_t$.

Consistency Policy for Efficient Branching

While our diffusion-based policy $\pi_\theta(\cdot|\cdot)$ demonstrates strong performance, the multi-step sampling process required by the conventional diffusion model incurs substantial computational overhead, limiting its applicability in real-time branch-and-bound decisions. To enhance inference efficiency, we further develop a consistency modeling approach that distills the multi-step conditional diffusion policy into a single-step generation policy.

Consistency Function Learning. Starting from our pre-trained diffusion policy π_θ with T sampling steps, we learn a consistency function $\pi_\phi : (\mathbf{s}_t, t, \mathbf{c}_t) \mapsto \mathbf{s}_0$ that directly maps a noisy state to the clean score distribution. The key insight is that all noisy states along the same diffusion trajectory should correspond to the same underlying clean score $\mathbf{s}_0$. Formally, we enforce the following consistency constraint:

$$\pi_\phi(\mathbf{s}_t, t, \mathbf{c}_t) = \pi_\phi(\mathbf{s}_{t'}, t', \mathbf{c}_{t'}), \quad \forall t, t' \in [0, T] \quad (16)$$

where $\mathbf{s}_t$ and $\mathbf{s}_{t'}$ are different noisy states along the forward diffusion path originating from the same $\mathbf{s}_0$.

Consistency Training Objective. Building upon the pre-trained conditional diffusion policy $\pi_\theta(|)$, we fine-tune the consistency function π_ϕ by enforcing self-consistency across different diffusion timesteps:

$$\mathcal{L}_{\text{consistency}} = \mathbb{E}_{t_1, t_2, \mathbf{s}_0, c.}\left[d(\pi_\phi(\mathbf{s}_{t_1}, t_1, \mathbf{c}_{t_1}), \pi_\phi(\mathbf{s}_{t_2}, t_2, \mathbf{c}_{t_2}))\right] \quad (17)$$

where $t_1 \sim \mathcal{U}(0, T/2)$ and $t_2 \sim \mathcal{U}(T/2, T)$ are sampled from distinct time intervals, $d(\cdot, \cdot)$ denotes the Huber loss, which provides robustness to outliers, and $\mathbf{s}_{t_i} = \sqrt{\bar{\alpha}_{t_i}}\mathbf{s}_0 + \sqrt{1 - \bar{\alpha}_{t_i}}\epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \mathbf{I})$. The consistency function adopts the same architecture as our pre-trained model π_θ and is fine-tuned using the following composite loss:

$$\mathcal{L}_{\text{fine-tune}} = \mathcal{L}_{\text{consistency}} + \lambda_\mu \mathcal{L}_\mu \quad (18)$$

Method	Style	FA (↑, %) acc@1	SC (↑, %) acc@1	CA (↑, %) acc@1	IS (↑, %) acc@1
TREE	ML.	41.88	12.50	22.07	19.13
RF	ML.	49.38	12.50	31.25	14.34
ExtraTree	ML.	23.75	18.75	36.66	18.26
GNN	IL.	66.89	46.50	44.51	49.60
π_ϕ w/o cons	Diff.	67.13	49.20	45.85	51.95
π_ϕ (Ours)	Diff.	**68.73**	**49.30**	**46.25**	**53.05**

Table 1: **Same-Scale branching** with Top-1 accuracy (acc@1).

where $\mathcal{L}_\mu$ is a regularization deterministic decomposition loss that follows the same paradigm as $\mathbb{E}_{(\mathbf{c}_0, \mathbf{s})}\left[\|\mathbf{s}^* - \mu_\phi(c_0)\|^2\right]$ in Eq. 15, ensuring the policy retains the denoising behavior of the original mean estimator with $\mu_\phi(\mathbf{c}_0)$. λ_μ is a hyperparameter. With this training strategy, the consistency function distills the multi-step reasoning capability of the pre-trained diffusion model into a single-step generation policy, with same parameterization as π_θ:

$$\mathbf{s}_{\text{pred}} = \pi_\phi(\epsilon, T, \mathbf{c}_T), \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \quad (19)$$

enabling one-step branching score generation while retaining the exploration and generalization capabilities acquired during pre-training.

