Journal of Artificial Intelligence Research 80 (2024) 327-376 Submitted 05/2023; published 06/2024

Scalable Primal Heuristics Using Graph Neural Networks for Combinatorial Optimization

Furkan Cantürk FURKAN.CANTURK@OZU.EDU.TR
Artificial Intelligence and Data Engineering, Özyeğin University, Istanbul, Türkiye
Taha Varol TAHA.VAROL@OZU.EDU.TR
Industrial Engineering, Özyeğin University, Istanbul, Türkiye
Reyhan Aydoğan REYHAN.AYDOGAN@OZYEGIN.EDU.TR
Artificial Intelligence and Data Engineering, Özyeğin University, Istanbul, Türkiye
Computer Science, Özyeğin University, Istanbul, Türkiye
Interactive Intelligence Group, Delft University of Technology, The Netherlands
Okan Örsan Özener ORSAN.OZENER@OZYEGIN.EDU.TR
Industrial Engineering, Özyeğin University, Istanbul, Türkiye

Abstract

By examining the patterns of solutions obtained for various instances, one can gain insights into the structure and behavior of combinatorial optimization (CO) problems and develop efficient algorithms for solving them. Machine learning techniques, especially Graph Neural Networks (GNNs), have shown promise in parametrizing and automating this laborious design process. The inductive bias of GNNs allows for learning solutions to mixed-integer programming (MIP) formulations of constrained CO problems with a relational representation of decision variables and constraints. The trained GNNs can be leveraged with primal heuristics to construct high-quality feasible solutions to CO problems quickly. However, current GNN-based end-to-end learning approaches have limitations for scalable training and generalization on larger-scale instances; therefore, they have been mostly evaluated over small-scale instances. Addressing this issue, our study builds on supervised learning of optimal solutions to the downscaled instances of given large-scale CO problems. We introduce several improvements on a recent GNN model for CO to generalize on instances of a larger scale than those used in training. We also propose a two-stage primal heuristic strategy based on uncertainty-quantification to automatically configure how solution search relies on the predicted decision values. Our models can generalize on 16x upscaled instances of commonly benchmarked five CO problems. Unlike the regressive performance of existing GNN-based CO approaches as the scale of problems increases, the CO pipelines using our models offer an incremental performance improvement relative to CPLEX. The proposed uncertainty-based primal heuristics provide 6-75% better optimality gap values and 45-99% better primal gap values for the 16x upscaled instances and brings immense speedup to obtain high-quality solutions. All these gains are achieved through a computationally efficient modeling approach without sacrificing solution quality.

1. Introduction

Combinatorial optimization (CO) problems arise in myriad fields and are generally hard to scale due to their exponential complexity (Korte & Vygen, 2012). Real-world applications of CO are complex systems of many interacting entities with intrinsic relationships; therefore, they require large-scale optimization. The prohibitive combinatorial structure

makes scaling CO algorithms challenging, demanding significant computational power. As a result, there is a continuous need for efficient solution methods to tackle CO problems. On the other hand, various industrial CO problems are routinely solved (Bengio, Lodi, & Prouvost, 2021), such as fleet routing (Choudhury et al., 2021), scheduling for data center management (Mao, Schwarzkopf, Venkatakrishnan, Meng, & Alizadeh, 2019), optimal power flow (Fioretto, Van Hentenryck, Mak, Tran, Baldo, & Lombardi, 2021), and energy market auction (Derinkuyu, Tanrisever, Kurt, & Ceyhan, 2020). These CO routines enable the deduction of the distribution of problem parameters from their varying scenarios. Besides, these scenarios could occur with a constant or similar problem structure. For example, in the optimal power flow problem, energy is transmitted through a grid with differing amounts in time, but the transmission amount follows a specific grid capacity and a time-series pattern. In addition to accumulating problem data, solving CO problems repetitively produces vast empirical data of solution traces by algorithms. All these presented aspects of CO practices create opportunities to solve problems efficiently by exploiting machine learning (ML) to utilize empirical data of historical problem instances. Adopting this idea, artificial intelligence and operations research communities, with a recent surge in interest, develop ML-augmented CO solvers, aiming to construct heuristic solutions end-to-end and enhance existing generalized approaches like Mixed-Integer Programming (MIP) solvers (Wilder et al., 2019; Bengio et al., 2021; Cappart et al., 2021).

Traditionally CO problems are tackled using Branch & Bound (B&B) search, the core algorithm of MIP solvers, by formulating them as *mixed-integer programs*. B&B algorithm solves MIPs by proving optimality; however, it becomes intractable as problems get larger. MIP solvers have been developed by addressing some inherent decision-making procedures of B&B algorithm like *variable selection, node selection*, and *cut selection* and running advanced subroutines like *primal heuristics* and *tightening cut generation*. These various methods have been developed in years of laborious designs and computational studies (Achterberg & Wunderling, 2013; Bestuzheva et al., 2023). Particularly, primal heuristics are especially appealing in scenarios where high-quality solutions are demanded in a short time and periodically, rather than prioritizing the proof of solution optimality. Therefore, neural networks show promise in scaling primal heuristics for CO as they could approximate problem-solution mappings or parameterize algorithms tailored to specific problem data with matrix-based operations on GPUs (Bengio et al., 2021; Chen, Liu, Wang, Lu, & Yin, 2023). Particularly, Graph Neural Networks (GNNs) significantly contribute to MIP, CO, constraint satisfaction, and algorithmic reasoning domains (Cappart et al., 2021). The capabilities of GNNs, such as inductive bias, permutation invariance and equivariance, enable effective pattern recognition from graph-structured information with varying input sizes (Hamilton, Ying, & Leskovec, 2017).

Utilizing ML techniques for CO problems poses several challenges. Firstly, end-to-end learning approaches struggle to generalize on larger-scale problems, possibly due to the increasing complexity of relations among decision variables (Joshi, Cappart, Rousseau, & Laurent, 2021; Liu, Zhang, Tang, & Yao, 2022). Secondly, supervised learning models used in end-to-end approaches require expensive labeled training data, limiting their practicality for NP-hard problems (Joshi et al., 2021). Recent studies (Khalil et al., 2022; Han et al., 2023) indicate that previous work adopting GNNs for branching policy and primal heuristics (Gasse et al., 2019; Ding et al., 2020) experimented with small-scale CO problems that the

state-of-the-art MIP solvers like CPLEX and Gurobi can solve instantly. Additionally, Khalil et al. (2022) remark that their proposed framework MIP-GNN is beneficial for CO problems with past instances and their near-optimal solutions available, which limits its practicality for solving large-scale problems.

Another challenge lies in how the trained supervised models are utilized for downstream optimization tasks. Using the predictions of supervised models often involves ad-hoc processes to obtain feasible solutions for constrained optimization problems and post-hoc strategies for solution search in discrete domains. Common strategies are problem reduction by fixing variables to the predictions, guiding B&B search, and local branching around the predicted solution (Ding et al., 2020; Nair et al., 2020; Khalil et al., 2022; Han et al., 2023; Huang et al., 2022). However, these strategies typically require post-processes and tuning by solving instances of a target problem, leading to development overhead that increases with the number of hyperparameters.

Overall, our study addresses the following limitations and drawbacks of the supervised learning methodology for CO adopted in the literature:

1. A commonly adopted training data generation depends on the collection of high-quality solutions by solving a set of instances of the target problem with a MIP solver. Obtaining such solutions for large instances is time-consuming due to the exponential space complexity of CO problems, causing a bottleneck for the scalability of ML models for solving CO.
2. Most training data generation approaches use multiple solutions per problem instance to set a target space to be learned, which may hinder models from capturing patterns of feasibility and optimality of solutions.
3. Training GNN models requires a massive amount of GPU memory, depending on the scale of training instances. This poses a challenge for the computational feasibility of supervised GNN models for large-scale CO problems.
4. Several studies (Gasse et al., 2019; Ding et al., 2020; Shen et al., 2021; Huang et al., 2022) explore the generalization needed to solve larger instances of CO problems than those used in training, but they lack explicit conclusions. The scalability of end-to-end learning approaches to CO still requires further evidence.
5. The primal heuristics proposed in the current studies rely on class probabilities of predictions or prediction coverage rates to create partial solutions. However, class probabilities might mislead for unusual samples (e.g., larger-scale instances) compared to those in the training dataset, leading to declining solution performance for different-scale instances.

