On Threshold Functions in Learning to Generate Feasible Solutions of Mixed Integer Programs

Anonymous authors
Paper under double-blind review

Abstract

Finding a high-quality feasible solution to a combinatorial optimization problem in a given time budget is a challenging task due to its discrete nature. Neural diving is a learning-based approach to generating partial assignments for the discrete variables in MIP. We find that there usually is a small range of selection rates which lead to feasible and optimal solutions; when too many parameters are selected, the solution space is too restricted to find a feasible solution; when too few parameters are selected, the solution space is too wide to efficiently find a feasible solution. Therefore, the choice of selection rate is the critical determinant of the Neural diving performance. In this context, we present theoretical insights that there exist threshold functions in feasibility and feasible optimality over the selection rate. Based on the theoretical foundations, we introduce a post-hoc method, and a learning-based approach to optimize the selection rate for partial discrete variable assignments in MIP more efficiently. A key idea is to jointly learn to restrict the selection rate search space, and to predict the selection rate in the learned search space that results in a high-quality feasible solution. MIP solver is integrated into the end-to-end learning framework. We suggest that learning a deep neural network to generate a threshold-aware selection rate is effective in finding high-quality feasible solutions more quickly. Experimental results demonstrate that our method achieves state-of-the-art performance in NeurIPS ML4CO datasets. In the workload apportionment dataset, our method achieves the optimality gap of 0.45%, which is around $10\times$ better than SCIP, at the one-minute time limit.

1 Introduction

In diverse areas, such as transportation, finance, communication, manufacturing, retail, and system design, many optimization problems have combinatorial characteristics. Mixed integer programming (MIP) is a mathematical optimization model to solve such combinatorial optimization problems. A workhorse of the MIP solver is a group of heuristics (Berthold, 2006; Achterberg et al., 2005; 2012; Fischetti et al., 2005) engineered to solve problems of a specific class in practice. Recently, there has been an increase in interest to apply data-driven methods to complement heuristics in the MIP solvers (He et al., 2014; Khalil et al., 2017; Balcan et al., 2018; Gasse et al., 2019; Gupta et al., 2020; Sun et al., 2020; Tang et al., 2020; Yilmaz & Yorke-Smith, 2020; Song et al., 2020; Paulus et al., 2021; Qi et al., 2021; Zarpellon et al., 2020; Huang et al., 2022).

Finding a high-quality feasible solution to a combinatorial optimization problem in a given time budget is an essential, yet challenging task due to its discrete nature. In this context, learning-based approaches to accelerate the primal solution process (Xavier et al., 2021; Nair et al., 2020; Sonnerat et al., 2021; Shen et al., 2021; Ding et al., 2020; Khalil et al., 2022) have been proposed. These methods show empirical results that outperform the conventional solver-tuning approaches. On the other hand, an unsupervised learning framework to learn to solve combinatorial optimization problems on graphs proposed by Karalias & Loukas (2020) provides theoretical results based on probabilistic methods. Still, it differs from other methods, such that the framework in Karalias & Loukas (2020) does not leverage the MIP solver. Nair et al. (2020) suggests Neural diving along with a variant of SelectiveNet (Geifman & El-Yaniv, 2019) to jointly learn diving style primal heuristics to generate feasible solutions and to adjust the coverage of a discrete variable assignment. The problem is that one must set the target coverage as a hyperparameter to train SelectiveNet while being unaware of the optimal coverage prior to training Neural diving models. Thus, training multiple models with different coverage hyperparameters to find the model with the optimal coverage is inefficient. Also, we empirically find that SelectiveNet's choice of variables is often inferior to simple heuristics (see Section 5). To the best of our knowledge, our work is the first attempt to study theoretical aspects of partial variable assignments from the feasible solution generative models in MIP.

We propose a simple heuristic of setting a confidence threshold hyperparameter, and selecting all variables of which the confidence score is above this threshold. We call this Post-hoc Confidence Thresholding. We further propose a systematic approach to optimize the assignment coverage of discrete variables on theoretical grounds. Our approach is inspired by a connection between the threshold behavior in fixing integer variables of MIP and threshold functions in probabilistic combinatorics (Erdos et al., 1960; Bollobás, 1981), characterizing the point at which an abrupt state change occurs in an asymptotic case. The key insight in this paper is to estimate threshold functions in a data-driven way to provide a smaller coverage search space with a theoretical guarantee that the near-optimal coverage resides in such space. For gradient-based learning, we propose a new loss function to jointly estimate the threshold functions and the optimal coverage in a restricted search space with theoretical results.

Our contributions are threefold:

1) Theoretically, we show the existence of threshold functions in learning to generate feasible solutions of MIP.
2) Methodologically, we devise a new framework and loss function that enables more efficient coverage optimization by learning to restrict the cardinality of the search space with theoretical justifications.
3) Empirically, we demonstrate that our method outperforms the existing methods in various MIP datasets.

2 Preliminaries

2.1 Notations on Mixed Integer Programming

Let n be the number of variables, m be the number of linear constraints, $\mathbf{x} = [x_1, \dots, x_n] \in \mathbb{R}^n$ be the vector of variable values, $\mathbf{c} \in \mathbb{R}^n$ be the vector of objective coefficients, $\mathbf{A} \in \mathbb{R}^{m\times n}$ be the linear constraint coefficient matrix, $\mathbf{b} \in \mathbb{R}^m$ be the vector of linear constraint upper bound. Let $\{\mathbf{e}_i = \mathbf{e}_i^{(n)} \in \mathbb{R}^n : i = 1, \dots, n\}$ be the Euclidean unit basis vector of $\mathbb{R}^n$ and $[n] := \{1, 2, \dots, n\}$. Hence we can decompose the variable vector as $\mathbf{x} = \sum_{i\in B} x_i \mathbf{e}_i$.

A Mixed Integer Program (MIP) is a mathematical optimization problem, in which variables are constrained by linear and integrality constraints. Let r be the number of discrete variables. For a given triplet $(\mathbf{A}, \mathbf{b}, \mathbf{c})$, we express an MIP problem $M = (\mathbf{A}, \mathbf{b}, \mathbf{c})$ as follows:

$$\begin{aligned} &\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \\ &\text{subject to } \mathbf{A}\mathbf{x} \le \mathbf{b} \\ &\qquad \ell_i \le x_i \le u_i, \quad i = 1, \dots, n. \end{aligned} \tag{1}$$

where $\mathbf{x} = [x_1, \dots, x_n] \in \mathbb{Z}^r \times \mathbb{R}^{n-r}$ and $-\infty \le \ell_i, u_i \le \infty$ be the lower and upper variable, respectively. We say $\mathbf{x} \in \mathbb{Z}^r \times \mathbb{R}^{n-r}$ is a *feasible solution* and write $\mathbf{x} \in \mathcal{R}(M)$ if equation 1 holds. We call $\mathbf{x} \in \mathbb{R}^n$ is a *LP-feasible solution* and write $\mathbf{x} \in \bar{\mathcal{R}}(M)$ when $\mathbf{x}$ satisfies the linear constraint regardless of the integrality constraints. We denote discrete variables as $\mathbf{x}_{\text{int}} \in \mathbb{Z}^r$, and continuous variables as $\mathbf{x}_{\text{cont}} \in \mathbb{R}^{n-r}$. We write $\mathbf{x} = [\mathbf{x}_{\text{int}}; \mathbf{x}_{\text{cont}}]$ where $[\cdot;\cdot]$ means a usual concatenation operation between two vectors, matrices, or tensors.

2.2 Neural diving

Neural diving (Nair et al., 2020) is a method to learn a generative model that targets diving heuristics to find a high-quality feasible solution. By learning a Bernoulli distribution of a solution value of each discrete variable in a supervised manner, Neural diving generates a partial assignment to the variables using the learned model. To combine model predictions with the off-the-shelf solver, it is more effective to assign a subset of the predicted solution values than to assign the full sample $\mathbf{x} \sim p_\theta(\mathbf{x}|M)$ for an unseen problem M, since the full sample is not necessarily feasible or high-quality. In this context, Neural diving adopts Selectivenet (Geifman & El-Yaniv, 2019) to optimize the coverage for variable assignment in an integrated manner. The Neural diving with the Selectivenet model jointly learns to predict solution value and selectively refrain from predicting for each variable. The output of the Selectivenet is $s_d \in \{0, 1\}$, such that the variable x_d is fixed with the sample $\mathbf{x} \sim p_\theta(\mathbf{x}|M)$ if $s_d = 1$, otherwise not fixed. In the evaluation phase, the subset of discrete variables is assigned with the learned model, then an off-the-shelf MIP solver optimizes the resultant sub-MIP.

3 Post-hoc Confidence Thresholding

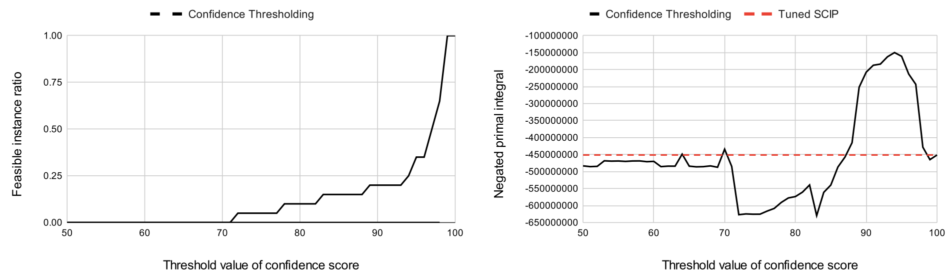

Figure 1: Feasible instance ratio and negated primal integral over the threshold value of the Confidence Thresholding method, at the 30-minute time limit. Negated primal integral represents the solution quality, the higher the better. The target dataset is Maritime Inventory Routing from (Papageorgiou et al., 2014; Gasse et al., 2022). The threshold value of the confidence score is inversely proportional to the coverage rate in general.

From our exploratory experiments, we observe that: (1) Adjusting coverage rate is critical to improving the primal objective; (2) Ordering variables by confidence score is effective to improve the primal objective; (3) An abrupt change appears in the feasibility and optimality of the MIP solution over the coverage rate of discrete variables, as shown in Figure 1. Here we define the coverage $\rho \in [0, 1]$, such that $\rho = n_s/n_v$, where n_s is the number of discrete variables to be fixed, and n_v is the number of total discrete variables in the problem.

In Neural diving, the SelectiveNet module aims to optimize the coverage with the target hyperparameter. However, the optimal coverage is unknown in advance, *i.e.* we need to train the multiple neural network models to explore the optimal coverage. It requires either a non-trivial amount of computation resource or training time, whereas fast adaptation or calibration is important in the practical use case. On the other hand, the key advantage of using SelectiveNet is an improved predictive power in a covered domain. We empirically point out that there is a wide discrepancy between the measure for a neural network prediction, *i.e.* accuracy, and the actual MIP solution quality induced from the neural network. Instead, we find that the solution feasibility and quality show a drastic change depending on the coverage rate, as shown in Figure 1.