Branching with the Consistency Score. We deploy the trained consistency policy $\pi_\phi^{(\text{cons})}$ to perform single-step generation at each encoding state $\mathbf{c}_0$ during the B&B process. To fully leverage the diversity inherent in the score distributions generated by the consistency model, we employ a multi-sampling strategy. Specifically, we draw multiple samples from the consistency distribution, evaluate the confidence of each variable's score across samples, and select the variable with the highest confidence as the final branching variable. Given an encoding state c_T, we perform K independent samplings:

$$\mathbf{s}^{(k)} = \pi_\phi(\epsilon^{(k)}, T, \mathbf{c}_T), \quad \epsilon^{(k)} \sim \mathcal{N}(0, \mathbf{I}), \quad k = 1, \ldots, K \quad (20)$$

For each candidate variable j, we define its confidence as a stability measure of the scores across samples:

$$\mathbf{n}_j = \exp\left(-\frac{\text{Var}(\{s_j^{(k)}\}_{k=1}^K)}{\text{Mean}(\{s_j^{(k)}\}_{k=1}^K)^2 + \delta}\right) \quad (21)$$

where $s_j^{(k)}$ denotes the branching score of variable j in the k-th sampling, and $\delta \geq 0$ is a constant for numerical stability. The final branching variable $a^*_{\text{consistency}}$ given by π_ϕ^{cons} is as:

$$a^*_{\text{consistency}} = \arg\max_j \left[\text{Mean}(\{s_j^{(k)}\}_{k=1}^K)\mathbf{n}_j^\lambda\right] \quad (22)$$

where λ controls the relative importance of confidence versus the expected score.

Experiments

We conduct experiments on a system equipped with an NVIDIA V100 GPU (32GB) and an Intel Core i5 processor

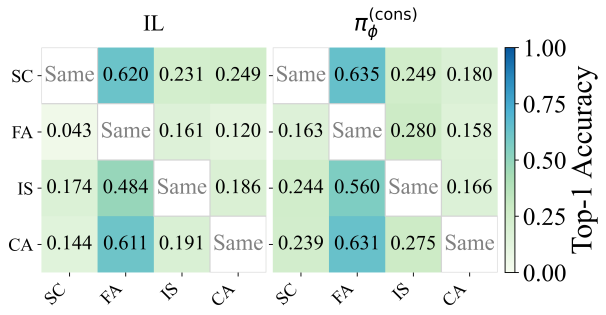

Figure 2: **Cross-problem** branching evaluation

(4 cores, 8 threads). Our experimental evaluation consists of two main components designed to assess the generalization capabilities of our method in both cross-size and cross-problem settings: **(a) Branching Evaluation Experiments:** We train branching policies using individual node state-action pairs to learn effective decision-making strategies at each branching step. **(b) B&B Solving Evaluation:** We apply the trained policies from (a) to the complete branch-and-bound solving process to achieve end-to-end problem resolution.

We first train the conditional multi-step diffusion policy π_θ on a training dataset with 100,000 observation-action pairs, then apply consistency fine-tuning on a separate medium-scale dataset with 5,000 observation-action pairs to get the consistency policy π_ϕ^{cons}(limited in 100/200 epochs). We then evaluate our consistency policy π_ϕ^{cons} in B&B solving experiments on entirely unseen instances.

Benchmark and Empirical Setting

We conduct comprehensive experiments to evaluate our proposed method across multiple tasks and benchmarks, measuring both individual branching results and final solving performance.

Datasets We follow the benchmark from (Gasse et al. 2019) and evaluate on four NP-hard problems: **(a) Set Covering (SC)** with 1,000 columns, training on 500 rows and testing on 500 and 1,000 rows for medium and large scale, respectively(Balas and Ho 1980); **(b) Combinatorial Auction (CA)** with 100 items, training on 500 bids and testing on 500 and 1,000 bids for medium and large scale, respectively(Sandholm 2002); **(c) Capacitated Facility Location (FA)** with 100 facilities, training on 100 customers and testing on 100 and 200 customers for medium and large scales, respectively(Cornuejols, Nemhauser, and Wolsey 1983);**(d) Maximum Independent Set (IS)** on Erdős-Rényi graphs with Medium(500 nodes),Large(1000 nodes)(Karp 1972). We evaluating optimal Gap(%), number of the branching nodes, and solving time within solving limit. For same/cross problem solving, we average the results across five runs.