Considering the status quo of the supervised GNN-based solution approaches to CO, we focus on how GNNs can accurately infer solutions to large-scale instances of a given CO problem with tractable training data generation, and their predictions can be efficiently utilized for CO by designing automated primal heuristics. Firstly, we provide an extensive overview of related data-driven optimization approaches, particularly end-to-end learning with GNNs for primal heuristics, and argue their efficacy for scalability in solving CO problems. Accordingly, our study proposes several supplementations for a recent GNN model

for CO, MIP-GNN (Khalil et al., 2022), to allow effectively knowledge transfer for solving larger-scale instances of CO problems. The contributions of the proposed methodology are unfolded below, each addressing the corresponding challenges (listed above) denoted in parentheses.

- We set up the training dataset by collecting optimal solutions of the downscaled instances of a given CO problem in which actual instances are large to hinder obtaining high-quality solutions. This approach allows us to obtain models that capture solution patterns useful for much larger instances, with significantly reduced dataset generation and training costs (Challenges 1, 2, and 3).
- To facilitate knowledge transfer for solving larger-scale instances, we enhance MIP-GNN with various normalization techniques and a different architecture component. Consequently, our models trained on the downscaled instances maintain a steady predictive performance and can scale the proposed primal heuristics for solving larger problems (Challenges 3 and 4).
- We train the models with Evidential Deep Learning (EDL) (Sensoy, Kaplan, & Kandemir, 2018), an uncertainty-quantification method, to calibrate their prediction uncertainty and form our primal heuristics to leverage the predictions considering the associated uncertainty values. In this way, our models are fine-tuned based on decision quality, achieving a superior optimization performance (Challenges 4 and 5).
- We propose *Uncertainty-based Problem Reduction* as a new a primal heuristic algorithm for warm-start solution generation and update the Guided B&B Node Selection algorithm by Khalil et al. (2022) using EDL. These proposed primal heuristics automatically adapt to changing solution patterns due to varying problem sizes, considering the estimated uncertainty values through EDL (Challenges 4 and 5).

We conducted extensive experiments involving five different CO problems commonly benchmarked in the related literature. These experiments benchmark the proposed methodology against the pre-trained MIP-GNN models and CPLEX, an industry-standard solver software that is challenging to outperform with ML approaches (Khalil et al., 2022). We also present an ablation study of the proposed architectural improvements. The models trained on small-scale (1x) instances are evaluated for large-scale (2x, 4x, 8x, and 16x) transfer instances of the same problems. The trained GNNs can accurately generalize on these transfer instances, matching the predictive performance on the test instances (1x). The GNN-augmented CPLEX pipelines, which utilize the proposed primal heuristics (referred to as 'the CO pipelines'), show an increasing trend of performance gain relative to the default CPLEX performance as the scale of problem instances increases. This is in contrast to the deteriorating performance observed in other GNN-based CO approaches (Gasse et al., 2019; Ding et al., 2020; Joshi et al., 2021; Liu et al., 2022). Our experiments aim to efficiently solve the largest scale transfer instances, which are well above the transfer scale attempted in those studies. In comparison to CPLEX over those transfer instances, the CO pipelines achieve approximately 80-99% better primal gap values for three fundamental CO problems and 45% and 60% better primal gap values for the other two more challenging problems presented in the study by Khalil et al. (2022).

As significant evidence of offering a generalized primal heuristics approach, the CO pipelines can boost the efficacy of B&B algorithm for all benchmarked problems—each posing distinct solution processes— and yield high orders of time efficiency to obtain high-quality solutions. Additionally, the transfer performance of the CO pipelines is close to or better than the testing performance of MIP-GNN models. This corroborates the proposed methodology can effectively enable knowledge transfer to solve larger scale instances. Last but not least, our methodology is also cost-efficient while scaling in terms of required computation for training dataset generation and GNN model training, which demand intensive CPU and GPU workload during the development of ML-augmented CO approaches. The source code of our study is publicly available[1].

This article is organized as follows. Section 2 presents the background on MIP and GNN-based methodologies for CO. Section 3 peruses studies on ML techniques for CO and the ML-augmented primal heuristics comprehensively. Section 4 proposes our methodology to solve CO problems at scale with GNN-based primal heuristics. Sections 5 and 6 present the experimental evaluation of our study. Sections 7 and 8 discuss how the proposed methodology can scale and envision further research directions. Lastly, we finalize our paper with a discussion of how it can affect applications in NP-hard domains.

2. Background

We present preliminary information about the methodologies covered in our study and introduce our notation along this section.

2.1 Combinatorial Optimization

CO is a branch of mathematics and computer science that deals with finding the best solution in a finite decision space (Korte & Vygen, 2012). In general, CO problems are formulated as mixed integer linear programming (MILP), which is a mathematical modeling of an optimization problem with integer and continuous decision variables to minimize (or maximize) a linear objective function under a set of linear inequality and/or equality constraints. Formally MILP is the problem form of

$$\arg\min_{\mathbf{x}}\{\mathbf{c}^\top\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbb{R}^n,\ \mathbf{x}_j \in \mathbb{Z}^{|\mathcal{J}|},\ j \in \mathcal{J}\}, \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^n$ is the vector of objective coefficients, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the matrix of constraint coefficients, $\mathbf{b} \in \mathbb{R}^m$ is the vector of constraint bounds, $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ are bounds of decision variables, and $\mathcal{J}$ is the index set of integer decision variables.

2.2 Branch & Bound Search Algorithm

B&B search algorithm is an exact optimization method that is able to quantify the optimality of integral solutions by providing an upper and lower bound on their objective values. During B&B search, a binary tree is constructed by dividing solution space with constraining non-integer solutions. In each B&B node, a linear problem is obtained by relaxing (removing) integrality constraints, then it is solved with linear programming (LP). If

1. https://github.com/furkancanturk/gnn4co

the relaxed solution $\mathbf{x}^{(lp)}$ of a node does not contain fractional values for integer variables, it is a feasible solution to MIP. Otherwise, one of the integer variables taking fractional value $x_j^{(lp)}$ in the relaxed solution is selected to branch the node with two separate bounding constraints $\mathbf{x}_j \leq \lfloor \mathbf{x}_j^{(lp)} \rfloor$ and $\mathbf{x}_j \geq \lceil \mathbf{x}_j^{(lp)} \rceil$ by generating two new nodes. These nodes cover two divergent parts of a constrained decision space that eliminates solutions having fractional values for $\mathbf{x}_j$ from the decision space of the parent node.

B&B algorithm tackles NP-hard problems with two tasks: *Primal task* is finding feasible solutions, and *dual task* is certifying the optimality of the best solution found during the search. *Primal heuristics* aim to achieve feasible solutions quickly to improve *primal bound*. *Dual methods* select variables to branch on so that a tighter *dual bound* (the best LP-relaxed solution value) is computed to prune unexplored B&B nodes with higher (lower) bounds for a given minimization (maximization) problem. In this way, unpromising parts of exponentially growing B&B trees are eliminated, and the search continues on the rest promising unexplored nodes, which are prioritized with respect to a node selection policy. Node selection strategies can be basically categorized in two: (i) selecting the node with the lowest LP-solution value to increase (decrease) the dual bound and (ii) selecting deeper nodes to find integral solutions decreasing (increasing) primal bound for a minimization (maximization) problem (Achterberg, Berthold, Koch, & Wolter, 2008).