In this context, we propose Post-hoc Confidence Thresholding (CT) to calibrate the coverage rate in place of the SelectiveNet. Our method drastically improves the training inefficiency of SelectiveNet in Neural diving by requiring a single Neural diving model without SelectiveNet. To control the coverage of discrete variables to fix, CT adjusts the threshold value for the confidence score of the Neural diving model output $p_\theta(\mathbf{x}|M)$. We define the confidence score as $|p_\theta(x_d|M) - 0.5|$. Suppose the target threshold value is Γ, then the selection of variable x_d to fix is determined as

$$s_d = \begin{cases} 1 & \text{if } |p_\theta(x_d|M) - 0.5| \geq \Gamma \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

In the NeurIPS 2021 ML4CO competition (Gasse et al., 2022), the Confidence Thresholding method ranked second on the global leaderboard, showing the best performance in the primal task among other learning-based methods. We use this method as a motivation point to unfold the theory of threshold functions for coverage of the discrete variables in learning to generate feasible solutions of Mixed Integer Programs.

4 Threshold functions of partial variables in MIP

In this paper, we follow the convention in (Bollobás & Thomason, 1987). Let $\mathscr{P}(B_n)$ denotes the power set of $B_n = \{1, \ldots, n\}$. A *nontrivial property* $\mathcal{H}_n$ is a nonempty collection of set $B \in \mathscr{P}(B_n)$ with $\mathcal{H}_n \neq \mathscr{P}(B_n)$ satisfying the corresponding property of $\mathcal{H}_n$. We call $\mathcal{H}_n$ *monotone increasing* if $B_{ss} \in \mathcal{H}_n$ and $B_{ss} \subset B_s \in \mathscr{P}(B_n)$ implies $B_s \in \mathcal{H}_n$. We call $\mathcal{H}_n$ *monotone decreasing* if $B_s \in \mathcal{H}_n$ and $B_{ss} \subset B_s$ implies $B_{ss} \in \mathcal{H}_n$. Note that a monotone increasing (decreasing) property $\mathcal{H}_n$ is non-trivial *if and only if* $\emptyset \notin \mathcal{H}_n$ and $B_n \in \mathcal{H}_n$ ($\emptyset \in \mathcal{H}_n$ and $B_n \notin \mathcal{H}_n$).

For $i = 0, \ldots, n$, let $\mathcal{B}_i \subset \mathscr{P}(B_n)$ denotes the collection of subsets $B \subset B_n$ with i elements, i.e., $|B| = i$ and $|\mathcal{B}_i| = \binom{n}{i}$, and define $\mathcal{H}_i := \{B \in \mathcal{H}_n \cap \mathcal{B}_i\}$. Note that

$$\mathbb{P}(\mathcal{H}_n | \mathcal{B}_i) = \frac{|\mathcal{H}_i|}{|\mathcal{B}_i|} = \frac{|\mathcal{H}_i|}{\binom{n}{i}} \tag{3}$$

where the conditional probability $\mathbb{P}(\mathcal{H}_n|\mathcal{B}_i)$ denotes the probability that a random set of $B \in \mathcal{B}_i$ has property $\mathcal{H}_n$. If an increasing sequence $m = m(n)$ satisfies that $\mathbb{P}(\mathcal{H}_n|\mathcal{B}_{m(n)}) \to 1$ as $n \to \infty$, then we say that m-subset of $B \in \mathcal{B}_{m(n)}$ has $\mathcal{H}_n$ almost surely. Similarly, we say that m-subset of $B \in \mathcal{B}_{m(n)}$ fails to have $\mathcal{H}_n$ almost surely if $\mathbb{P}(\mathcal{H}_n|\mathcal{B}_{m(n)}) \to 0$ as $n \to \infty$.

Definition 1. *A function $m^*(n)$ is a threshold function for a monotone increasing property $\mathcal{H}_n$ if for $m/m^* \to 0$ as $n \to \infty$ (we write $m \ll m^*$), m-subset of $\mathcal{B}_n$ fails to have $\mathcal{H}_n$ almost surely and for $m/m^* \to \infty$ as $n \to \infty$ (we write $m \gg m^*$), m-subset of $\mathcal{B}_n$ has $\mathcal{H}_n$ almost surely:*

$$\lim_{n\to\infty} \mathbb{P}(\mathcal{H}_n|\mathcal{B}_{m(n)}) = \begin{cases} 0 & : m(n) \ll m^*(n) \\ 1 & : m(n) \gg m^*(n). \end{cases} \tag{4}$$

Similarly, a function $m^*(n)$ is said to be a threshold function for a monotone decreasing property $\mathcal{H}_n$ if for $m/m^* \to \infty$ as $n \to \infty$, m-subset of $\mathcal{B}_n$ fails to have $\mathcal{H}_n$ almost surely and for $m/m^* \to 0$ as $n \to \infty$, m-subset of $\mathcal{B}_n$ has $\mathcal{H}_n$ almost surely:

$$\lim_{n\to\infty} \mathbb{P}(\mathcal{H}_n|\mathcal{B}_{m(n)}) = \begin{cases} 0 & : m(n) \gg m^*(n) \\ 1 & : m(n) \ll m^*(n). \end{cases} \tag{5}$$

(Bollobás & Thomason, 1987, Theorem 4) states that every monotone increasing non-trivial property has a threshold function. In Theorem 3 and Corollary 2, we introduce the formal statement about the existence of a threshold function in monotone increasing and decreasing properties with a slight variation from (Bollobás & Thomason, 1987). See Appendix A.2 for detail.

We say $\mathcal{H}_n$ is *bounded monotone decreasing* if $B_s \in \mathcal{H}_i$ and $B_{ss} \subset B_s \in \mathcal{B}_i$ implies $B_{ss} \in \mathcal{H}_i$ for $i < n$. We say $\mathcal{H}_n$ is *bounded monotone increasing* if $B_{ss} \in \mathcal{H}_i$ and $B_{ss} \subset B_s \in \mathcal{B}_i$ implies $B_s \in \mathcal{H}_i$ for $i < n$. We say $\mathcal{H}_n$ *bounded monotone decreasing* if $B_s \in \mathcal{H}_i$ and $B_{ss} \subset B_s \in \mathcal{B}_i$ implies $B_{ss} \in \mathcal{H}_i$ for $i < n$. We introduce a local version of a threshold function in the Definition 1 for a bounded monotone increasing and decreasing property.

Definition 2. *A function $l^*(n)$ is a local threshold function for a bounded monotone increasing property $\mathcal{H}_n$ if for $l/l^* \to 0, l^*/i \to 0$ as $n \to \infty$, l-subset of $\mathcal{B}_n$ fails to have $\mathcal{H}_n$ almost surely, and for $l/l^* \to \infty, i/l \to \infty$ as $n \to \infty$, l-subset of $\mathcal{B}_n$ has $\mathcal{H}_n$ almost surely:*

$$\lim_{n\to\infty} \mathbb{P}(\mathcal{H}_n|\mathcal{B}_{l(n)}) = \begin{cases} 0 & : l(n) \ll l^*(n) \ll i(n) \\ 1 & : i(n) \gg l(n) \gg l^*(n). \end{cases} \tag{6}$$

Similarly, a function $l^*(n)$ is said to be a local threshold function for a bounded monotone decreasing property $\mathcal{H}_n$ if for $l/l^* \to \infty, i/l \to \infty$ as $n \to \infty$, l-subset of $\mathcal{B}_n$ fails to have $\mathcal{H}_n$ almost surely and for $l/l^* \to 0, l^*/i \to 0$ as $n \to \infty$, l-subset of $\mathcal{B}_n$ has $\mathcal{H}_n$ almost surely:

$$\lim_{n\to\infty} \mathbb{P}(\mathcal{H}_n|\mathcal{B}_{l(n)}) = \begin{cases} 0 & : i(n) \gg l(n) \gg l^*(n) \\ 1 & : l(n) \ll l^*(n) \ll i(n). \end{cases} \tag{7}$$

From Definition 2, we have the statement about local threshold functions in bounded monotone properties (See Theorem 4 in Appendix A.2). Theorem 4 states that every nontrivial bounded monotone increasing property has a local threshold function in a bounded domain. Refer to Appendix A.2 for all of the proofs in this manuscript.

Random subset We introduce a *random subset* $S(n, \rho)$ as a set-valued random variable. Here n is the number of elements of a superset $[n] \supset S(n, \rho)$, and $\rho \in [0, 1]$ is the probability of an element to be included in the subset. The inclusion of each element is statistically independent of the inclusion of all other elements. In this paper, $S(n, \rho) \in \mathcal{P}$ for some collection of subsets $\mathcal{P}$ means that a realization of random variable $S(n, \rho)$ belongs to $\mathcal{P}$.

4.1 Properties of partial variables

Given an MIP problem $M = (\mathbf{A}, \mathbf{b}, \mathbf{c})$, we have $\mathbf{x} = \mathbf{x}_{\text{int}}$ from the Neural diving model $p_\theta(M)$.

Feasibility of partial variables We formulate the property of partial variables feasibility to show the property has a threshold function. The threshold function serves as a criterion for choosing a coverage rate for partial variable assignment to generate a feasible solution, depending on the learned model and the input problem. We assume M has a feasible solution, and $\mathbf{x}$ is not necessarily a feasible solution.

Given $\mathbf{x}$, and M, we define a property $\mathcal{P}$ of an arbitrary set $B_{\text{sub}} \subset [n]$ as

$$\mathcal{P}(\mathbf{x}, \mathbf{A}, \mathbf{b}) := \{B_{\text{sub}} \subset [n] : \text{ there exists } \mathbf{y}' \in \mathbb{Z}^r \times \mathbb{R}^{n-r} \text{ subject to } \mathbf{A}[\mathbf{y} + \mathbf{y}'] \leq \mathbf{b}\}, \tag{8}$$

where $\mathbf{y} = \sum_{i \in B_{\text{sub}}} x_i \mathbf{e}_i$, and $\mathbf{y}' = \sum_{i \in [n] \setminus B_{\text{sub}}} y_i \mathbf{e}_i$. If $\mathbf{x} \in \mathcal{R}(M)$, then $\mathcal{P} = \mathscr{P}(B)$, i.e, $\mathcal{P}$ is trivial. If $\mathbf{x} \notin \mathcal{R}(M)$ and there exist feasible partial variables, then $\mathcal{P}$ is nontrivial, i.e, we can find a projection $\mathbf{y}'$ of $\mathbf{x}$ such that $[\mathbf{y} + \mathbf{y}'] \in \mathcal{R}(M)$. We assume $\mathbf{x} \notin \mathcal{R}(M)$ and there exist feasible partial variables, such that $\mathcal{P}$ is nontrivial. We say $B_{\text{sub}} \in \mathcal{P}$ if there exists a feasible solution with variable values of indices in $B_{\text{sub}} \subset [n]$. Let $j(n)$ represent the number of elements in $B_{\text{sub}} \subset [n]$. We define the coverage $p(n) := \frac{j(n)}{n}$. If $\mathcal{P}$ is a nontrivial monotone decreasing property, we can define threshold function p_0 of $\mathcal{P}$ as

$$\mathbb{P}(S(n, p) \in \mathcal{P}) \to \begin{cases} 1 & \text{if } p \ll p_0 \\ 0 & \text{if } p \gg p_0 \end{cases} \tag{9}$$

$\mathcal{P}$ is monotone decreasing. By Theorem 3, $\mathcal{P}$ has a threshold function p_0.