Baselines We compare against the following baselines: Expert solvers, FSB (Strong Branching),and RPB (Reliability Pseudo-cost Branching) from SCIP (Bestuzheva et al. 2021);Strong imitation learning baselines: GNN-based B&B (Gasse et al. 2019), and T-BranT B&B (Lin et al. 2022);Strong RL-based method (Feng and Yang 2025); and ML-based methods: Extra-Trees, Random forest, and Decision Tree, by part of Khalil features (Khalil et al. 2016).

Cross-problem-type Experiment Settings In our cross-problem transfer experiments, we train the consistency policy on medium-scale MILP instances from a single problem category, denoted as `Source` (*e.g.*, FA), and then evaluate its generalization by transferring the trained policy to medium-scale MILP instances from other problem categories, denoted as `Target` (*e.g.*, SC, CA, IS), where `Target` $\neq$ `Source`.The `Source` (Rows) and `Target` (Columns) notation is used consistently for all cross-problem-type experiments in Figures 2 to 4.

Branching Evaluation

Same-problem-type Evaluation We evaluated our model in two variants, π_ϕ (with consistency fine-tuning) and π_ϕ w/o cons (without consistency fine-tuning), on 1,000 different branching steps from MILP instances of the same type, measuring their inference accuracy against strong imitation learning baselines. Results in Table 1 show that the consistency policy outperforms all baseline methods in Top-1 accuracy, demonstrating its significant potential for performance improvement under computational constraints. Additionally, π_ϕ shows notable improvement in single-step branching accuracy compared to the original model without consistency fine-tuning, further proving that consistency training can simultaneously enhance both inference efficiency and quality.

Cross-problem-type Evaluation We assessed cross-problem branching accuracy, which is a challenging test of model generalization. As shown in Figure 2, our consistency policy achieves significant advantages over the GNN imitation learning baseline in this challenge. Specifically, when trained on `Source` datasets, the consistency policy achieves an average 29.5% improvement over the strong GNN-based imitation learning baseline on `Target` datasets. This proves that our model can learn more robust decision-making patterns by modeling score distributions, thereby enabling effective generalization to different problem categories.

B&B Solving Evaluation

Scale-varying solving We validated our method's generalization by transferring the policy trained on medium-scale problems to both medium- and large-scale problems within the same category. Table 2 shows that, for problems of the same scale, our method significantly outperforms imitation learning across key metrics, including solving time, number of branching nodes, and final optimality gap. In cross-scale transfer experiments, results in Table 3 demonstrate that our method consistently surpasses all baseline methods across all four benchmark datasets, providing strong evidence of its superior cross-scale generalization in problem solving.

Cross-problem-type solving We assessed cross-problem-type generalization, with results for solving time presented and gap in Figure 3 and Figure 4, and summarized in