2.3 Graph Neural Networks

GNNs are a generic way of modeling knowledge through relations within data (Scarselli, Gori, Tsoi, Hagenbuchner, & Monfardini, 2009; Kipf & Welling, 2017). The fundamental concept underlying GNNs involves the computation of a vectorial representation for each node in a given graph. This is achieved through an iterative process of aggregating embeddings of the neighboring nodes. The aggregation step is parameterized by employing (stochastic) first-order optimization techniques to adapt to the underlying data distribution. The potential benefit of using GNNs for CO is that the learned representations can encode critical graph structures that facilitate the efficient resolution of a CO problem. GNNs are inherently invariant and equivariant, i.e., they can automatically leverage invariances or symmetries that exist in the input instance or data distribution. Due to their local functionality, GNNs naturally exploit sparsity by aggregating neighborhood information, leading to more scalable models on sparse inputs. In our study, we define GNNs in their generalized form, Message Passing Neural Network (MPNN) (Gilmer, Schoenholz, Riley, Vinyals, & Dahl, 2017) in which vectorial messages are exchanged between nodes and node representations are updated using neural networks. MPNN computes a representation vector $\mathbf{h}_v^{(k)}$ for each node at each layer k with the following general form.

$$\mathbf{h}_v^{(k)} = \gamma^{(k)}\Big(\mathbf{h}_v^{(k-1)}, \bigoplus_{u \in N(v)} \big(\phi^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)})\big)\Big) \tag{2}$$

Each layer k has an *update* function $\gamma^{(k)}$, an *aggregation* operator $\bigoplus$, and a *message* function $\phi^{(k)}(\cdot,\cdot)$. Each function at each layer is parameterized with its own nonlinear function, such as a Rectifier Linear Unit (ReLU), a Perceptron, a Multi-layer Perceptron (MLP), and an attention network. $\bigoplus$ can be a permutation-invariant operator such as *sum*, *mean*, *max*, *min*, and *standard deviation*. For a given node v, firstly, message vectors are

computed by ϕ through the adjacent nodes, $u \in N(v)$. Then, these vectors are aggregated as the neighborhood message by $\bigoplus$. After merging the neighborhood message vector with the previous representation vector $h_v^{(k-1)}$ by one of the aggregation operators or column-wise concatenation, $\gamma^{(k)}(\cdot,\cdot)$ computes a new representation for node v. Thus, a k-layer MPNN computes a representation vector for each node which captures information within its k-depth neighborhood in the graph.

2.4 MIP Representation as a Bipartite Graph

Values of decision variables in a constrained optimization problem depend on each other through the constraints and the objective function of the problem. The earlier studies by Gasse et al. (2019) and Ding et al. (2020) model MIPs by representing decision variables and constraints as nodes in an undirected graph. Next, supervised GNNs are used to encode the relationships among the nodes and induce solutions to MIPs. Basically, these models learn how decision variables in the problem take values to form a solution that satisfies a set of constraints and is optimal (or high quality) concerning an objective function. In the following, we describe this bipartite graph representation.

A multiset, a set allowing multiple identical elements, of n nodes (V) for decision variables and a multiset of m nodes (C) for constraints constitute an undirected bipartite graph to represent a MIP formulation, which is depicted in Figure 1. A multiset of edges (E) connects the constraint nodes with corresponding variable nodes. The variable nodes are characterized by $\mathbf{c}$ and their domain types (binary, continuous, and integer). If decision variables are in an integral domain, $\mathbf{l}$ and $\mathbf{u}$ are also used in the feature set of variable nodes. The constraint nodes are characterized by $\mathbf{b}$ and their types ($\leq$ and $=$). Types of decision variables and constraints are one-hot encoded in the nodes. Lastly, the edges are characterized by $a_{i,j} \in \mathbf{A}$. This bipartite graph with the mentioned features fully embodies information in MIP problems, yet more features could be extracted from the root and intermediate nodes of B&B tree as long as B&B search algorithm works (Gasse et al., 2019; Ding et al., 2020; Gupta et al., 2020).

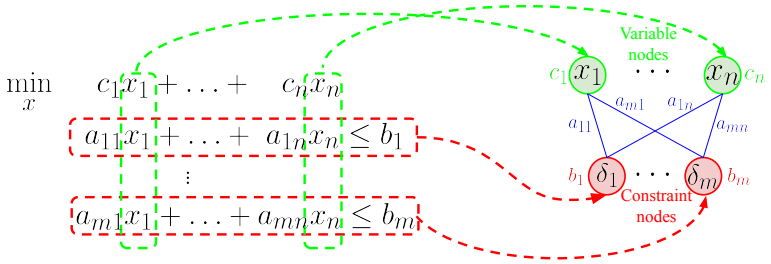


Figure 1: MIP Representation as a Bipartite Graph. Image credit to (Nair et al., 2020).

For each side of the bipartite graph, one graph convolution layer operates to update node representations with passed messages from the other side. We present a definition of graph convolution layers in the form of MPNN for variable and constraint nodes separately in Section 4.1. We recommend readers refer to the study of Chen et al. (2023) on the investigation of the expressive power of GNNs to approximate solutions to given bipartite graph representations of MILPs.

3. Related Work

Existing methods accompanying B&B algorithm in MIP solvers have been advanced to make *node selection* and *variable selection* procedures efficient and incorporate advanced features like *cutting plane generation* and *heuristic method selection* (Zhang et al., 2023). However, developing general MIP solvers and problem-tailored heuristic designs demands considerable time and expert knowledge (Achterberg & Wunderling, 2013; Bestuzheva et al., 2023). Alternatively, ML-augmented MIP solvers leverage data of solutions or trajectories of search or iterative algorithms for a given problem information prior to make one or both of the primal and dual tasks of B&B algorithm more efficient (Cappart et al., 2021; Zhao, Pan, Chen, & Low, 2023).

In the domain of end-to-end learning for CO, two primary ML methodologies emerge: *learning constructive solutions on graphs*, and *ML-augmented primal heuristics* (Cappart et al., 2021; Kotary et al., 2021b). Constructive solution policies for CO problems are obtained through reinforcement learning or tree search which train GNNs to encode graph-structured problems, often unconstrained or loosely constrained like TSP and Minimum Vertex Cover (Dai et al., 2017; Joshi et al., 2021; Peng et al., 2021; Böther et al., 2022). The latter methodology trains GNNs with supervision which predict values of decision variables in near/sub-optimal solutions to constrained CO problems in an end-to-end fashion, but it is challenging that the predicted solution is feasible. Instead, distilling of heuristic information from the trained models to utilize within B&B algorithm can lead to finding high-quality solutions to CO problems faster (Zhang et al., 2023).

The prior works leveraging GNNs within MIP solvers are done by Gasse et al. (2019) for dual branching and Ding et al. (2020) for primal heuristics. Gasse et al. report that the performance improvement of learned variable selection policy in solving CO problems decreases as the size of problems increases. Similarly, the model proposed by Ding et al. can only generalize to solving larger-scale instances of some of benchmarking CO problems. The recent studies by Shen et al. (2021) and Huang et al. (2022) train GNNs on problems with 500-1000 decision variables and evaluate their performance on instances with 1500-3000 decision variables. These two CO pipelines can lead to finding high-quality solutions faster compared to SCIP, an open-source and academic-purpose MIP solver (Bestuzheva et al., 2023), and the framework of Ding et al. (2020). However, the average improvement rates in objective values achieved by the CO pipelines in these three studies are less than 1% compared to SCIP. Another related study proposes a Siamese GNN model to imitate an expert oracle that prioritizes B&B nodes for exploration (Labassi, Chételat, & Lodi, 2022). The oracle accesses the optimal solution obtained before and compares each pair of open B&B nodes accordingly. The study provides a limited assessment of generalization on larger instances, including only 15-50% more graph nodes than those used in training.

Even though GNNs make data-driven CO approaches flexible in solving different-sized problems, they either offer limited improvement or face declining optimization performance on larger-scale instances due to lack of predictive generalization ability and/or unscalable training approaches. Considering the gap between the scales of industrial applications and benchmarking problems in the related works, the scalability of the proposed GNNs for CO remains questionable. Thus, this study is positioned on how to leverage GNNs within primal

heuristics to scale CO solvers. The following subsections compare the related studies with ours and discuss training data generation approaches to ML-augmented optimization.

3.1 ML-Augmented Primal Heuristics

Finding high-quality feasible solutions can be challenging due to complex constraint structures and/or exponentially large decision spaces. Therefore, primal heuristics are more critical for solving hard CO problems than proving the optimality of solutions with dual branching (Khalil, Dilkina, Nemhauser, Ahmed, & Shao, 2017; Berthold, 2018). Primal heuristics construct a solution by virtue of heuristic information about solution pieces or problem structure. In this way, these methods try to reach a good solution by an oriented search toward a particular region of the decision space. The large impact of primal heuristics on B&B algorithm is attested by computational studies on MIP solvers like CPLEX and SCIP (Achterberg & Wunderling, 2013; Berthold, 2018). Berthold reports that the utility of primal heuristics is more prominent on hard problems or ones that cannot be solved in a limited time, according to the experiments with SCIP. However, ascertaining effective ones among a set of alternative heuristics for a given problem requires expert knowledge and rigorous empirical validation. For example, 67 primal heuristic methods have been implemented in SCIP (SCIP Optimization Suite, 2024b). Alternatively, metaheuristics and ML-driven approaches accompany B&B algorithm to run heuristics in more sophisticated and automated ways (Khalil et al., 2017; Karimi-Mamaghan et al., 2022).