LP-relaxation objective satisfiability of partial variables We formulate the property of LP-relaxation objective satisfiability of the partial variables to show there exists a local threshold function in the property. LP-relaxation objective satisfiability of the partial variables refers to the condition in which the sub-MIP formed by a partial variables assignment is LP-feasible and the solution objective value of the LP-relaxation of the sub-MIP satisfies the criterion κ set in advance. Since the objective value of the LP-relaxation solution poses as the lower bound of the objective value of the feasible solution, we set κ to represent the lower bound of the primal bound of the sub-MIP given the learned model. The local threshold function serves as a criterion to form sub-MIPs to generate a feasible solution with an objective value of at least κ, depending on the learned model and the input problem.

Let β_0 be the LP-solution objective value without assigning any discrete variable values from $\mathbf{x}$, i.e., $\beta_0(M) = \min_{\bar{\mathbf{x}} \in \bar{\mathcal{R}}(M)} \mathbf{c}^\top \bar{\mathbf{x}}$. Similarly, let $\beta_\rho(M, \mathbf{x})$ denote the LP-solution objective value after assigning discrete variable values from $\mathbf{x}$ with coverage ρ. Let $\alpha_\rho(M, \mathbf{x}, \tau)$ be a primal bound of a given MIP problem M solving for a time limit of τ, after partially assigning variable values of $\mathbf{x}$ with coverage of ρ. Since $\hat{\mathcal{R}}(M) \subset \mathcal{R}(M)$, it suffices that $\alpha_\rho(M, \mathbf{x}, \tau) \geq \beta_\rho(M, \mathbf{x})$.

Given MIP problem $M = (\mathbf{A}, \mathbf{b}, \mathbf{c})$, $\mathbf{x} \in \mathbb{R}^n$, and $\kappa \in \mathbb{R}$, we define a property $\mathcal{Q}_\kappa$ such that

$$\begin{aligned} \mathcal{Q}_\kappa(\mathbf{x}, \mathbf{A}, \mathbf{b}, \mathbf{c}) := &\{B_{\text{sub}} \subset [n] : \text{ there exists } \mathbf{z}' \in \mathbb{R}^n \\ &\text{subject to } \mathbf{c}^\top [\mathbf{z} + \mathbf{z}'] \geq \kappa \text{ and } \mathbf{A}[\mathbf{z} + \mathbf{z}'] \leq \mathbf{b}\}, \end{aligned} \tag{10}$$

where $\mathbf{z} = \sum_{i \in B_{\text{sub}}} x_i \mathbf{e}_i$ and $\mathbf{z}' = \sum_{i \in [n] \setminus B_{\text{sub}}} z_i \mathbf{e}_i$. If $\mathbf{x} \in \mathcal{R}(M)$ and $\kappa \to -\infty$, then $\mathcal{Q}_\kappa = \mathscr{P}([n])$, i.e, $\mathcal{Q}_\kappa$ is trivial. Suppose the LP-relaxation solution $\bar{\mathbf{x}}$ and the optimal solution $\mathbf{x}^\star$ are given. If $\mathbf{x} \notin \mathcal{R}(M)$ or $\mathbf{c}^\top \bar{\mathbf{x}} < \kappa < \mathbf{c}^\top \mathbf{x}^\star$, and there exist LP-feasible partial variables, then $\mathcal{Q}_\kappa$ is nontrivial. We say $B_{\text{sub}} \in \mathcal{Q}_\kappa$ if the objective value of the LP-solution with the fixed variable values of indices in $B_{\text{sub}} \subset [n]$ is at least κ.

Let $g(n)$ represent the number of elements in $B_{\text{sub}} \subset [n]$. We define the coverage $q(n) := \frac{g(n)}{n}$. By Definition 2, if $\mathcal{Q}_\kappa$ is a nontrivial bounded monotone increasing property, in which the objective criterion is fixed to κ and the bound is $i(n)$. We define the local threshold function $q_{0,\kappa}$ as

$$\mathbb{P}(S(n,q) \in \mathcal{Q}_\kappa) \to \begin{cases} 0 & \text{if } q \ll q_{0,\kappa} \ll i \\ 1 & \text{if } i \gg q \gg q_{0,\kappa} \end{cases} \tag{11}$$

To show there exists a local threshold function in the LP-relaxation objective satisfiability, we prove that the property is bounded monotone increasing. We assume $\mathbf{x} \notin \mathcal{R}(M)$ and there exist feasible partial variables, such that $\mathcal{P}$ is nontrivial. Also, we assume there exist LP-feasible partial variables, such that $\mathcal{Q}_\kappa$ is nontrivial.

If $\mathcal{P}$ is nontrivial and κ is fixed, then $\mathcal{Q}_\kappa$ is bounded monotone increasing. By the assumption, $\mathcal{P}$ is nontrivial. Consequently, $\mathcal{Q}_\kappa$ has a local threshold function by Theorem 4. The local threshold function $q_{0,\kappa}$ is dependent of κ, for which $q_{0,\kappa}$ is monotonic increasing with regard to κ. See Lemma 1 in Appendix A.2 for detail.

4.2 LP-OBJECTIVE SATISFIABILITY AND FEASIBILITY OF PARTIAL VARIABLES

We formulate the property $\mathcal{F}_\kappa$ as an intersection of the property $\mathcal{P}$ and $\mathcal{Q}_\kappa$ in Section 4.1. We show that $\mathcal{F}_\kappa$ has two local thresholds p_a, and p_b, such that $p \in [p_a, p_b]$ implies that $S(n,p)$ satisfies the MIP feasibility and the LP-relaxation objective criterion of partial variables. Thus, the local thresholds form an interval for the coverage of partial variables to generate solutions that are feasible and have a lower bound of quality.

Formally, given that $\mathcal{P}$ is nontrivial monotone decreasing and $\mathcal{Q}_\kappa$ is nontrivial bounded monotone increasing,

$$\mathcal{F}_\kappa(\mathbf{x}, \mathbf{A}, \mathbf{b}, \mathbf{c}) := \mathcal{P} \cap \mathcal{Q}_\kappa, \tag{12}$$

A family of subsets $\mathcal{H}_n$ is convex if $B_a \subseteq B_b \subseteq B_c$ and $B_a, B_c \in \mathcal{H}_n$ imply $B_b \in \mathcal{H}_n$. If we bound the domain of $\mathcal{F}_\kappa$ with p_0, such that $\rho \in [0, p_0]$ in $S(n,\rho)$ for $\mathcal{F}_\kappa$, then $\mathcal{F}_\kappa$ is convex in the domain, since the intersection of an increasing and a decreasing property is a convex property (Janson et al., 2011). Let $h(n)$ represent the number of elements in $B_{\text{sub}} \subset [n]$. We define the coverage $\rho(n) := \frac{h(n)}{n}$.

Theorem 1. *Fix κ. If $q_{0,\kappa} \ll p_0$, then $\mathcal{F}_\kappa$ has an interval of certainty, such that $q_{0,\kappa} \ll \rho \ll p_0$ implies $\mathbb{P}(S(n,\rho) \in \mathcal{F}_\kappa) \to 1$.*

Theorem 1 states there exists an interval enabled by $q_{0,\kappa}$, and p_0, in which the coverage for feasible solutions satisfying the LP-relaxation solution objective criterion belongs.

4.3 THRESHOLD-AWARE LEARNING

Through experiments using the Confidence Thresholding method, we observe that 1) ordering variables by confidence score and 2) controlling the coverage to fix variable values of $\mathbf{x}$ is critical for a fast improvement of the feasible solution quality in a given time limit. Particularly, the main motivation of Threshold-aware learning is dimensionality reduction of discrete variables in MIP by optimizing the coverage of decision variables to fix with theoretical insights. We empirically find that scaling up n, and m shows a consistent trend in threshold points over the coverage domain, as illustrated in Figure 2. In this context, we introduce Threshold-aware learning to optimize the coverage ρ more efficiently than the Confidence Thresholding method. We define Δ-optimal solution $\mathbf{x}_\Delta \in \mathbb{Z}^n$, such that $\mathbf{c}^\top \mathbf{x}_\Delta \leq \mathbf{c}^\top \mathbf{x}^\star + \Delta$. Let Δ-optimal coverage p^*_Δ be the coverage, such that $\alpha_{p^*_\Delta}(M, \mathbf{x}, \tau) \leq \mathbf{c}^\top \mathbf{x}_\Delta$. The candidates for the criterion to decide the Δ-optimal coverage are the

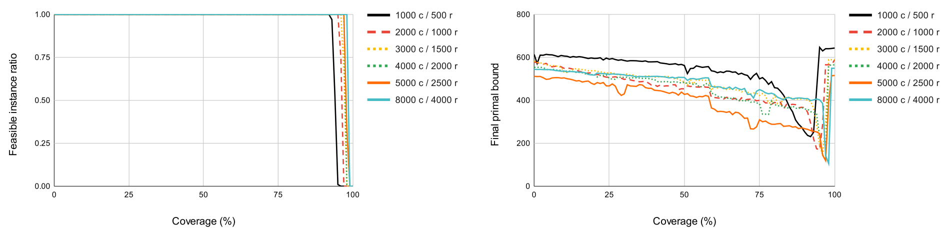

Figure 2: Threshold points of the feasibility and the final primal bound of set covering instances over the discrete variable assignment coverage. We scale the number of discrete variables n, and the number of constraints m, by $1\times$, $2\times$, $3\times$, $4\times$, $5\times$, and $8\times$. We use the same Neural diving model trained on the dataset of the least scale, $n = 1000, m = 500$.

primal bound or primal integral. We use the primal bound as our optimality criterion for training efficiency.

By Proposition 1, $q_{0,\kappa} \ll \rho \ll p_0$ implies $\mathbb{P}(S(n,\rho) \in \mathcal{F}_\kappa) \to 1$. In practice, we deal with MIP problems with a finite and large enough number of variables n. We refer to $p_1(n) \lesssim p_2(n)$ if there exists a positive constant C independent of n, such that $p_1(n) \leq Cp_2(n)$, for $n > 1/\epsilon$. We relax the result of Proposition 1 for large enough n as follows.

Remark 1. $q_{0,\kappa} \lesssim \rho \lesssim p_0$ *implies* $\mathbb{P}(S(n,\rho) \in \mathcal{F}_\kappa) \geq 1 - \epsilon$

We assume $q_{0,\kappa} \lesssim p_0$. Since we do not know the analytical expression of p_0 and $q_{0,\kappa}$, we minimize our proposed threshold-aware loss function to learn p_0 and $q_{0,\kappa}$.