Method	Style	FA(↓)			SC(↓)		
		Nodes	Time(s)	Gap(%)	Nodes	Time(s)	Gap(%)
FSB(SCIP)	Expert	107.71 ± 114.92	41.71 ± 33.80	0.00 ± 0.00	37.40 ± 57.74	17.13 ± 25.42	0.00 ± 0.00
RPB(SCIP)	Expert	267.30 ± 346.13	38.02 ± 36.03	0.00 ± 0.00	136.23 ± 398.60	8.55 ± 3.46	0.00 ± 0.00
TREE	ML.	512.42 ± 521.04	35.24 ± 36.85	0.02 ± 0.07	362.73 ± 447.44	11.81 ± 11.44	0.36 ± 0.01
RF	ML.	571.22 ± 506.68	36.92 ± 32.51	0.02 ± 0.09	916.48 ± 225.54	19.90 ± 5.38	5.87 ± 2.87
ExtraTree	ML.	482.36 ± 507.44	37.73 ± 39.71	0.01 ± 0.07	344.28 ± 463.25	11.53 ± 14.57	0.44 ± 1.39
GNN	IL.	77.70 ± 49.66	**22.47** ± 4.77	0.00 ± 0.00	194.66 ± 225.62	8.20 ± 4.08	0.00 ± 0.00
T-BranT	IL.	**77.70** ± 65.77	22.77 ± 6.60	0.00 ± 0.00	527.62 ± 318.72	11.98 ± 5.01	0.00 ± 0.00
SORREL	RL.	545.18 ± 1384.46	40.57 ± 91.23	0.00 ± 0.00	453.08 ± 522.01	8.85 ± 5.36	0.00 ± 0.00
$\pi_\phi^{(\text{cons})}$ (Ours)	Diff.	83.64 ± 104.00	23.17 ± 5.62	**0.00** ± 0.00	**166.98** ± 245.95	**7.86** ± 4.38	**0.00** ± 0.00
Method	Style	CA(↓)			IS(↓)		
		Nodes	Time(s)	Gap(%)	Nodes	Time(s)	Gap(%)
FSB(SCIP)	Expert	11.17 ± 91.62	3.21 ± 3.52	0.00 ± 0.00	34.30 ± 59.50	38.66 ± 59.50	0.00 ± 0.00
RPB(SCIP)	Expert	23.43 ± 29.46	2.19 ± 0.96	0.00 ± 0.00	123.68 ± 206.56	8.77 ± 0.98	0.00 ± 0.00
TREE	ML.	270.68 ± 310.61	3.62 ± 2.02	0.00 ± 0.00	750.72 ± 809.87	10.21 ± 9.47	0.48 ± 0.84
RF	ML.	417.07 ± 443.75	17.32 ± 4.64	1.87 ± 1.02	1024.46 ± 812.06	36.33 ± 28.03	1.01 ± 1.09
ExtraTree	ML.	242.82 ± 225.12	3.94 ± 2.94	0.00 ± 0.00	732.88 ± 799.67	19.04 ± 19.21	0.39 ± 0.83
GNN	IL.	105.74 ± 95.24	2.60 ± 0.72	0.00 ± 0.00	99.06 ± 151.29	7.85 ± 2.98	0.00 ± 0.00
T-BranT	IL.	716.06 ± 245.91	6.63 ± 1.89	0.19 ± 0.12	270.20 ± 342.62	9.92 ± 5.15	0.45 ± 0.87
SORREL	RL.	**99.76** ± 104.97	2.62 ± 0.41	0.00 ± 0.00	88.36 ± 140.21	8.08 ± 1.36	0.00 ± 0.00
$\pi_\phi^{(\text{cons})}$ (Ours)	Diff.	113.00 ± 92.67	**2.58** ± 0.75	**0.00** ± 0.00	**65.86** ± 89.51	**7.59** ± 2.59	**0.00** ± 0.00

Table 2: **Same-scale** B&B solving results for the consistency policy on separate instances, reporting solving time, gap(%) of the instances, and number of resulting B&B nodes (lower is better).