This section presents related studies on supervised learning approaches using GNNs to solve CO problems with primal heuristics in a general-purpose (not a problem-specific) way. A standard workflow of these approaches consists of (i) training data generation, (ii) training a GNN model end-to-end with respect to a loss function based on target solutions, (iii) fine-tuning the strategy of using predictions for the downstream optimization task, and (iv) running a MIP solver[2] with proposed primal heuristics utilizing the model predictions. We describe the previous studies considering these four phases.

In the pioneer study by Ding et al. (2020), *stable* decision variables, whose values remain the same in all local solutions around an initial solution, are labeled and included in the training dataset. The most confident predictions are used to generate a partial solution with a predefined size, and a local branching cut is generated to search around this solution up to a predefined Hamming distance. Note that finding an initial feasible solution and ensuring its quality could be challenging for an arbitrary MIP problem. Besides, this approach might prevent learning the global structure of MIP decision space since the stable variables represent information about local structure of solutions. In addition to these challenges, this framework requires several hyperparameters to be fine-tuned for generating the initial solution, local solutions, and partial solution, whereas our study offers much automated design without such features that limit solution quality.

In the study of Nair et al. (2020), Neural Diving generates multiple solution vectors for a given MIP using an energy-based generative GNN. Then, a secondary binary classifier selects a fine-tuned portion of decision variables to be fixed to their predicted values for each sampled prediction. These partial solutions reduce the MIP to different sub-MIPs and

2. ML-augmented MIP approaches are generally solver-independent and have been developed on different MIP solvers like CPLEX, Gurobi, and SCIP as in the related studies.

the best solution obtained solving all sub-MIPs separately is considered the final solution of the pipeline. This approach comes with heavier computation workload of training two models and solving multiple sub-MIPs. Instead, Han et al. (2023) remove the secondary classifier from Neural Diving and propose a local search around a partial solution up to a predefined distance, like the local branching in the study of Ding et al. (2020). The common side of these three studies (Ding et al., 2020; Nair et al., 2020; Han et al., 2023) is that they build on a partial solution generation procedure with a hyperparameter for selecting how many predictions are used. Our study adopts the problem reduction approach as well, but, it leverages all predictions through a new algorithm for warm-starting. This algorithm automatically determines how many predictions with the least uncertainty are used, based on empirical statistics over a validation set used in the training. As a simpler design, Khalil et al. (2022) propose MIP-GNN framework. It learns a *bias vector*, the average vector of the collected solutions with a MIP solver per training instance, binarized with a given bias threshold for classification. The trained model guides B&B search based on the class probabilities (i.e., softmax outputs) for decision values. Our study firstly improves the proposed MIP-GNN architecture to generalize on solving larger-scale instances than those in the training. Besides, it deviates by learning the optimal solutions to small-scale instances and follows an uncertainty-quantification approach in the model training and the B&B node selection strategy. Moreover, our study introduce a new algorithm for generating high-quality warm-start solutions. These advancements result in a scalable and better performing framework with much less cost of training data generation.

The aforementioned three studies rely on collecting high-quality solutions per training instance. These data generation procedures are fine-tuned with several hyperparameters critical for the performance of both subsequent ML and optimization tasks, which is discussed in Section 3.2. Unlike the previous studies, another approach to generate a training dataset is to obtain optimal solutions for small-scale instances of a given CO problem (Shen et al., 2021). In this study, the predicted probabilities by a GNN model are used as scores to guide the variable selection within SCIP until reaching the first feasible solution to a given large-scale problem instance. Upon this study, Huang et al. (2022) propose a bi-level framework which first generates a partial solution (at a predefined size) with the most confident predictions. Then, a secondary GNN model predicts the values of the unfixed decision variables—those not in the partial solution. Interestingly, after obtaining the first feasible solution, both of the proposed methods in these studies do not continue on B&B search neither with the proposed variable selection guide nor with the default configuration of SCIP. In other words, these studies limit the proposed methods as a warm-start solution procedure.

The studies presented so far, except that of Khalil et al. (2022), build on problem reduction or local search approaches, which make them incomplete solution methods as the typical characteristic of primal heuristics. Thus, the proposed methodologies obstruct the global search for better solutions and obtaining the optimal solution. Our study include a global search stage modifying the node selection strategy proposed by Khalil et al., which makes our solution approach a complete optimization method. Another distinct feature of the proposed primal heuristics is the adoption of an uncertainty-quantification approach to utilize the predictions of trained GNNs. In this way, the confidence of the predictor is

quantified regarding its impact on decision quality in the downstream optimization task, instead of using the class probabilities which all presented methods rely on.

Our study comes out to design a scalable and generalized approach to solve CO problems efficiently. On the other hand, only three studies (Ding et al., 2020; Shen et al., 2021; Huang et al., 2022) cover evaluation of the proposed methods over CO problem instances of scales larger than the training instances. Yet, they reach a slightly better performance in solving some of the experimented larger-scale CO problems compared to the default configuration of SCIP. It is worth noting that achieving the performance gain against the state-of-art solvers CPLEX and Gurobi is more challenging than benchmarking with SCIP (Nair et al., 2020; Han et al., 2023). This study employs CPLEX as the baseline solver and achieves substantial performance improvement. We present the further details of the related studies alongside our study as a comparative overview in Table 1.

3.2 Training Data Generation in ML-Augmented Primal Heuristics

Supervised learning-based methodologies for CO aim to map from optimization problems to feasible and high-quality solutions. Ideally, a target object for a MIP instance to be learned by a supervised model is its optimal solution. However, NP-hard problems hinder the traceability of training data generation for supervised models by achieving (near-)optimal solutions (Joshi et al., 2021; Khalil et al., 2022). Hence, the common training data generation procedure is collection of a set of feasible solutions per instance without necessitating an optimal solution to be in the set. However, training data generation procedures prescribed in the studies presented in Section 3.1 lead ML models to different learning tasks than mapping an optimization problem to a single solution vector. This training regime can cause several challenges for both the learning and optimization stages. One challenge related to the nature of optimization problems is that the solution space of an optimization problem can include

- trivial but feasible solutions (e.g., all variables are zeros/ones or randomly generated),
- disparate solutions with close objective values,
- highly similar solutions having a large difference between their objective values,
- symmetrical solutions, and
- alternate optimal solutions.

From the perspective of supervised learning, using multiple solutions per training instance without considering the presence of the listed solution types might prevent ML models from learning valid solution patterns of CO problems. Approximating alternative, symmetrical, or disparate CO solutions together might cause a loss of meaningful solution patterns. Hence it could obstruct GNNs from the induction of feasibility and optimality patterns and such a model would approximate more obscure solutions. Kotary, Fioretto, and Hentenryck (2021a) show that training data generation by eliminating solutions with different structures can improve the prediction performance of learning models and the downstream optimization task. Moreover, Han et al. (2023) show that fixing variables to their predicted values to get a partial solution can form a subproblem that barriers reaching the optimal solution during B&B search.

Table 1: An overview of the related studies on ML-augmented primal heuristics and our study.