Threshold-aware Loss We propose threshold-aware loss to learn a GNN to find the optimal coverage more efficiently in a restricted search space with a theoretical justification. We model p_0, and $q_{0,\kappa}$ with neural network $p_\psi : M \to \mathbb{R} \in (0,1)$, and $p_\phi : M \to \mathbb{R} \in (0,1)$, respectively. We approximate p^*_Δ with a neural network $\rho_\pi : M \to \mathbb{R} \in (0,1)$, parameterized by π. Note that the neural networks p_ψ, p_ϕ, and ρ_π share the weights of the Neural diving backbone. Let $P_\Psi(p_\psi, \mathcal{P}) : \mathbb{R}^n \to \mathbb{R}$ be a neural network that maps $S(n, p_\psi)$ to $\mathbb{P}(S(n,p_\psi) \in \mathcal{P})$, parameterized by Ψ. Let $P_\Phi(p_\phi, \mathcal{Q}_\kappa) : \mathbb{R}^n \to \mathbb{R}$ be a neural network that maps $S(n,p_\phi)$ to $\mathbb{P}(S(n,p_\phi) \in \mathcal{Q}_\kappa)$, parameterized by Φ. Notice that the neural networks P_Ψ, and P_Φ share the weights of the Neural diving backbone. We learn parameters $\psi, \phi, \pi, \Psi,$ and Φ by minimizing the following threshold-aware loss function. We define

$$\mathbb{1}_{\mathcal{P}}(S(n,p_\psi)) = \begin{cases} 1, & \text{if } S(n,p_\psi) \in \mathcal{P} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Similarly,

$$\mathbb{1}_{\mathcal{Q}_\kappa}(S(n,p_\phi)) = \begin{cases} 1, & \text{if } S(n,p_\phi) \in \mathcal{Q}_\kappa \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Let $\mathrm{BCE}(a,b) = -(a \log b + (1-a)\log(1-b))$ for $a, b \in [0,1]$ be the binary cross entropy function. We implement the loss function using the negative log-likelihood function, where the minimization of the negative log-likelihood is the same as the minimization of binary cross entropy. Let i be the iteration step in the optimization loop. The total loss we aim to minimize is

$$\mathcal{L} = \mathcal{L}_{\text{coverage}} + \mathcal{L}_{\text{threshold}} + \mathcal{L}_{\text{prob}}, \tag{15}$$

where

$$\mathcal{L}_{\text{coverage}}(\rho_\pi, p^*_\Delta; \pi) = \|\rho_\pi - p^*_\Delta\|_2^2, \tag{16}$$

$$\begin{aligned}\mathcal{L}_{\text{threshold}}(p_\psi, p_\phi, &P_\Psi(p_\psi,\mathcal{P}), P_\Phi(p_\phi,\mathcal{Q}_\kappa); \psi, \phi) \\ &= \mathrm{BCE}(P_\Psi(p_\psi^{(i-1)}, \mathcal{P}), p_\psi^{(i)}) + \mathrm{BCE}(P_\Phi(p_\phi^{(i-1)}, \mathcal{Q}_\kappa), p_\phi^{(i)}),\end{aligned} \tag{17}$$

$$\begin{aligned}\mathcal{L}_{\text{prob}}(P_\Psi(p_\psi,\mathcal{P}),&P_\Phi(p_\phi,\mathcal{Q}_\kappa),\mathbb{1}_{\mathcal{P}}(S(n,p_\psi)),\mathbb{1}_{\mathcal{Q}_\kappa}(S(n,p_\phi));\Psi,\Phi)\\ &=\text{BCE}(\mathbb{1}_{\mathcal{P}}(S(n,p_\psi)),P_\Psi(p_\psi,\mathcal{P}))+\text{BCE}(\mathbb{1}_{\mathcal{Q}_\kappa}(S(n,p_\phi)),P_\Phi(p_\phi,\mathcal{Q}_\kappa))\end{aligned} \tag{18}$$

In practice, we obtain

$$p^*_\Delta = \max(\arg\min_{\bar{\rho}_\pi \in [p_\phi,p_\psi]} \alpha_{\bar{\rho_\pi}}(M,\mathbf{x},\tau) \text{ or} \tag{19}$$

$$\min(\arg\min_{\bar{\rho}_\pi \in [p_\phi,p_\psi]} \alpha_{\bar{\rho_\pi}}(M,\mathbf{x},\tau)) \tag{20}$$

By Theorem 3, and 4, there exist $\xi_1,\xi_2 \in (0,1)$, such that $\mathbb{P}(S(n,\xi_1)\in\mathcal{P})=\xi_1$, and $\mathbb{P}(S(n,\xi_2)\in\mathcal{Q}_\kappa)=\xi_2$. Hence, $P_\Psi(p_\psi,\mathcal{P})$, and $P_\Phi(p_\phi,\mathcal{Q}_\kappa)$ in $\mathcal{L}_{\text{threshold}}$, and $\mathcal{L}_{\text{prob}}$ aim to fit ξ_1 and ξ_2, respectively. On the other hand, $P_\Psi(p_\psi,\mathcal{P})$, and $P_\Phi(p_\phi,\mathcal{Q}_\kappa)$ work as smoothing functions for $\mathbb{1}_{\mathcal{P}}(S(n,p_\psi))$, and $\mathbb{1}_{\mathcal{Q}_\kappa}(S(n,p_\phi))$ in practice. Note that the loss function in equation (15) is fully differentiable. To calculate the loss, we integrate MIP solver into the learning framework to obtain $\alpha_{p^*}(M,\mathbf{x},\tau)$, $\mathbb{1}_{\mathcal{P}}(S(n,p_\psi))$ and $\mathbb{1}_{\mathcal{Q}_\kappa}(S(n,p_\phi))$, on-the-fly (see Algorithm 1, and 2 in Appendix A.3).

Proposition 1. *Fix p_ϕ and $P_\Phi(p_\phi,\mathcal{Q}_\kappa)$. If $p_\psi \sim$ Uniform$(0,1)$. then the optimal p_ψ in $\mathcal{L}_{threshold}$ is p_0.*

Let $\kappa^* := \max\kappa$ subject to $\beta_p=\kappa$ and $\mathbb{P}(S(n,p)\in\mathcal{Q}_\kappa)=\xi$ for some $p\in[0,1]$. We assume κ^* is given as ground truth in theory, while we optimize κ in practice (see Algorithm 2 in Appendix A.3).

Corollary 1. *Fix κ^*, p_ψ and $P_\Psi(p_\psi,\mathcal{P})$. If $p_\phi \sim$ Uniform$(0,1)$. then the optimal p_ϕ in $\mathcal{L}_{threshold}$ is q_{0,κ^*}.*

We assume that the solution from the Neural diving model $\mathbf{x}=\mathbf{x}_{\text{partial}}+\mathbf{x}'_{\text{partial}}$, such that the partial solution $\mathbf{x}_{\text{partial}}$ from the Neural diving model output is a partial solution of the Δ-optimal solution $\mathbf{x}_\Delta$, once satisfying certain LP-relaxation objective criterion. Formally, we assume there exists $\tilde{\mathbf{x}}'_{\text{partial}}$, such that $\mathbf{c}^\top[\mathbf{x}_{\text{partial}}+\tilde{\mathbf{x}}'_{\text{partial}}]\le\mathbf{c}^\top\mathbf{x}_\Delta$, and $\mathbf{x}_{\text{partial}}=\sum_{i\in B_{\text{sub}}} x_i\mathbf{e}_i$, for $B_{\text{sub}}\in\mathcal{Q}_{\kappa^*}$. Let $\mathcal{A}$ be the original search space to find the Δ-optimal coverage p^*_Δ. Let t be the complexity to find the optimal point as a function of search space size $|\mathcal{A}|$, such that $|\mathcal{A}|\to t(|\mathcal{A}|)$. It follows that the complexity to find the optimal coverage is $\mathcal{O}(t(|\mathcal{A}|))$. Practically, configuring the optimal setting of the MIP solver in an automated and efficient way is an important task to find higher quality feasible solutions in a shorter time (). In this context, we prove that Threshold-aware learning improves the complexity to find the Δ-optimal coverage p^*_Δ by restricting the search space, such that $p^*_\Delta\in[q_{0,\kappa^*},p_0]$, if the test dataset and the training dataset is i.i.d and the time budget to find the feasible solution is sufficient in the test phase.

Theorem 2. *If $\mathbf{x}=\mathbf{x}_{partial}+\mathbf{x}'_{partial}$, such that there exists $\tilde{\mathbf{x}}'_{partial}:\mathbf{c}[\mathbf{x}_{partial}+\tilde{\mathbf{x}}'_{partial}]\le\mathbf{c}^\top\mathbf{x}_\Delta$, where $\mathbf{x}_{partial}=\sum_{i\in B_{sub}}x_i\mathbf{e}_i$, for $B_{sub}\in\mathcal{F}_{\kappa^*}$, then the complexity of finding the Δ-optimal coverage is $\mathcal{O}(t((p_0-q_{0,\kappa^*})|\mathcal{A}|))$ at the global optimum of $\mathcal{L}_{threshold}$.*

Specifically, we optimize p^*_Δ using derivative-free optimization (DFO), such as Bayesian optimization or ternary search in the search space restricted into $[q_{0,\kappa},p_0]$. Note that searching for the optimal coverage in the restricted search space $[q_{0,\kappa},p_0]$ is more efficient than in the original search space $[0,1]$, by Theorem 2.

The gradient-based training for threshold-aware learning is summarized in Algorithm 1 in Appendix A.3. The loss function in 15 is minimized by iterating over the inner loop of the algorithm. It is sufficient to assume that the loss becomes small enough through iterative updates. Consequently, $p_\psi\approx p_0$, and $p_\phi\approx q_{0,\kappa^*}$, by Proposition 1, and Corollary 1. By Theorem 2 and our assumption, the cardinality of the search space of the ThresholdDFO function in Algorithm 2 in Appendix A.3 to find p^*_Δ becomes $(p_0-q_{0,\kappa^*})|\mathcal{A}|$.

5 Experiments

We evaluate the performance of threshold-aware learning with various problems against other methods. See Appendix A.5.1 for details of the methods used in the experiments.

5.1 Data and evaluation settings

We evaluate the methods in Appendix A.5.1 with the datasets introduced from (Gasse et al., 2022; 2019). Specifically, we choose workload apportionment, maritime inventory routing problem from (Gasse et al., 2022), and set covering, capacitated facility location, and maximum independent set problem from (Gasse et al., 2019). For workload apportionment and maritime inventory routing problems, we set a time limit of 1 minute to evaluate. For set covering, capacitated facility flow, and maximum independent set problems, we solve the problems for 1, 10, and 5 seconds, respectively. We convert the objective of the capacitated facility flow problem into a minimization problem by negating the objective coefficients, where it is originally formulated as a maximization problem in (Gasse et al., 2019). Additionally, we measure the performance of the model trained on the maritime inventory routing dataset augmented with MIPLIB 2017 (Gleixner et al., 2021) to verify the effectiveness of data augmentation.

5.2 Metrics

To evaluate the solution quality over the time dimension, we also use Primal Integral (PI), as proposed by Berthold (2013). Primal integral measures the area between the primal bound and the optimal solution objective value over time.