Method	Style	FA(↓)			SC(↓)		
		Nodes	Time(s)	Gap(%)	Nodes	Time(s)	Gap(%)
FSB(SCIP)	Expert	226.70 ± 204.60	368.90 ± 313.60	0.00 ± 0.00	357.40 ± 347.13	423.11 ± 414.10	0.00 ± 0.00
RPB(SCIP)	Expert	1001.58 ± 1001.36	211.39 ± 210.89	0.00 ± 0.00	3719.78 ± 8170.42	47.47 ± 53.30	0.00 ± 0.00
TREE	ML.	869.40 ± 513.77	204.12 ± 100.27	0.13 ± 0.17	1307.15 ± 66.93	33.86 ± 4.26	3.55 ± 3.36
RF	ML.	1099.90 ± 442.94	226.77 ± 85.96	0.23 ± 0.26	916.48 ± 225.54	19.90 ± 5.38	10.83 ± 2.48
ExtraTree	ML.	815.85 ± 418.30	203.97 ± 97.19	0.02 ± 0.07	1348.80 ± 499.31	41.46 ± 16.70	3.46 ± 3.27
GNN	IL.	**358.62** ± 212.88	214.31 ± 87.68	0.20 ± 0.34	693.72 ± 174.94	40.09 ± 10.47	5.31 ± 3.64
T-BranT	IL.	360.12 ± 188.29	256.70 ± 78.54	0.28 ± 0.37	718.20 ± 154.81	39.45 ± 8.70	5.95 ± 3.74
SORREL	RL.	943.70 ± 702.87	**189.47** ± 142.25	0.01 ± 0.04	3301.20 ± 4177.52	40.66 ± 49.40	0.00 ± 0.00
$\pi_\phi^{(\text{cons})}$ (Ours)	Diff.	**499.90** ± 252.56	272.02 ± 160.94	**0.00** ± 0.00	**691.50** ± 360.81	**33.53** ± 13.38	**0.00** ± 0.00
Method	Style	CA(↓)			IS(↓)		
		Nodes	Time(s)	Gap(%)	Nodes	Time(s)	Gap(%)
FSB(SCIP)	Expert	73.14 ± 76.38	99.85 ± 61.76	0.00 ± 0.00	183.38 ± 137.32	1638.10 ± 642.51	2.77 ± 2.39
RPB(SCIP)	Expert	1536.12 ± 1131.83	14.56 ± 5.89	0.00 ± 0.00	6515.05 ± 6666.89	111.51 ± 91.42	0.00 ± 0.00
TREE	ML.	1447.85 ± 525.27	12.30 ± 3.88	0.77 ± 0.70	1576.50 ± 169.51	64.38 ± 4.61	2.33 ± 0.66
RF	ML.	1562.90 ± 160.42	34.87 ± 2.35	2.82 ± 0.99	1407.80 ± 156.62	155.59 ± 13.77	2.76 ± 0.76
ExtraTree	ML.	1436.00 ± 538.76	31.41 ± 11.24	0.74 ± 0.65	1648.10 ± 175.78	93.86 ± 6.74	2.32 ± 0.75
GNN	IL.	693.36 ± 293.22	16.14 ± 4.89	0.22 ± 0.45	737.94 ± 214.28	59.06 ± 12.10	1.24 ± 1.19
T-BranT	IL.	720.08 ± 111.38	20.76 ± 4.47	2.51 ± 1.13	762.50 ± 177.80	61.40 ± 13.16	1.93 ± 1.27
SORREL	RL.	1410.30 ± 1331.14	**10.40** ± 5.78	0.00 ± 0.00	9070.45 ± 10679.15	106.49 ± 98.17	**0.00** ± 0.00
$\pi_\phi^{(\text{cons})}$ (Ours)	Diff.	**686.04** ± 390.09	14.35 ± 6.25	**0.00** ± 0.00	**662.71** ± 486.95	**49.69** ± 18.43	0.81 ± 1.08

Table 3: **Cross-scale** B&B solving results for the consistency policy on separate **larger** instances.

Table 4. Our consistency policy achieves significant improvements on cross-category problems: the model trained on the `Source` dataset reduces average solving time by 11.0% and improves the final optimality gap by 27.8% on `Target` datasets within same gap/step limit on extension 50 same-medium-scale instances. The results demonstrate that, by modeling branching score distributions, our consistency model captures richer decision-making patterns and achieves substantially better generalization than imitation learning on challenging cross-category tasks.

Source	Gap Improve(↑,%)	Time Improve (↑,%)
SC	+7.3	+6.9
FA	+34.4	+13.1
IS	+60.2	+13.2
CA	+9.2	+10.9
Avg Imp.	+27.8	+11.0

Table 4: Improvements of π_ϕ over IL on cross-problem tasks.