Study	**Training Data Generation**	**Problem Representation**	**ML Task**	**Primal Heuristics Strategy**	**Benchmarking**
(Ding et al., 2020)	Local solutions around an initial solution per problem instance	A tripartite graph representation with features extracted from the root node of B&B tree	Classification of values of *stable* binary decision variables	Local search around a partial solution generated from the most confident predictions	Baseline: SCIP Problems: CFL, FCNF, GA, MIS, MK, MSC, TS, and VR
Neural Diving (Nair et al., 2020)	A set of high-quality solutions per problem instance	The bipartite graph with features extracted from the root node of B&B tree (Gasse et al., 2019)	Predicting the probability of being 1 for binary decision variables and classification of decision variables to be fixed	Solving different sub-problems generated from sampled partial solutions	Baselines: Fine-tuned SCIP and Gurobi Problems: CORLAT, MIPLIB, NNV, GP Pack, GP Plan, and EG
(Shen et al., 2021; Huang et al., 2022)	The optimal solutions to the small-scale instances of CO problem.	A graph of decision variables with hand-crafted statistical features	Predicting the probability of being 1 in the optimal solution for binary decision variables	Two-level inference based on the problem reduction and guiding B&B variable selection	Benchmark: SCIP and (Ding et al., 2020) Problems: CA, DS, MIS, and MVC
MIP-GNN (Khalil et al., 2022)	A set of high-quality solutions per problem instance	A bipartite graph with raw features	Classification of *bias* values of binary decision variables	Guiding B&B node selection or variable selection	Baseline: CPLEX Problems: FCMNF and GIS
(Han et al., 2023)	A set of high-quality solutions collected per problem instance	The bipartite graph with features extracted from the root node of B&B tree (Gasse et al., 2019)	Predicting the probability of being 1 for binary decision variables	Local search around a partial solution generated from the most confident predictions	Baselines: SCIP, Gurobi, and Neural Diving Problems: BIP, CA, MIS, and WA
Our study	The optimal solutions to the small-scale instances of CO problem.	A bipartite graph with normalized raw features	Classification of binary decision variable values in the optimal solution	Warm-start solution generation and guiding B&B node selection based on model uncertainty	Baselines: CPLEX and MIP-GNN Problems: CA, FCMNF, GIS, MIS, and MSC

- BIP: Balanced Item Placement
- CA: Combinatorial Auction
- CFL: Capacitated Facility Location
- CORLAT Dataset (Conrad et al., 2007)
- DS: Dominant Set
- EG: Electric Grid Optimization
- FCMNF: Fixed-Charge Multi-Commodity Network Flow
- FCNF: Fixed-Charge Network Flow
- GA: Generalized Assignment
- GIS: Generalized Independent Set
- GP Pack: Google Production Packing
- GP Plan: Google Production Planning
- MK: Multidimensional Knapsack
- MIPLIB: Mixed-Integer Programming Library (Gleixner et al., 2021)
- MIS: Maximal Independent Set
- MSC: Minimum Set Cover
- MVC: Minimum Vertex Cover
- NNV: Neural Network Verification
- TS: Traveling Salesman
- VR: Vehicle Routing
- WA: Workload Appointment

Secondly, setting up a target solution space from a pool of solutions per problem instance is subject to several hyperparameters, which require a fine-tuning procedure analyzing their effect on both ML and optimization tasks. Considering MIP-GNN framework by Khalil et al. (2022), one needs to determine the size of the solution pool (the collection of feasible solutions per instance), a time limit to run a solver to collect solutions, the search strategy of the MIP solver (i.e., managing trade-off of finding feasible solutions or proving the optimality gap), an optimality gap threshold to include solutions in the pool, and a bound on the relative gap between the objective values of the worst and the best solution in the pool. Similarly, the approach proposed by Ding et al. (2020) requires several design aspects to be considered. The proposed training dataset generation is based on a local search around an initial feasible solution, which requires to be fine-tuned by finding the best scope size and stopping criteria for the local search. Additionally, the size of the partial solution constructed from the most confident predictions and the scope size of the local branching are also fine-tuned for each problem domain.

We further propound the training data generation based on the optimal solutions for the end-to-end learning for CO, considering the outlook on the adopted approaches in this section and aiming to come up with a simpler yet effective design.

4. Proposed Methodology

In this study, we propose a supervised GNN enabling knowledge transfer to solve CO problems at scale after training over downscaled instances of a problem with their optimal solutions. We first provide a formal definition of our approach as the following.

Let $f_\theta(M) \rightarrow \mathbf{x}^*$ be a function parameterized with θ that maps a CO problem $M = (\mathbf{A}, \mathbf{b}, \mathbf{c}) \sim p(M)$ with $n \sim p(n)$ decision variables to its optimal solution x^*. To optimize f_θ, obtaining a training dataset $D = \{(M_i, \mathbf{x}_i^*)\}_{i=1}^{I}$, where i is problem instance index, is practically infeasible for large instances of M since finding $\mathbf{x}^*$ is NP-hard and the space complexity of M is $\mathcal{O}(2^n)$. Although CO problems have exponential space complexity, small-scale instances can be solved exactly with a MIP solver in a practically reasonable time. Assume $M' \sim p(M)$ is a downscaled instance of M with $n' \ll n$ decision variables. The optimal solution to M', $\mathbf{x}'^*$ can be found in a practical time as long as n' is sufficiently small. Therefore, instead of D, we generate a small-scale problem instances M_i' with n' variables by sampling its parameters $(\mathbf{A}', \mathbf{b}', \mathbf{c}')$ from $p(M)$ and obtain the optimal solution for each instance $(\mathbf{x}_i'^*)$ with a MIP solver. Then, a GNN-based model trained on the dataset $D' = \{(M_i', \mathbf{x}_i'^*)\}_{i=1}^{I'}$ is used to solve the actual-scale instances of M.

The proposed training scheme necessitates that the distribution of the target problem parameters $p(M)$ is already known. This condition is practically satisfiable since $p(M)$ is revealed as long as instances of the target CO problem M occur repetitively in real-world applications.

We employ a recent GNN model for CO, MIP-GNN (Khalil et al., 2022), and propose several supplementations and changes for it, which are introduced in sections 4.1.1, 4.1.2, and 4.1.3. Lastly, Section 4.2 proposes two primal heuristics utilizing the predictions and uncertainty estimations by the models trained on D' to solve D.

4.1 GNN Architecture for Combinatorial Optimization

MIP-GNN architecture depicted in Figure 2 constitutes a graph embedding block of multiple MPNN layers and a classifier neural network that receives variable node representations from each MPNN layer. It differs from the commonly adopted model of Gasse et al. (2019) by using different graph convolutions and an additional layer named *Error Layer*, which propagates error messages between consecutive variable node embedding layers. In this section, we describe the MIP-GNN that uses Edge Convolution (EC)[3] (Simonovsky & Komodakis, 2017) as the MPNN layers and adopt it in our study with some changes introduced in the following subsection. Note that the activation functions of MLP networks mentioned throughout the study are ReLU.

MIP-GNN uses $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ as the features for edges, constraint nodes, and variable nodes, as described in Section 2.4. It also uses node degrees, variable types, and constraint types as additional features. We define feature vectors for variable and constraint nodes as follows.

Let $\mathbf{t}^{(v)} \in \{0,1\}^{n\times 3}$ be a matrix in which each row $\mathbf{t}_j^{(v)}$ is the one-hot encoding for the type of variable j (binary, continuous, or integer). Let $\mathbf{t}^{(c)} \in \{0,1\}^m$ be a vector for constraint types if ones for $\leq$ expressions and zeros for equalities[4]. Let $\mathbf{d}_i^{(v)} \in \mathbb{Z}^n$, and $\mathbf{d}_i^{(c)} \in \mathbb{Z}^m$ be the vectors of constraint and variable node degrees, respectively. Therefore, the feature vector of variable node j is represented by a column-wise concatenation $\mathbf{v}_j^{(\cdot)} = [\mathbf{c}_j, \mathbf{t}_j^{(v)}, \mathbf{d}_j^{(v)}]$ and the feature vector of constraint node i is represented by a column-wise concatenation $\mathbf{c}_i^{(\cdot)} = [\mathbf{b}_i, \mathbf{t}_i^{(c)}, \mathbf{d}_i^{(c)}]$.