5.3 Results

Table 1: PI and PB are shorthand for Primal Integral and Primal Bound, respectively. OG and OR are shorthand for Optimality Gap and Optimal instance Rate, respectively.

	Workload Apportionment				Maritime Inventory Routing (non-augmented)				Maritime Inventory Routing (augmented)			
	PI ↓	PB ↓	OG (%) ↓	OR (%) ↑	PI ↓	PB ↓	OG (%) ↓	OR (%) ↑	PI ↓	PB ↓	OG (%) ↓	OR (%) ↑
SCIP (30 hrs)	-	708.31	0.01	98	-	50175.95	0	100	-	50175.95	0	100
SCIP (1 min)	48182.31	738.85	4.36	0	41682758.55	647961.18	305.48	30	41682758.55	647961.18	305.48	30
Neural diving (Nair et al., 2020)	44926.06	713.10	0.69	2	28357794.50	372770.44	143.08	30	30101028.39	322748.95	115.88	30
GNNExplainer (Ying et al., 2019)	47165.63	719.96	1.65	0	29343874.24	369122.56	141.50	20	39004703.50	509108.16	249.95	25
CT (Ours)	44862.77	711.43	0.47	2	24581204.50	202526.50	83.91	**35**	**24795796.07**	306165.57	136.55	30
TaL (Ours)	**44634.85**	**711.34**	**0.45**	**4**	**24288490.56**	**192074.00**	**69.97**	30	33639950.16	**287189.69**	**106.68**	**35**

	Set Covering				Capacitated Facility Flow				Maximum Independent Set			
	PI ↓	PB ↓	OG (%) ↓	OR (%) ↑	PI ↓	PB ↓	OG (%) ↓	OR (%) ↑	PI ↓	PB ↓	OG (%) ↓	OR (%) ↑
SCIP	788.04	606.68	165.96	2	834198.50	43172.72	139.40	0	−1009.56	−218.64	3.55	10
Neural diving (Nair et al., 2020)	493.65	246.12	8.99	0	**312589.87**	18221.34	0.93	**34**	−1062.37	−226.51	0.05	95
CT (Ours)	431.31	233.06	3.13	**5**	319075.00	18214.33	0.91	16	**−1081.87**	−226.44	0.08	87
TaL (Ours)	**411.45**	**232.99**	**3.09**	3	318984.36	**18147.82**	**0.54**	23	−1075.95	**−226.57**	**0.02**	**96**

Table 1 shows that our method outperforms the other methods on all five datasets. In the workload apportionment dataset, Threshold-aware Learning shows a 0.45% optimality gap at the one-minute time limit, which is roughly $10\times$ better than the optimality gap of SCIP. In the maritime inventory routing dataset, the performance of the best method in the data-augmented case is comparable to the best performance in the non-augmented case. It implies the possibility of a general-purpose model trained on a heterogeneous class of MIP instances.

6 Conclusion

In this work, we establish theoretical grounds for optimizing the coverage in learning to generate feasible solutions of MIP from a probabilistic combinatorics perspective. Based on our theoretical insights, we devise a provably efficient learning framework that jointly learns the coverage and the coverage search space by modeling threshold functions. Empirically, our method shows competitive results against other methods. In future work, we would like to extend our learning framework to a broader area of machine learning in a high-dimensional setting. Also, we consider coverage distribution estimation can be further investigated, *i.e.* Beta distribution of our target coverage.

References

Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

Tobias Achterberg, Timo Berthold, and Gregor Hendel. Rounding and propagation heuristics for mixed integer programming. In *Operations research proceedings 2011*, pp. 71–76. Springer, 2012.

Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International conference on machine learning*, pp. 344–353. PMLR, 2018.

Timo Berthold. *Primal heuristics for mixed integer programs*. PhD thesis, Zuse Institute Berlin (ZIB), 2006.

Timo Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6): 611–614, 2013.

Béla Bollobás. Threshold functions for small subgraphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 90, pp. 197–206. Cambridge University Press, 1981.

Béla Bollobás and Arthur G Thomason. Threshold functions. *Combinatorica*, 7(1):35–38, 1987.

Jian-Ya Ding, Chao Zhang, Lei Shen, Shengyin Li, Bing Wang, Yinghui Xu, and Le Song. Accelerating primal solution findings for mixed integer programs based on solution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1452–1459, 2020.

Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Maxime Gasse, Quentin Cappart, Jonas Charfreitag, Laurent Charlin, Didier Chételat, Antonia Chmiela, Justin Dumouchelle, Ambros Gleixner, Aleksandr M Kazachkov, Elias Khalil, et al. The machine learning for combinatorial optimization competition (ml4co): Results and insights. *arXiv preprint arXiv:2203.02433*, 2022.

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International conference on machine learning*, pp. 2151–2159. PMLR, 2019.

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. Miplib 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, 2021.

Prateek Gupta, Maxime Gasse, Elias Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. *Advances in neural information processing systems*, 33: 18087–18097, 2020.

He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27, 2014.

Zeren Huang, Kerong Wang, Furui Liu, Hui-Ling Zhen, Weinan Zhang, Mingxuan Yuan, Jianye Hao, Yong Yu, and Jun Wang. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123:108353, 2022.

Svante Janson, Andrzej Rucinski, and Tomasz Luczak. *Random graphs*. John Wiley & Sons, 2011.

Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33: 6659–6672, 2020.

Elias B Khalil, Bistra Dilkina, George L Nemhauser, Shabbir Ahmed, and Yufen Shao. Learning to run heuristics in tree search. In *Ijcai*, pp. 659–666, 2017.

Elias B Khalil, Christopher Morris, and Andrea Lodi. Mip-gnn: A data-driven framework for guiding combinatorial solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

Dimitri J Papageorgiou, George L Nemhauser, Joel Sokol, Myun-Seok Cheon, and Ahmet B Keha. Mirplib–a library of maritime inventory routing problem instances: Survey, core model, and benchmark results. *European Journal of Operational Research*, 235(2):350–366, 2014.

Anselm Paulus, Michal Rolínek, Vít Musil, Brandon Amos, and Georg Martius. Comboptnet: Fit the right np-hard problem by learning integer programming constraints. In *International Conference on Machine Learning*, pp. 8443–8453. PMLR, 2021.

Meng Qi, Mengxin Wang, and Zuo-Jun Shen. Smart feasibility pump: Reinforcement learning for (mixed) integer programming. *arXiv preprint arXiv:2102.09663*, 2021.

Yunzhuang Shen, Yuan Sun, Andrew Eberhard, and Xiaodong Li. Learning primal heuristics for mixed integer programs. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Jialin Song, Yisong Yue, Bistra Dilkina, et al. A general large neighborhood search framework for solving integer linear programs. *Advances in Neural Information Processing Systems*, 33: 20012–20023, 2020.

Nicolas Sonnerat, Pengming Wang, Ira Ktena, Sergey Bartunov, and Vinod Nair. Learning a large neighborhood search algorithm for mixed integer programs. *arXiv preprint arXiv:2107.10201*, 2021.

Haoran Sun, Wenbo Chen, Hui Li, and Le Song. Improving learning to branch via reinforcement learning. 2020.

Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *International Conference on Machine Learning*, pp. 9367–9376. PMLR, 2020.

Álinson S Xavier, Feng Qiu, and Shabbir Ahmed. Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*, 33(2):739–756, 2021.

Kaan Yilmaz and Neil Yorke-Smith. Learning efficient search approximation in mixed integer branch and bound. *arXiv preprint arXiv:2007.03948*, 2020.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*, 2019.

Giulia Zarpellon, Jason Jo, Andrea Lodi, and Yoshua Bengio. Parameterizing branch-and-bound search trees to learn branching policies. *arXiv preprint arXiv:2002.05120*, 12, 2020.

A Appendix

A.1 Notations

- n: the number of variables
- m: the number of linear constraints
- r: the number of discrete variables
- $\mathbf{x} = [x_1, \ldots, x_n] \in \mathbb{R}^n$: the vector of variable values
- $\mathbf{x}_{\text{int}} \in \mathbb{Z}^r, \mathbf{x}_{\text{cont}} \in \mathbb{R}^{n-r}$ and $\mathbf{x} = [\mathbf{x}_{\text{int}}; \mathbf{x}_{\text{cont}}]$
- $\mathbf{x}^\star$: the vector of optimal solution variable values
- $\mathbf{c} \in \mathbb{R}^n$: the vector of objective coefficients

- $\mathbf{A} \in \mathbb{R}^{m\times n}$: the linear constraint coefficient matrix
- $\mathbf{b} \in \mathbb{R}^{m}$: the vector of linear constraint upper bound
- $(\boldsymbol{\ell}, \boldsymbol{u})$: the vector of lower and upper variable
- $\mathbf{e}_i = \mathbf{e}_i^{(n)} \in \mathbb{R}^n, i = 1, \ldots, n$: the Euclidean standard basis
- $[n] = \{1, 2, \ldots, n\}$: index set
- $M = (\mathbf{A}, \mathbf{b}, \mathbf{c}, \boldsymbol{\ell}, \boldsymbol{u})$: MIP problem
- $\mathcal{I}$: the set of dimensions of $\mathbf{x}$ that is discrete
- x_d: d-th dimension of $\mathbf{x}$
- s_d: binary classifier output to decide to fix x_d
- p_θ: Neural diving model parameterized by θ
- N_p: the number of MIP problems in the dataset
- Γ: threshold value for Post hoc Confidence Thresholding
- $\mathcal{R}(M)$: the set of feasible solution $\mathbf{x} \in \mathbb{Z}^r \times \mathbb{R}^{n-r}$
- $\bar{\mathcal{R}}(M)$: the set of LP-feasible solution $\mathbf{x} \in \mathbb{R}^n$ which satisfies $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ ignoring the integrality constraints
- $\hat{\mathcal{R}}(M)$: the set of LP-feasible solution $\mathbf{x} \in \mathbb{Z}^r \times \mathbb{R}^{n-r}$ which satisfies $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ after partially fixing discrete variables of $\mathbf{x}$
- $\xi \in (0, 1)$: probability at which threshold behavior occurs
- $\mathscr{P}(X)$: powerset of X
- $a \ll b$ or $a(n) \ll b(n)$: $\lim\limits_{n\to\infty} a(n)/b(n) = 0$
- $a \gg b$ or $a(n) \gg b(n)$: $\lim\limits_{n\to\infty} a(n)/b(n) = \infty$
- $a \lesssim b : a(n) \lesssim b(n)$: there exists $C > 0$, s. t. $a(n) \leq Cb(n)$, where $n \in \mathcal{N}$, for some set $\mathcal{N}$
- κ: target objective value of the LP-solution for partial discrete variables fixed
- $\beta_0(M) = \min \mathbf{c}^\top \bar{\mathcal{R}}(M)$: the LP-solution objective value without assigning any discrete variable values from $\mathbf{x}$
- $\beta_p(M, \mathbf{x}) = \mathbf{c}^\top[\mathbf{z} + \mathbf{z}']$, where $\mathbf{z} = \sum\limits_{i\in S(B,p)} x_i \mathbf{e}_i \in \mathbb{Z}^n$, and $\mathbf{z}' = \sum\limits_{i\in[n]\setminus S(B,p)} z_i \mathbf{e}_i \in \mathbb{R}^n$: the LP-solution objective value after assigning discrete variable values from $\mathbf{x}$ with coverage p
- $\alpha_p(M, \mathbf{x}, \tau)$: a primal bound of a given MIP problem M solving for a time limit of τ, after partially assigning variable values of $\mathbf{x}$ with coverage of p.