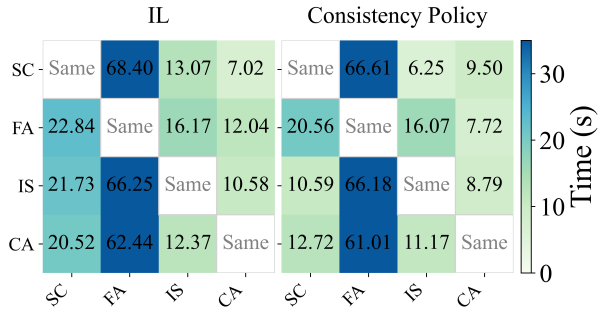

Figure 3: **Cross-problem** generalization results on solving time.

Ablation study

We conducted ablation studies to analyze the impact of consistency fine-tuning.

Inference time. We compared the inference time per branching step for the policies before and after consistency fine-tuning. Table 5 shows that the consistency policy significantly reduces inference time.

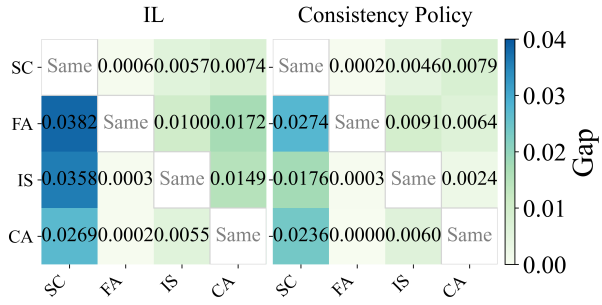

Figure 4: **Cross-problem** generalization results on Gap.

Cross-problem-type branching. We evaluated the generalization capability before and after consistency fine-tuning on cross-category problems with results in Table 6. While the consistency model continues to achieve significant improvements over strong imitation learning baselines, its generalization capability is slightly weakened compared to its pre-fine-tuning variant. This may be attributed to the loss of fine-grained information and intermediate representations during the consistency fine-tuning process, which can be beneficial for cross-category generalization.

Trade-off between generalization and efficiency. Although there is a slight reduction in generalization, the consistency fine-tuned model achieves inference speeds that are several to dozens of times faster than the pre-fine-tuning model during branch-and-bound search. This highlights a favorable trade-off, demonstrating that our approach achieves an effective balance between generalization performance and computational efficiency.

Source	Sample step η (↓)		Inference Time (ms)(↓)	
	π_ϕ w/o cons	π_ϕ	π_ϕ w/o cons	π_ϕ
FA	5	1	93.2	5.7
SC	50	1	1071.7	7.2
CA	10	1	216.2	6.1
IS	5	1	127.6	6.1

Table 5: Inference time per branching step for policies before and after consistency fine-tuning.

Policy Style	acc@1 Improve (↑,%)				Avg.
	FA	SC	CA	IS	
π_ϕ w/o cons	+**95.7**	+**9.2**	+8.9	+**24.4**	+**34.5**
π_ϕ(Ours)	+85.5	−3.3	+**21.0**	+14.9	+29.5

Table 6: Top-1 accuracy (acc@1) on cross-problem instances for policies before and after consistency fine-tuning.

Conclusion

In this paper, we propose the consistency policy for efficient generation of branching score distributions in branch-and-bound solvers. By reformulating branching variable selection in B&B as a conditional generative modeling task, our model addresses the limitations of imitation learning methods, particularly their lack of uncertainty modeling and exploration. Specifically, we leverage diffusion policies for branching score generation and incorporate consistency-based inference to improve efficiency while preserving decision quality. Experimental results demonstrate that our consistency policy achieves significant improvements in both cross-scale and cross-problem-type generalization. Overall, this work highlights the potential of generative modeling in B&B solving and paves the way for future MILP B&B solvers.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedback. This work was partially supported by the NSFC under Grants 92270125 and 62276024; by the Fundamental Research Funds for the Central Universities, JLU, under Grant 93K172025K01;and by the Fundamental Research Funds for the Central Universities under Grant 2025CX01010;and funded by the Beijing Natural Science Foundation (Grant No.: 4262066).

References

Achterberg, T.; Koch, T.; and Martin, A. 2005. Branching rules revisited. *Oper. Res. Lett.*, 33(1): 42–54.