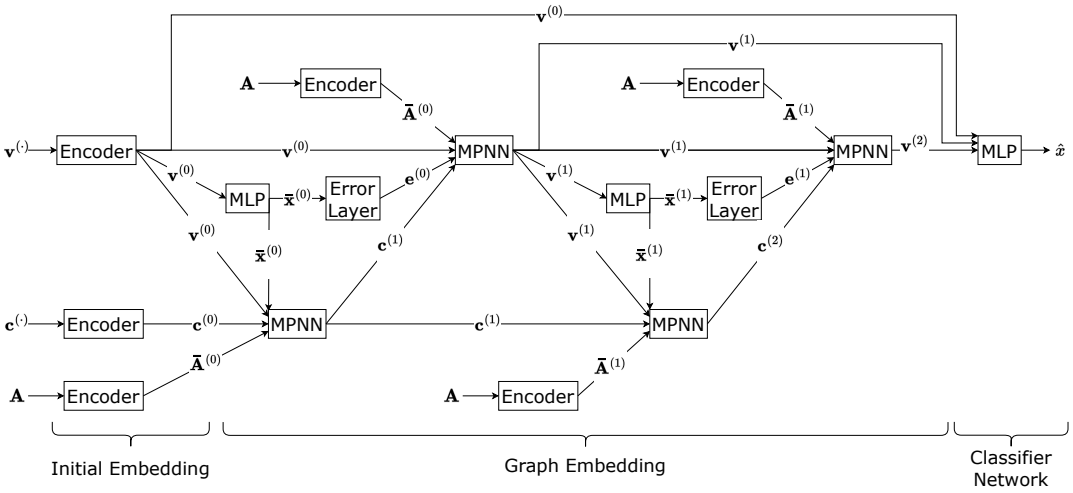


Figure 2: MIP-GNN Architecture with Two-Layer Graph Embedding

3. The definition of MIP-GNN in the study of Khalil et al. (2022) differs from its official implementation at `https://github.com/lyeskhalil/mipGNN`. In our study, we provide a definition for the MIP-GNN with EC considering the implementation.
4. Note that $<$, $>$, and $\geq$ expressions can be converted to $\leq$ expressions. For example, a $>$ expression is converted to a $\leq$ expression by adding ϵ quantity to the right-hand side and then multiplying that expression by -1. Therefore, $\mathbf{t}^{(c)}$ is a sufficient encoding to represent the type of any equality or inequality expression.

MIP-GNN builds on three blocks: (i) an initial layer to embed the raw feature vectors of the nodes, (ii) multiple consecutive message-passing layers to learn hidden representations of nodes, and (iii) a task layer to classify the binary decision values for variable nodes. The initial embedding layers for variable and constraint nodes, each a two-layer MLP network, map the dimension of node features to the hidden layer dimension (d) of the following MPNN layers.

Variable-to-constraint pass Let $f_e^{(k)}$ be an edge encoder at layer k that is parameterized by two-layer MLP with Batch Normalization. It receives constraint coefficients $\mathbf{A}$ and computes $\bar{\mathbf{A}}^{(k)} = f_e^{(k)}(\mathbf{A}) \in \mathbb{R}^{m \times d}$. Let $f_s^{(k)}$ a two-layer MLP followed by a sigmoid function receiving embedding of variable nodes $\mathbf{v}^{(k-1)}$ and predict a soft decision value vector $\hat{\mathbf{x}}^{(k)} \in [0,1]^n$ for variable nodes, $\bar{\mathbf{x}}^{(k)} = f_s^{(k)}(\mathbf{v}^{(k-1)})$. Accordingly, an MPNN with EC at layer k computes constraint node representations as

$$\mathbf{c}_i^{(k)} = \mathrm{ReLU}\Big(\bigoplus_{j \in N(i)} \phi_c^{(k)}\big([\mathbf{c}_i^{(k-1)}, \mathbf{v}_j^{(k-1)}, \bar{\mathbf{A}}_{i,j}^{(k-1)}, \hat{\mathbf{x}}_j^{(k-1)}]\big)\Big),\ \forall i \in C, \tag{3}$$

where $\bigoplus$ is `mean` aggregator and $\phi_c^{(k)}$ is a two-layer MLP receiving the column-wise concatenation of $\mathbf{c}_i^{(k-1)}, \mathbf{v}_j^{(k-1)}, \bar{\mathbf{A}}_{i,j}^{(k-1)}$, and $\hat{\mathbf{x}}_i$.

Constraint-to-variable pass Analogously, variable node representations are computed as

$$\mathbf{v}_j^{(k)} = \mathrm{ReLU}\Big(\bigoplus_{i \in N(j)} \phi_v^{(k)}\big([\mathbf{v}_j^{(k-1)}, \mathbf{c}_i^{(k)}, \bar{\mathbf{A}}_{i,j}^{(k-1)}, \mathbf{e}_i^{(k-1)}]\big)\Big),\ \forall j \in V, \tag{4}$$

where $\mathbf{e}_i^{(k-1)}$ is the error signal computed by *Error Layer* defined in Equation 5.0 and 5.1. It receives a soft solution vector $\bar{\mathbf{x}}^{(k)}$ predicted by $f_s^{(k)}$ and computes a residual vector $\Delta \in \mathbb{R}^m$ (Equation 5.0). Then, a two-layer MLP followed by a softmax function $f_e^{(k)}$ computes an error signal vector $\mathbf{e}^{(k)} \in \mathbb{R}^m$ (Equation 5.1). The softmax function normalizes the output of MLP row-wise, so how much each constraint contributes to the total error signal is computed with Error Layer.

$$\Delta = \mathbf{A}\bar{\mathbf{x}}^{(k)\top} - \mathbf{b} \tag{5.0}$$

$$\mathbf{e}^{(k)} = f_e^{(k)}(\Delta) \tag{5.1}$$

Lastly, a four-layer MLP classifier predicts solution values $\hat{\mathbf{x}}$ for the nodes of decision variables. It receives a column-wise concatenation of hidden representations from each MPNN layer and outputs softmax values for 0 and 1 values for each decision variable node. For the given node representation of a decision variable in a CO problem instance, the class having the largest softmax value is attributed to the predicted value of the decision variable.

4.1.1 Feature Scaling and Normalization Layers

AbcNorm We propose a set of operations called `AbcNorm` to scale $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ coefficients to be in $[-1, 1]$ by considering that $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ might be scaled with respect to the number of decision variables in a CO domain. Therefore, the scale of features of the bipartite graph is made invariant to the problem size. For a given feature tuple $(\mathbf{A}, \mathbf{b}, \mathbf{c})$, Equation 6.0 scales

the coefficient of decision variable j in constraint i ($\mathbf{a}_{i,j}$) by the maximum of the absolute values of coefficients in constraint i and constraint bound b_i. Equation 6.1 scales constraint bound $\mathbf{b_i}$ likewise. Equation 6.2 scales the objective coefficients by the maximum of their absolute values.

$$\mathbf{A}_{i,j} \leftarrow \frac{\mathbf{A}_{i,j}}{\max_{i \in C, j \in V} \left\{ |\mathbf{A}_{i,j}|, |\mathbf{b}_i| \right\}} \; \forall i \in C, \; \forall j \in V \tag{6.0}$$

$$\mathbf{b}_i \leftarrow \frac{\mathbf{b}_i}{\max_{i \in C, j \in V} \{ |\mathbf{A}_{i,j}|, |\mathbf{b}_i| \}} \; \forall i \in C \tag{6.1}$$

$$\mathbf{c}_j \leftarrow \frac{\mathbf{c}_j}{\max_{j \in V} \{ |\mathbf{c}_j| \}} \; \forall j \in V \tag{6.2}$$

Similarly, the variable node and the constraint node degree values are scaled by the maximum degree of the same node types as follows.

$$\mathbf{d}_j^{(v)} \leftarrow \frac{\mathbf{d}_j^{(v)}}{\max_{j \in V} \{ \mathbf{d}_j^{(v)} \}} \; \forall j \in V \tag{7.0}$$

$$\mathbf{d}_i^{(c)} \leftarrow \frac{\mathbf{d}_i^{(c)}}{\max_{i \in C} (\{ \mathbf{d}_i^{(c)} \}} \; \forall i \in C \tag{7.1}$$

PreNorm (Gasse et al., 2019) It is used at the initial embedding layers for variable nodes, constraint nodes, and edges to stabilize model training. It standardizes feature vectors of nodes and edges in a graph by computing the empirical mean and standard deviation of each feature.