A.2 Statements and proofs

Theorem 3. *Let $\xi \in (0, 1)$. For each $n \in \mathbb{N}$, let $\mathcal{H}_n$ be a monotone increasing non-trivial property and set*

$$m_*(n) := \max\left\{m : \mathbb{P}(\mathcal{H}_n|\mathcal{B}_m) \leq \xi\right\}. \tag{21}$$

If $m \leq m_*(n)$, then

$$\mathbb{P}(\mathcal{H}_n|\mathcal{B}_m) \leq 1 - \xi^{\frac{m}{m_*(n)}} \tag{22}$$

and, if $m \geq m_*(n) + 1$, then

$$\mathbb{P}(\mathcal{H}_n|\mathcal{B}_m) \geq 1 - \xi^{\frac{m}{m_*(n)+1}}. \tag{23}$$

In particular, $m_*(n)$ is a threshold function of $\mathcal{H}_n$.

Proof. We slightly modify the proof in (Bollobás & Thomason, 1987, Theorem 4) for rigorousness. Let $\mathcal{J}_n$ denote the negation of $\mathcal{H}_n$.Note that $\mathcal{J}_n$ is a nontrivial monotone decreasing property.

If $m \leq m_*(n)$, then

$$\mathbb{P}(\mathcal{J}_n|\mathcal{B}_m) \geq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_{m_*(n)}) \geq \xi^{\frac{m}{m_*(n)}}$$

If $m \geq m_*(n)+1$, then

$$\mathbb{P}(\mathcal{J}_n|\mathcal{B}_m) \leq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_{m_*(n)+1}) \leq \xi^{\frac{m}{m_*(n)+1}}$$

□

From Theorem 3, we have the following statement.

Corollary 2. *Let $\xi \in (0,1)$. For each $n \in \mathbb{N}$, let $\mathcal{H}_n$ be a monotone decreasing nontrivial property.*

$$m_*(n) := \min\{m : \mathbb{P}(\mathcal{H}_n|\mathcal{B}_m) \geq \xi\}. \tag{24}$$

If $m \leq m_*(n)$, then

$$\mathbb{P}(\mathcal{H}_n|\mathcal{B}_m) \geq 1 - \xi^{\frac{m}{m_*(n)}} \tag{25}$$

If $m \geq m_*(n)+1$, then

$$\mathbb{P}(\mathcal{H}_n|\mathcal{B}_m) \leq 1 - \xi^{\frac{m}{m_*(n)+1}} \tag{26}$$

In particular, $m_*(n)$ is a threshold function of $\mathcal{H}_n$.

Proof. Let $\mathcal{J}_n$ denote the negation of $\mathcal{H}_n$.Note that $\mathcal{J}_n$ is a nontrivial monotone increasing property.

If $m \leq m_*(n)$, then

$$\mathbb{P}(\mathcal{J}_n|\mathcal{B}_m) \leq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_{m_*(n)}) \leq \xi^{\frac{m}{m_*(n)}}$$

If $m \geq m_*(n)+1$, then

$$\mathbb{P}(\mathcal{J}_n|\mathcal{B}_m) \geq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_{m_*(n)+1}) \geq \xi^{\frac{m}{m_*(n)+1}}$$

□

Lemma 1. *If $\mathcal{P}$ is nontrivial, then $q_{0,\kappa}$ is monotonic increasing with regard to κ, bounded by p_0.*

Proof. By Lemma 3 and Theorem 4, $\mathcal{Q}_\kappa$ is bounded monotone increasing and has a local threshold function $q_{0,\kappa}$, where κ is fixed. We choose the constant $\xi \in (0,1)$ as in Theorem 3, such that $\mathbb{P}(S(n, q_{0,\kappa}) \in \mathcal{Q}_\kappa) = \xi$. We choose p_0 as a bound of $\mathcal{Q}_\kappa$ as in Lemma 3. Suppose $\kappa_1 \leq \kappa_2$, such that $\mathbf{c}^\top \bar{\mathbf{x}} < \kappa_1 \leq \kappa \leq \kappa_2 < \mathbf{c}^\top \mathbf{x}^\star$. If follows that

$$\mathbb{P}(S(n, q_{0,\kappa}) \in \mathcal{Q}_{\kappa_1}) \geq \mathbb{P}(S(n, q_{0,\kappa}) \in \mathcal{Q}_\kappa) \geq \mathbb{P}(S(n, q_{0,\kappa}) \in \mathcal{Q}_{\kappa_2}) \tag{27}$$

By Definition 2 and Theorem 4,

$$\mathbb{P}(S(n, q_{0,\kappa_1}) \in \mathcal{Q}_{\kappa_1}) = \mathbb{P}(S(n, q_{0,\kappa}) \in \mathcal{Q}_\kappa) = \mathbb{P}(S(n, q_{0,\kappa_2}) \in \mathcal{Q}_{\kappa_2}) = \xi \tag{28}$$

Hence,

$$q_{0,\kappa_1} \leq q_{0,\kappa} \leq q_{0,\kappa_2} \ll p_0 \tag{29}$$

□

Theorem 4. *Let $\xi \in (0,1)$. For each $n \in \mathbb{N}$, let $\mathcal{H}_n$ be a non-trivial bounded monotone increasing property, where the bound is i.*

$$l_*(n) := \max\{l : \mathbb{P}(\mathcal{H}_n|\mathcal{B}_l) \leq \xi\}. \tag{30}$$

If $l \geq i+1$, then it is of limited interest since $\mathcal{H}_n$ is non-monotone in such domain.

If $l \leq l_*(n) \leq i$, then

$$\mathbb{P}(\mathcal{H}_n|\mathcal{B}_l) \leq 1 - \xi^{\frac{l}{l_*(n)}} \text{ and } \mathbb{P}(\mathcal{H}_n|\mathcal{B}_{l_*(n)}) \leq 1 - \xi^{\frac{l_*(n)}{i}} \tag{31}$$

If $i \geq l \geq l_*(n)+1$, then

$$\mathbb{P}(\mathcal{H}_n|\mathcal{B}_l) \geq 1 - \xi^{\frac{l}{l_*(n)+1}} \text{ and } \mathbb{P}(\mathcal{H}_n|\mathcal{B}_i) \geq 1 - \xi^{\frac{i}{l}} \tag{32}$$

In particular, l_* is a local threshold function of $\mathcal{H}_n$ bounded by i.

Proof. We adapt the proof in (Bollobás & Thomason, 1987, Theorem 4) for Definition 2. Let $\mathcal{J}_n$ denote the negation of $\mathcal{H}_n$.Note that $\mathcal{J}_n$ is a nontrivial bounded monotone decreasing property.

If $l \leq l_*(n) \leq i$, then

$$\mathbb{P}(\mathcal{J}_n|\mathcal{B}_l) \geq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_{l_*(n)}) \geq \xi^{\frac{l}{l_*(n)}} \text{ and } \mathbb{P}(\mathcal{J}_n|\mathcal{B}_{l_*(n)}) \geq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_i) \geq \xi^{\frac{l_*(n)}{i}}$$

If $i \geq l \geq l_*(n)+1$, then

$$\mathbb{P}(\mathcal{J}_n|\mathcal{B}_l) \leq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_{l_*(n)+1}) \leq \xi^{\frac{l}{l_*(n)+1}} \text{ and } \mathbb{P}(\mathcal{J}_n|\mathcal{B}_i) \leq \mathbb{P}(\mathcal{J}_n|\mathcal{B}_l) \leq \xi^{\frac{i}{l}}$$

□

Lemma 2. *$\mathcal{P}$ is monotone decreasing.*

Proof. Let B_s be a set, such that $B_s \subset B$. Let B_{ss} be a set, such that $B_{ss} \subset B_s$. By definition, if $B_s \in \mathcal{P}$, then there exists $\mathbf{z}'_s$, such that $\mathbf{A}\left[\mathbf{z}_s + \mathbf{z}'_s\right] \leq \mathbf{b}$, where $\mathbf{z}_s = \sum_{i \in B_s} x_i \mathbf{e}_i$, and $\mathbf{z}'_s = \sum_{i \in [n] \setminus B_s} z_i \mathbf{e}_i$, given $\mathbf{x}$. Let $\mathbf{z}_{ss} = \sum_{i \in B_{ss}} x_i \mathbf{e}_i$, and $\mathbf{z}'_{ss} = \sum_{i \in [n] \setminus B_{ss}} z_i \mathbf{e}_i$ for $B_{ss} \subset B_s$. We can choose $\mathbf{z}'_{ss}$, such that $\mathbf{z}'_{ss} = \mathbf{z}'_s + \sum_{i \in B_s \setminus B_{ss}} x_i \mathbf{e}_i$, where $\sum_{i \in B_s \setminus B_{ss}} x_i \mathbf{e}_i = \mathbf{z}_s - \sum_{i \in B_{ss}} x_i \mathbf{e}_i$. It follows that there exists $\mathbf{z}'_{ss}$, such that $\mathbf{A}[\mathbf{z}_{ss} + \mathbf{z}'_{ss}] = \mathbf{A}\left[\mathbf{z}_s + \mathbf{z}'_s\right] \leq \mathbf{b}$. Hence, $B_s \in \mathcal{P}$ implies $B_{ss} \in \mathcal{P}$. □

Lemma 3. *Fix κ. If $\mathcal{P}$ is nontrivial, then $\mathcal{Q}_\kappa$ is bounded monotone increasing.*

Proof. $\mathcal{P}$ has a threshold function p_0 by Theorem 3 and Lemma 2. We choose p_0 for the bound of $\mathcal{Q}_\kappa$. Therefore, we inspect only the domain of interest $q \in [0, p_0]$ in $S(n, q)$ for $\mathcal{Q}_\kappa$. Let $\mathbf{u}_p = \sum_{i \in B_p} x_i \mathbf{e}_i \in \mathbb{Z}^n$ for $B_p \in \mathcal{P}$ and $B_p = S(B, p_0)$. Given $B_p \in \mathcal{P}$, there exists $\mathbf{u}'_p = \sum_{i \in [n] \setminus B_p} u_i \mathbf{e}_i \in \mathbb{R}^n$, such that $\mathbf{A}[\mathbf{u}_p + \mathbf{u}'_p] \leq \mathbf{b}$, since $\mathbb{Z}^n \subset \mathbb{R}^n$.