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

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

Bestuzheva, K.; Besançon, M.; Chen, W.-K.; Chmiela, A.; Donkiewicz, T.; van Doornmalen, J.; Eifler, L.; Gaul, O.; Gamrath, G.; Gleixner, A.; Gottwald, L.; Graczyk, C.; Halbig, K.; Hoen, A.; Hojny, C.; van der Hulst, R.; Koch, T.; Lübbecke, M.; Maher, S. J.; Matter, F.; Mühmer, E.; Müller, B.; Pfetsch, M. E.; Rehfeldt, D.; Schlein, S.; Schlösser, F.; Serrano, F.; Shinano, Y.; Sofranac, B.; Turner, M.; Vigerske, S.; Wegscheider, F.; Wellner, P.; Weninger, D.; and Witzig, J. 2021. The SCIP Optimization Suite 8.0. arXiv:2112.08872.

Chi, C.; Xu, Z.; Feng, S.; Cousineau, E.; Du, Y.; Burchfiel, B.; Tedrake, R.; and Song, S. 2024. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. arXiv:2303.04137.

Cook, S. A. 1971. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, 151–158. New York, NY, USA: Association for Computing Machinery. ISBN 9781450374644.

Cornuejols, G.; Nemhauser, G. L.; and Wolsey, L. A. 1983. The Uncapacitated Facility Location Problem. Technical report.

Feng, S.; and Yang, Y. 2025. SORREL: Suboptimal-Demonstration-Guided Reinforcement Learning for Learning to Branch. arXiv:2412.15534.

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.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. arXiv:2006.11239.

Karp, R. M. 1972. *Reducibility among Combinatorial Problems*, 85–103. Boston, MA: Springer US. ISBN 978-1-4684-2001-2.

Khalil, E.; Le Bodic, P.; Song, L.; Nemhauser, G.; and Dilkina, B. 2016. Learning to branch in mixed integer programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

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

Lin, J.; Zhu, J.; Wang, H.; and Zhang, T. 2022. Learning to branch with Tree-aware Branching Transformers. *Knowledge-Based Systems*, 252: 109455.

Linderoth, J. T.; and Savelsbergh, M. W. 1999. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2): 173–187.

Meng, L.; Zhang, C.; Zhang, B.; Gao, K.; Ren, Y.; and Sang, H. 2023. MILP modeling and optimization of multi-objective flexible job shop scheduling problem with controllable processing times. *Swarm and Evolutionary Computation*, 82: 101374.

Morrison, D. R.; Jacobson, S. H.; Sauppe, J. J.; and Sewell, E. C. 2016. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19: 79–102.

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

Nemhauser, G. L.; and Wolsey, L. A. 1988. *Integer and combinatorial optimization*. John Wiley & Sons.

Reda, M.; Onsy, A.; Haikal, A. Y.; and Ghanbari, A. 2024. Path planning algorithms in the autonomous driving system: A comprehensive review. *Robotics and Autonomous Systems*, 174: 104630.

Sandholm, T. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1): 1–54.

Scavuzzo, L.; Chen, F. Y.; Chételat, D.; Gasse, M.; Lodi, A.; Yorke-Smith, N.; and Aardal, K. 2022. Learning to branch with Tree MDPs. arXiv:2205.11107.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Wolsey, L. A. 2020. *Integer programming*. John Wiley & Sons.

Yao, Y.; Liu, Q.-H.; Li, X.-Y.; and Gao, L. 2023. A novel MILP model for job shop scheduling problem with mobile robots. *Robotics and Computer-Integrated Manufacturing*, 81: 102506.

Zelaschi, A.; Pliotti, L.; Betti, G.; Tonno, G.; Sgrò, D.; and Martelli, E. 2025. An effective MILP model for the optimal design of microgrids with high-reliability requirements. *Applied Energy*, 383: 125359.

Zhao, L.; Cheng, W.; Meng, L.; Zhang, C.; Ren, Y.; Zhang, B.; and Duan, P. 2025. MILP modeling and optimization of

flexible job shop scheduling problem with preventive maintenance. *Computers & Industrial Engineering*, 201: 110861.