GraphNorm (Cai, Luo, Xu, He, Liu, & Wang, 2021) To enhance the stability of message flow through the graph embedding block and the classifier network when inferring solutions for larger-scale instances of a CO problem, we use GraphNorm at the end of each MPNN layer before ReLU function and after each hidden layer of the classifier network. It normalizes hidden representations across nodes within a graph rather than across batches. For a given node embedding $\mathbf{h}_i$, it is defined as

$$\text{GraphNorm}(h_{i,d}) = \gamma_d \cdot \frac{\mathbf{h}_{i,d} - \alpha_d \cdot \mu_d}{\hat{\sigma}_d} + \beta_d, \tag{8}$$

where μ_d and $\hat{\sigma}_d$ are respectively the mean and the standard deviation of feature d in the graph nodes, γ_d is the scaling and β_d is the shifting parameter. Lastly, α_d is the learnable parameter for each feature d to control keeping the mean information of features.

4.1.2 Violation Layer

Considering the inference of solutions for larger-scale instances, we introduce this new layer on behalf of Error Layer. Similarly, *Violation Layer* receives a soft solution vector $\bar{\mathbf{x}}$ predicted by f_s. Then, it encodes an error signal vector $\mathbf{e}$, which is formulated in Equation 9, where $\mathbf{q}^{(c)} \in \mathbb{R}^m$ is the vector of reciprocal values of constraint node degrees $\mathbf{d}^{(c)}$. In contrast to Error Layer, Violation Layer normalizes Δ with $\mathbf{q}^{(c)}$ to make the magnitude of values in Δ invariant to the graph size. Also, it includes the constraint types for the

computation of the error signal so that all information related to constraints in a given problem is embedded in this layer. Lastly, it passes the encoding by the MLP through GraphNorm and tanh function so that the error signal by each constraint is normalized relative to other constraints and ranged between $[-1, 1]$ to avoid extreme signal values that are far away from the learned signal distribution. Note that Violation Layer does not use actual violation values by a solution vector for inequality constraints[5].

$$\Delta = \mathbf{A}\bar{\mathbf{x}}^{\top} - \mathbf{b} \tag{9.0}$$

$$\bar{\Delta} = \mathbf{q}^{(c)}\Delta \tag{9.1}$$

$$\mathbf{e} = \tanh\Big(\text{GraphNorm}\big(\text{MLP}([\bar{\Delta},\ \mathbf{t}^{(c)}])\big)\Big) \tag{9.2}$$

4.1.3 Combined Aggregation

The expressive power of GNN models heavily depends on the aggregator operator(s) used in them. `sum`, `mean`, `std`, `max`, and `min` aggregation operators can conduce to learning distinctive features of graphs. Alternatively, using multiple operators could improve the expressive power of GNN models (Corso, Cavalleri, Beaini, Liò, & Velickovic, 2020). Therefore, we opt for *combined aggregation* (`comb`) defined in Equation 10 according to the study of Corso et al. (2020), instead of `mean` aggregation used in the original form of MIP-GNN architecture. The messages with the dimension of $1 \times h$ by `mean`, `std`, `max`, and `min` aggregations are concatenated column-wise. Then, the concatenated message is linearly projected into a combined message $\mathbf{m}_{\texttt{comb}}$ by a learnable parameter matrix $\mathbf{W}_{4h\times h}$. We exclude `sum` operator in our study since it is sensitive to neighborhood size – it can cause unstable hidden states for the bipartite graph nodes of CO problem instances with different sizes (Joshi et al., 2021).

$$\mathbf{m}_{\texttt{comb}} = \big[\mathbf{m}_{\texttt{mean}}, \mathbf{m}_{\texttt{std}}, \mathbf{m}_{\texttt{max}}, \mathbf{m}_{\texttt{min}}\big]_{1\times 4h} \otimes \mathbf{W}_{4h\times h} \tag{10}$$

4.2 Uncertainty Quantification-based Primal Heuristics

As neural networks are approximation models, it is challenging that their outputs are guaranteed to satisfy a system of constraints (Donti et al., 2021; Fioretto et al., 2021; Zhao et al., 2023). Although the outputs as a whole are not guaranteed to be a feasible solution, fragments of them could be leveraged by heuristics to accelerate CO. Distinguishing which predictions are more reliable for constructing partial solutions typically depends on thresholding class probabilities or setting a prediction coverage rate to use the most confident predictions (recall Section 3.1). Class probabilities of classifier neural networks are traditionally derived by the softmax function, but, can be unreliably quantified for unusual samples (i.e., overconfident predictions even for samples not similar to those in the training) (Sensoy et al., 2018). Insted, we propose an uncertainty-quantification approach to harness the predictions of trained GNN models as heuristic information to get partial solutions and to guide B&B search.

Uncertainty quantification approaches such as Bayesian Neural Networks (Neal, 2012), Monte-Carlo Dropout (Gal & Ghahramani, 2016), and Evidential Deep Learning (EDL)

5. Early experiments of this study included $\text{ReLU}(\Delta)$ as the actual violation for inequality constraints ($\leq$), resulting decrease in the accuracy of models for larger-scale instances of some problems.

(Sensoy et al., 2018) aim to reduce model uncertainty and calibrate prediction confidence. If an ML model can successfully estimate the uncertainty of its predictions, those with low uncertainty are much more likely to be accurate than highly uncertain predictions. In this study, we employ EDL for training GNN models since EDL is a closed-form solution to estimate model uncertainty accurately and thus does not require a post-training process for the proposed primal heuristics. We describe EDL Loss in the following.

A classification problem is characterized by estimating the categorical distributions of the classes. Therefore, a Dirichlet distribution is predicted with EDL Loss for a K-class classification problem, which is parametrized by a K-dimensional vector $\alpha \in \mathbb{R}^+$. For a given MIP instance with n decision variables to be predicted, an evidence vector $\mathbf{e} \in \mathbb{R}^K$ is computed by using softplus function at the output layer of MIP-GNN and the Dirichlet parameters are predicted with $\hat{\alpha}_j = \mathbf{e}_j + 1$ to satisfy $\alpha \in \mathbb{R}^+$. Then, Kullback-Leibler Divergence Score (KL), which quantifies how much one distribution differs from another, is used as the regularizer term in EDL Loss so that total evidence is minimized for misclassified samples. Accordingly, EDL Loss is computed as

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \mathbf{y}_{jk} \big(\psi(S_j) - \psi(\hat{\alpha}_{jk}) \big) + \lambda KL(\hat{\mathbf{y}}_j, \hat{\alpha}_j), \tag{11}$$

where $\mathbf{y}_j$ is the one-hot encoded target value of decision variable j, ψ is digamma function, λ is the regularization coefficient, and $S_j = \sum_{k=1}^{K} \hat{\alpha}_{jk}$ (Sensoy et al., 2018). Each prediction of an EDL model is associated with a total uncertainty value which is computed as $u_j = 2/S_j$ when $K = 2$. Hence, we use $u_j \in (0, 1]$ as the uncertainty value associated with the prediction $\hat{\mathbf{x}}_\mathbf{j}$ for decision variable j in a given MIP. Similarly, we compute $\mathbf{u}_{j,k} = 1/\alpha_{j,k}$ as the model uncertainty for a given value k of decision variable j.

We use the predictions of our GNN model trained with EDL Loss and the associated uncertainty values $\hat{\alpha}$ together to get up partial solutions and to guide B&B search. Accordingly, our primal heuristic strategy consists of two stages: (i) *Warm-start solution generation by problem reduction* and (ii) *Uncertainty-guided B&B search*. The best solution obtained in the first stage is used as a warm-start solution in the second stage. Since we regularize our Evidential GNN model with λ to calibrate its prediction confidence, we could incorporate information from the downstream optimization task into the learning phase of the proposed methodology by using the model uncertainty in primal heuristic methods.

4.2.1 Warm-start Solution Generation by Problem Reduction

One can reduce the problem by fixing some decision variables to their predicted values by an end-to-end learning model. The best solution in the reduced decision space can be found in a relatively much shorter time, with a compromise of optimality for the original problem, but ensuring the feasibility of the reduction could be challenging. Adopting this heuristic approach, we propose a model uncertainty-based algorithm to select which decision variables are fixed to their predicted values. As predictions with lower uncertainty are more likely to be accurate, fixing those variables is likely to be a feasible problem reduction. Differently from the similar approaches presented in Section 3.1, our approach still leverages all predictions in the problem reduction, rather than using a static portion of the predictions.