Let B_s be a set, such that $B_s \subset B_p$. Let B_{ss} be a set, such that $B_{ss} \subset B_s$. By definition, $B_{ss} \in \mathcal{Q}_\kappa$ implies there exists $\mathbf{u}'_{ss} = \sum_{i \in [n] \setminus B_{ss}} u_i \mathbf{e}_i \in \mathbb{R}^n$, such that $\mathbf{A}\left[\mathbf{u}_{ss} + \mathbf{u}'_{ss}\right] \leq \mathbf{b}$, and $\mathbf{c}^\top\left[\mathbf{u}_{ss} + \mathbf{u}'_{ss}\right] \geq \kappa$ where $\mathbf{u}_{ss} = \sum_{i \in B_{ss}} x_i \mathbf{e}_i \in \mathbb{Z}^n$, given $\mathbf{x}$. Let $\mathbf{u}_s = \sum_{i \in B_s} x_i \mathbf{e}_i \in \mathbb{Z}^n$, and $\mathbf{u}'_s = \sum_{i \in [n] \setminus B_s} u_i \mathbf{e}_i \in \mathbb{R}^n$. We show that 1) $B_{ss} \in \mathcal{Q}_\kappa$ implies $\mathbf{A}[\mathbf{u}_s + \mathbf{u}'_s] \leq \mathbf{b}$, and 2) $B_{ss} \in \mathcal{Q}_\kappa$ implies $\mathbf{c}^\top[\mathbf{u}_s + \mathbf{u}'_s] \geq \kappa$.

1) Let $\mathbf{u}_{s-ss} = \sum_{i \in B_s \setminus B_{ss}} x_i \mathbf{e}_i \in \mathbb{Z}^n$. Let $\mathbf{u}'_p = \sum_{i \in [n] \setminus B_p} u_i \mathbf{e}_i \in \mathbb{R}^n$. For any $B_s \supset B_{ss}$, $B_s \setminus B_{ss} \in \mathcal{P}$. Also, we can find $\mathbf{u}'_s = \mathbf{u}_{p-s} + \mathbf{u}'_p$, such that $\mathbf{A}[\mathbf{u}_{ss} + \mathbf{u}_{s-ss} + \mathbf{u}_{p-s} + \mathbf{u}'_p] = \mathbf{A}[\mathbf{u}_s + \mathbf{u}'_s] = \mathbf{A}[\mathbf{u}_p + \mathbf{u}'_p] \leq \mathbf{b}$, since $\mathbb{Z}^n \subset \mathbb{R}^n$. Therefore, $\mathbf{A}[\mathbf{u}_s + \mathbf{u}'_s] \leq \mathbf{b}$.

2) Let $\hat{\mathcal{R}}(M)$ be the set of LP-feasible solution $\hat{\mathbf{u}}$, after fixing the nontrivial number of discrete variables of M. Let $\bar{\mathcal{R}}(M)$ be the set of LP-feasible solution $\bar{\mathbf{u}}$, without fixing variables of M. Since $\mathcal{R}(M) \subseteq \hat{\mathcal{R}}(M) \subseteq \bar{\mathcal{R}}(M)$, $\min_{\bar{\mathbf{u}} \in \bar{\mathcal{R}}(M)} \mathbf{c}^\top \bar{\mathbf{u}} \leq \min_{\hat{\mathbf{u}} \in \hat{\mathcal{R}}(M)} \mathbf{c}^\top \hat{\mathbf{u}} \leq \min_{\mathbf{u} \in \mathcal{R}(M)} \mathbf{c}^\top \mathbf{u}$. Therefore, for any $B_s \supset B_{ss}$, $\mathbf{c}^\top\left[\mathbf{u}_s + \mathbf{u}'_s\right] = \mathbf{c}^\top\left[\mathbf{u}_{ss} + \mathbf{u}_{s-ss} + \mathbf{u}'_s\right] \geq \mathbf{c}^\top\left[\mathbf{u}_{ss} + \mathbf{u}'_{ss}\right] \geq \kappa$.

By 1) and 2), $B_{ss} \in \mathcal{Q}_\kappa$ implies $B_s \in \mathcal{Q}_\kappa$. Hence, $\mathcal{Q}_\kappa$ is bounded monotone increasing, in which the bound is at most p_0, such that $q \in [0, p_0]$ in $S(B, q)$ for $\mathcal{Q}_\kappa$. □

Proof of Theorem 1

Proof. By Lemma 2 and 3, there exist p_0 and q_0 such that

$$\mathbb{P}(S(n,\rho) \in \mathcal{P}) \to \begin{cases} 1 & \text{if } \rho \ll p_0 \\ 0 & \text{if } \rho \gg p_0 \end{cases}$$

and

$$\mathbb{P}(S(n,\rho) \in \mathcal{Q}_\kappa) \to \begin{cases} 0 & \text{if } \rho \ll q_{0,\kappa} \ll p_0 \\ 1 & \text{if } p_0 \gg \rho \gg q_{0,\kappa} \end{cases}$$

Since

$$\mathbb{P}(S(n,\rho) \in \mathcal{F}_\kappa) \geq \mathbb{P}(S(n,\rho) \in \mathcal{P}) + \mathbb{P}(S(n,\rho) \in \mathcal{Q}_\kappa) - 1 \tag{33}$$

By equation (33),

$$q_{0,\kappa} \ll \rho \ll p_0 \text{ implies } \mathbb{P}(S(n,\rho) \in \mathcal{F}_\kappa) \to 1 \tag{34}$$

□

Proof of Proposition 1

Proof. $P_\Psi(p_\psi, \mathcal{P}), P_\Phi(p_\phi, \mathcal{Q}_\kappa)$ By Theorem 3 and Lemma 2, $\mathcal{P}$ has a threshold function p_0. For simplicity of notation, we denote the sample $X = \mathbb{1}_{\mathcal{P}}(S(n, p_\psi))$. Suppose there exist H samples. For p_ϕ and $P_\Phi(p_\phi, \mathcal{Q}_\kappa)$ fixed, the training of $\mathcal{L}_{\text{prob}}$ is to maximize

$$\sum_{i=1}^{H} X_i \log P_\Psi(p_\psi, \mathcal{P}) + \sum_{i=1}^{H} (1 - X_i) \log(1 - P_\Phi(p_\phi, \mathcal{Q}_\kappa)) \tag{35}$$

For any $(a, b) \in \mathbb{R}^2 \setminus (0, 0)$, let $Z : Y \to a \log(Y) + b \log(1 - Y)$. Then $\arg\max Z(Y) = \frac{a}{a+b}$. Hence,

$$\arg\min \mathcal{L}_{\text{prob}} = \frac{1}{H} \sum_{i=1}^{H} X_i \tag{36}$$

for $P_\Phi(p_\phi, \mathcal{Q}_\kappa)$ fixed. Similarly, the training of $\mathcal{L}_{\text{threshold}}$ is to maximize $\sum\limits_{i=1}^{H} P_\Psi^{(i)}(p_\psi, \mathcal{P}) \log p_\psi + \sum\limits_{i=1}^{H} (1 - P_\Psi^{(i)}(p_\psi, \mathcal{P})) \log(1 - p_\psi)$, where $P_\Psi^{(i)}(p_\psi, \mathcal{P})$ corresponds to $P_\Psi(p_\psi, \mathcal{P})$ at i-th iteration for X_i.

$$\arg\min \mathcal{L}_{\text{threshold}} = \frac{1}{H} \sum_{i=1}^{H} P_\Psi^{(i)}(p_\psi, \mathcal{P}) \tag{37}$$

At the global minima of $\mathcal{L}_{\text{threshold}}$, and $\mathcal{L}_{\text{prob}}$,

$$P_\Psi(p_\psi, \mathcal{P}) = \frac{1}{H} \sum_{i=1}^{H} X_i \tag{38}$$

and

$$p_\psi = \frac{1}{H} \sum_{i=1}^{H} P_\Psi^{(i)}(p_\psi, \mathcal{P}) \approx \frac{1}{H} \sum_{i=1}^{H} X_i, \tag{39}$$

for large enough H. Let $\xi = \mathbb{P}(S(n, p_\psi) \in \mathcal{P}) = \frac{1}{H} \sum\limits_{i=1}^{H} X_i = \arg\min \mathcal{L}_{\text{prob}}$. Choose $p_0 = \xi$. □

Proof of Corollary 1

Proof. By Lemma 3, and Theorem 4, $\mathcal{Q}_\kappa$ has a local threshold function $q_{0,\kappa}$. Immediately from the proof of Proposition 1, choose $q_{0,\kappa^*} = \xi = \frac{1}{H} \sum\limits_{i=1}^{H} X_i$. □

Proof of Theorem 2

Proof. First, we prove the following statement:

$$\text{If } p_\triangle^* = \arg\min_p \alpha_p(M, \mathbf{x}, \tau), \text{ then } q_{0,\kappa^*} \lesssim p_\triangle^* \lesssim p_0 \tag{40}$$

under the condition:

$$\begin{aligned} &\mathbf{x} = \mathbf{x}_{\text{partial}} + \mathbf{x}'_{\text{partial}}, \\ \text{such that } &\text{there exists } \tilde{\mathbf{x}}'_{\text{partial}} : \mathbf{c}^\top[\mathbf{x}_{\text{partial}} + \tilde{\mathbf{x}}'_{\text{partial}}] \le \mathbf{c}^\top \mathbf{x}_\triangle, \\ \text{where } &\mathbf{x}_{\text{partial}} = \sum_{i \in B_{\text{sub}}} x_i \mathbf{e}_i, \text{ for } B_{\text{sub}} \in \mathcal{F}_{\kappa^*} \end{aligned} \tag{41}$$

To obtain a contradiction, suppose $p_\triangle^* = \max(\arg\min_p \alpha_p(M, \mathbf{x}, \tau))$. Suppose also $p_\triangle^* \lesssim q_{0,\kappa^*}$ or $p_\triangle^* \gtrsim p_0$.

1) $p_\triangle^* \gtrsim p_0$ implies $S(n, p_\triangle^*) \notin \mathcal{P}$ with high probability. This contradicts $p_\triangle^* = \max(\arg\min_p \alpha_p(M, \mathbf{x}, \tau))$.

2) $p_\triangle^* \lesssim q_{0,\kappa^*}$ implies $S(n, p_\triangle^*) \notin \mathcal{Q}_{\kappa^*}$ with high probability. Since $q_{0,\kappa^*} \lesssim p_0$, $S(n, p_\triangle^*) \in \mathcal{P}$ with high probability. By condition in (41), there exists $p' : S(n, p') \in \mathcal{Q}_{\kappa^*}$ and $\alpha_{p'}(M, \mathbf{x}, \tau) \le \alpha_{p_\triangle^*}(M, \mathbf{x}, \tau)$, and $p' > p_\triangle^*$. This contradicts $p_\triangle^* = \max(\arg\min_p \alpha_p(M, \mathbf{x}, \tau))$.

Hence, (40) is true under the condition (41). By Proposition 1, $\arg\min_{p_\psi} \mathcal{L}_{\text{threshold}} = p_0$, with p_ϕ and $P_\Phi(p_\phi, \mathcal{Q}_{\kappa^*})$ fixed. By Corollary 1, $\arg\min_{p_\phi} \mathcal{L}_{\text{threshold}} = q_{0,\kappa^*}$, with κ^*, p_ψ and $P_\Psi(p_\psi, \mathcal{P})$ fixed. By Remark 1 for large enough n,

$$q_{0,\kappa^*} \lesssim \rho \lesssim p_0 \text{ implies } \mathbb{P}(S(n, \rho) \in \mathcal{F}_{\kappa^*}) \ge 1 - \epsilon \tag{42}$$

Thus, $S(n, p_\triangle^*) \in \mathcal{F}_{\kappa^*}$ with high probability, for $p_\triangle^* \in [q_{0,\kappa^*}, p_0]$. Therefore, we search $p_\triangle^* \in [q_{0,\kappa^*}, p_0]$ at the global optimum of $\mathcal{L}_{\text{threshold}}$, such that the cardinality of the search space becomes $(p_0 - q_{0,\kappa^*})|\mathcal{A}|$. □

A.3 ALGORITHMS

Algorithm 1 ThresholdAwareLearning

```
 1: procedure THRESHOLDAWARELEARNING($M$)
 2:     Batch size $H$, the number of iterations $T$
 3:     ThresholdAwareGNN parameterized by $\psi, \phi, \pi$.
 4:     ThresholdAwareProbGNN parameterized by $\Psi, \Phi$.
 5:     $i \leftarrow 0$
 6:     for $i \le T$ do
 7:         for $j \le H$ do                                              ▷ in parallel
 8:             $\mathbf{x}, p_\psi, p_\phi, \rho_\pi \leftarrow$ ThresholdAwareGNN($M$)
 9:             $P_\Psi(p_\psi, \mathcal{P}), P_\Phi(p_\phi, \mathcal{Q}_\kappa) \leftarrow$ ThresholdAwareProbGNN($M, p_\psi, p_\phi, \rho_\pi$)
10:             $\mathbb{1}_{\mathcal{P}}(S(n, p_\psi)), \mathbb{1}_{\mathcal{Q}_\kappa}(S(n, p_\phi)), p_\triangle^* \leftarrow$ ThresholdSolve($M, \mathbf{x}, p_\psi, p_\phi, \rho_\pi$)
11:             if $p_\triangle^*$ is Null then break
12:             Compute $\mathcal{L}_{\text{coverage}}, \mathcal{L}_{\text{threshold}}, \mathcal{L}_{\text{prob}}$ by eq. equation 16, equation 17, equation 18
13:             $\boldsymbol{g}_{\text{coverage}}, \boldsymbol{g}_{\text{threshold}}, \boldsymbol{g}_{\text{prob}} \leftarrow \nabla_\pi \mathcal{L}_{\text{coverage}}, \nabla_{\psi,\phi} \mathcal{L}_{\text{threshold}}, \nabla_{\Psi,\Phi} \mathcal{L}_{\text{prob}}$
14:             Update $\pi, (\psi, \phi), (\Psi, \Phi)$ by gradient descent with $\boldsymbol{g}_{\text{coverage}}, \boldsymbol{g}_{\text{threshold}}, \boldsymbol{g}_{\text{prob}}$
15:             $j \leftarrow j + 1$
16:         $i \leftarrow i + 1$
        return ThresholdAwareGNN
```

Algorithm 2 ThresholdSolve

```
1: procedure ThresholdSolve($M, \mathbf{x}, p_\psi, p_\phi, \rho_\pi$)
2:     $\beta_0(M) \leftarrow$ SolveLP($M, \mathbf{x}, 0$)
3:     $\beta_{\rho_\pi}(M, \mathbf{x}) \leftarrow$ SolveLP($M, \mathbf{x}, \rho_\pi$)
4:     $\alpha_{\rho_\pi}(M, \mathbf{x}, \tau) \leftarrow$ SolveMIP($M, \mathbf{x}, \rho_\pi, \tau$)
5:     if $S(n, p_\psi) \in \mathcal{P}$ then
6:         $\mathbb{1}_\mathcal{P}(S(n, p_\psi)) \leftarrow 1$
7:         $\alpha_{p_\psi}(M, \mathbf{x}, \tau) \leftarrow$ SolveMIP($M, \mathbf{x}, p_\psi, \tau$)
8:     else
9:         $\mathbb{1}_\mathcal{P}(S(n, p_\psi)) \leftarrow 0$
10:        $\alpha_{p_\psi}(M, \mathbf{x}, \tau) \leftarrow Null$
11:    if $S(n, p_\phi) \in \mathcal{P}$ then
12:        $\alpha_{p_\phi}(M, \mathbf{x}, \tau) \leftarrow$ SolveMIP($M, \mathbf{x}, p_\phi, \tau$)
13:        $\kappa \leftarrow p_\phi \cdot (\alpha_{p_\phi}(M, \mathbf{x}, \tau) - \beta_0(M)) + \beta_0(M)$
14:        if $\beta_{\rho_\pi}(M, \mathbf{x}) \geq \kappa$ then
15:            $\mathbb{1}_{\mathcal{Q}_\kappa}(S(n, p_\phi)) \leftarrow 1$
16:        else
17:            $\mathbb{1}_{\mathcal{Q}_\kappa}(S(n, p_\phi)) \leftarrow 0$
18:    else
19:        $\mathbb{1}_{\mathcal{Q}_\kappa}(S(n, p_\phi)) \leftarrow 0$
20:        $\alpha_{p_\phi}(M, \mathbf{x}, \tau) \leftarrow Null$
21:    if $S(n, \min(p_\psi, p_\phi)) \in \mathcal{P}$ then
22:        $p^*_\Delta \leftarrow$ ThresholdDFO($p_\psi, p_\phi, \mathbf{x}, \alpha_{p_\psi}(M, \mathbf{x}, \tau), \alpha_{p_\phi}(M, \mathbf{x}, \tau)$)
23:    else
24:        $p^*_\Delta \leftarrow Null$
       return $\mathbb{1}_\mathcal{P}(S(n, p_\psi)), \mathbb{1}_{\mathcal{Q}_\kappa}(S(n, p_\phi)), p^*$
```

Solving LP-relaxation and sub-MIP using MIP solver SolveLP($M, \mathbf{x}, 0$) refers to the process of solving LP-relaxation of M to return the objective value of the solution. SolveLP($M, \mathbf{x}, \rho_\pi$) is the process of solving LP-relaxation of the sub-MIP of M by fixing variables with coverage ρ_π and values $\mathbf{x}$. Since the objective term is unbounded, we set $\kappa \in (\beta_0(M), \alpha_\rho(M, \mathbf{x}, \tau))$, and calibrate κ to κ^*.

A.4 Neural network architecture

We model the thresholds p_ψ, p_ϕ and the coverage ρ_π by constructing a multipartite graph representation of threshold-related nodes along with a coverage node on top of the Neural diving backbone model. Multipartite graph representation enables scalable representation learning by allowing a flexible number of input variable nodes, while MLP requires a fixed size of the input.

A.5 Experimental details

A.5.1 Experimental methods

Threshold-aware Learning To optimize the loss function in equation (15) in a scalable and structured manner, Threshold-aware Learning (TaL) extends the existing bipartite graph representation of MIP (Gasse et al., 2019; Nair et al., 2020; Khalil et al., 2022) into a multipartite graph representation to jointly learn threshold-related parameters ψ, ϕ, and the coverage parameter f. As shown in Figure 3, we use the GNN architecture from (Gasse et al., 2019) to pre-train the Neural diving model without Selectivenet, denoted by constraint and variable nodes. After training the Neural diving model, we build the multipartite graph representation on top of the Neural diving GNN by constructing confidence nodes based on the confidence scores $|p_\theta(\mathbf{x}|M) - \mathbf{1}/2|$ of the Neural diving model output $p_\theta(\mathbf{x}|M)$, where $\mathbf{1} \in \mathbb{Z}^n$ is the vector of 1's. The output of the threshold-aware GNN are p_ψ, p_ϕ, and f parameterized by ψ, ϕ, and π, respectively. In the training phase, we freeze the weights of Neural diving GNN and update weights of MLP layers of node embedders of confidence nodes, threshold nodes, and a coverage node, along with edge embedders connecting them. For or-

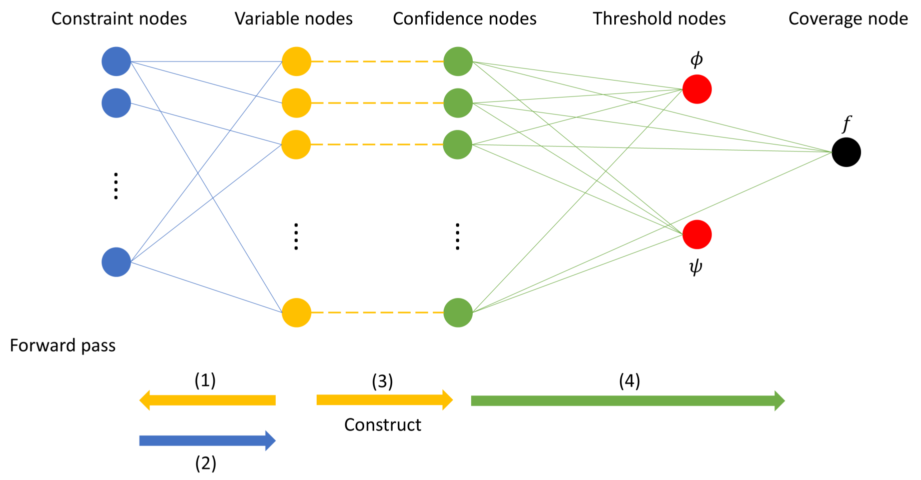

Figure 3: Forward pass of the multipartite graph representation.

dering of variables to fix, one may consider input order, objective coefficient order, and fractionality order as suggested in (Nair et al., 2020).

Post-hoc Confidence Thresholding As a motivation point of threshold aware learning, Post-hoc Confidence Thresholding (CT) method shows that the confidence score is a competitive measure for ordering variables. We build on top of the Neural diving model (Nair et al., 2020) without Selectivenet (Geifman & El-Yaniv, 2019) to control the coverage for partial variable assignment by using a post-hoc technique. We set a threshold on the confidence score $|p_\theta(\mathbf{x}|M) - \mathbf{1}/2|$ of the Neural diving model output $p_\theta(\mathbf{x}|M)$. If the confidence score of a certain variable exceeds the threshold, then we assign the rounded value of the corresponding Neural diving output to the variable value.

Neural diving To jointly learn the coverage and variable assignment values, Neural diving (Nair et al., 2020) adopts Selectivenet (Geifman & El-Yaniv, 2019). We train multiple models for each coverage rate whose value is close to the coverage rate of the best result of other methods. The difference from (Nair et al., 2020) is that we use L_1 norm in place of L_2 norm for the coverage penalty term to allow fluctuations around the target coverage to obtain more diverse coverage output across neural network weight checkpoints. Since Selectivenet learns to abstain from predicting variable values, it is not necessary to consider the ordering of variables.

GNNExplainer GNNExplainer (Ying et al., 2019) is a post-hoc interpretability method for any GNN predictions. The idea of GNNExplainer is to find a simplified model that maximizes the mutual information between the prediction of the original model and the simplified model. We adapt GNNExplainer to the Neural diving setting, where the objective is to find the proper subset of predicted variables.