Let $\mathbf{u}$ be the vector of uncertainty values associated with the predicted values $\hat{\mathbf{x}}$ for the decision variables in a MIP instance M. The computation of $\mathbf{u}$ for EDL was described above. For the models trained with BCE Loss, we set $\mathbf{u}_j = 1 - \max\{p(\mathbf{x}_{j,0}), p(\mathbf{x}_{j,1})\}$, where $p(\mathbf{x}_{j,k})$ is corresponding softmax value for class k, as used in the study of Khalil et al. (2022). Accordingly, a set of cuts Ω is generated (Equation 12) by fixing the decision variables to the rounded prediction values $\text{round}(\hat{\mathbf{x}}) \in \{0,1\}^n$ in which associated uncertainty values are less than a threshold value $\bar{u}$.

$$\Omega = \{x_j := \text{round}(\hat{\mathbf{x}}_j) \mid \mathbf{u}_j \leq \bar{u},\ j \in \mathcal{V}\} \tag{12}$$

Adding the confident cuts Ω to M results in a subproblem M_{sub} covering a decision space of the unfixed variables. A MIP solver runs to solve M_{sub} under a limited time if M_{sub} is a feasible problem. The incumbent solution obtained by this run serves as a warm-start solution for a subsequent global search for M.

A design question is how to set $\bar{u}$. If the predictions for larger-scale instances are as accurate as the predictions for unseen instances at the problem scale of those in training instances (in short, for the testing instances), one can determine what portion of predictions are totally/almost accurate based on a validation set. We empirically validate this claim by presenting the predictive performance results in Section 6.1. Note that the $\bar{u}$ threshold filters the most confident predictions to get a partial solution. However, if a model accurately predicts most of the values of decision variables in the optimal solution, it is better to cover the predictions as much as possible, rather than only the most confident predictions. Therefore, we opt for utilizing all predictions to generate a warm-start solution in an iterative manner. Accordingly, we propose *Uncertainty-based Problem Reduction* (UPR) in Algorithm 1. It removes a portion of uncertain predictions from the predicted solution iteratively and obtains a smaller partial solution at each iteration which covers a subset of the previous (partial) solution. In this way, this algorithm does not require finding the best coverage rate for each problem domain if $\bar{u}$ can be set to cover only accurate predictions, as distinct from the similar ML-augmented approaches presented in Section 3.1.

In Algorithm 1, Line 1 generates all cuts $\Omega^{(0)}$ by assigning the values of all variables to the corresponding rounded values of $\hat{\mathbf{x}}$. Line 2 modifies M by adding the cuts Ω and results in a (totally) reduced problem $M_{sub}^{(0)}$. Line 3 calculates a minimum percentile value to keep p_{min}% of the most confident cuts in Ω. Line 4 calculates dp, which quantifies what percentile of the most uncertain cuts will be removed in each iteration of the subsequent loop. A loop begins where top p% of the most uncertain cuts in Ω are removed from $M_{sub}^{(i-1)}$, and a MIP solver runs to solve $M_{sub}^{(i)}$ for the given time t with the incumbent solution from the previous iteration as a warm-start. Note that no cuts are removed, and no warm-start solution is available at the beginning of the first iteration. Unless M_{sub} is infeasible or a feasible solution cannot be found within the iteration time limit τ_{i-1}, the best solution found at each iteration is given to the solver as a warm-start solution. The best solution obtained eventually $\mathbf{x}^{(N)}$ will be used as a warm-start solution later for a global search with the solver.

To enhance the completeness of the proposed warm-start solution generation, we also consider the case that the last sub-MIP in Algorithm 1 ($M_{sub}^{(N)}$) is infeasible and introduce a repair procedure at Appendix A. Note that if $M_{sub}^{(i+1)}$ is infeasible due to $\Omega^{(i+1)}$, then $M_{sub}^{(i)}$

Algorithm 1: Uncertainty-based Problem Reduction

Input : MIP instance M, set of confident cuts Ω, number of iterations N, vector of iteration time limits τ

Output: A solution $\mathbf{x}$

$\Omega^{(0)} = \{x_j := \text{round}(\hat{\mathbf{x}}_\mathbf{j}),\ \mathbf{j} \in \mathcal{V}\}$

$M_{sub}^{(0)} \leftarrow \text{AddCuts}(M, \Omega^{(0)})$

$p_{min} = 100 * \frac{||\Omega||}{N}$

$\text{d}p = \frac{1-p_{min}}{N}$

$\mathbf{x}^{(0)} \leftarrow \emptyset$ // Initialize a null solution

for $i \in \{1, ..., N\}$ **do**

  $p = 1 - (i-1) * \text{d}p$

  $\Omega^{(i)} = \text{GetUncertainCuts}(\Omega, p)$

  $M_{sub}^{(i)} \leftarrow \text{RemoveCuts}(M_{sub}^{(i-1)}, \Omega^{(i)})$

  $\mathbf{x}^{(i)} \leftarrow \text{Solve}(M_{sub}^{(i)}, \mathbf{x}^{(i-1)}, \tau_{i-1})$

end

return $\mathbf{x}^{(N)}$

was also infeasible since $\Omega^{(i)} \subset \Omega^{(i+1)}$. Therefore, a feasible solution cannot be obtained at any iteration of Algorithm 1 if $M_{sub}^{(N)}$ is infeasible.

4.2.2 Uncertainty-Guided B&B Search

We adopt *Guided node selection* technique proposed by Khalil et al. (2022) where B&B nodes are prioritized based on the predictions of the trained MIP-GNN model. Khalil et al. define a confidence score for B&B nodes as the sum of corresponding softmax values of predicted classes (zero or one) for fixed variables in a B&B node. In this way, B&B nodes aligned with model predictions are prioritized to explore earlier, and the search is oriented around the predicted solution by the model.

Instead of using the class probabilities by models trained with BCE Loss, we propose leveraging the evidence values computed by EDL models for decision variable values (0 and 1) using $\mathbf{u}_{j,k} = 1/\alpha_{j,k}$ defined previously. If the model computes high evidence for the decision variable value k, then $\mathbf{u}_{j,k} \to 0$. Otherwise, $\mathbf{u}_{j,k} \to 1$. Accordingly, for a given B&B node b, a confidence score s_b is computed by Equation 13, where $\mathcal{J}_b$ is the indices of fixed variables in b and $x_i^{(b)}$ is the value to which decision variable j is fixed in node b.

$$s_b = \sum_{j \in \mathcal{J}_b,\ k = x_j^{(b)}} (1 - \mathbf{u}_{j,k}) \tag{13}$$

Since s_b increases as much asthe depth of nodes increases, B&B search starts diving toward the predicted solution vector. After branching the nodes aligning with the confident predictions, class uncertainty values are more likely to be equal (i.e., $\mathbf{u}_{j,0} \approx \mathbf{u}_{j,1}$) for the decision variables with higher prediction uncertainty. Henceforth, partial solutions (i.e., intermediate B&B nodes) that include the confident predictions are retained until deeper

B&B nodes including uncertain predictions are explored. This exploration continues by flipping the values of decision variables with high uncertainty at first, posing oscillations around the decision space where the model predictions are precise. In short, the proposed B&B search manages the trade-off between exploitation of the predicted solution and exploration for better ones by following a policy configured with the model uncertainty.

5. Experimental Setting

We have conducted extensive experiments for the evaluation of our methodology over five CO benchmarking datasets presented in Section 5.1. The implementation of our study is detailed in Section 5.2. We establish a CO pipeline that integrates a trained GNN into CPLEX and uses its predictions and associated uncertainty values within the proposed primal heuristics to solve CO problems. Our experimentation covers mainly 12 different CO pipeline configurations per CO problem, thus, 60 pipelines in total. Besides, 12 additional pipelines are set for benchmarking with the pre-trained MIP-GNN models using EC released by Khalil et al. (2022) (see Section 5.2). All these pipelines are configured with the following three settings.

- **Model Architecture**: Models with Violation Layer and Error Layer are denoted by EC+V and EC+E, respectively. The plain models are denoted by EC.
- **Loss Function**: Models trained with BCE and EDL loss functions are denoted by BCE and EDL, respectively.
- **Primal Heuristics Strategy**: CO pipelines that apply UPR algorithm for warm-starting before the global search with Guided B&B Node Selection (NS) strategy and those that do not apply UPR are denoted by UPR+NS and NS, respectively.

The CO pipelines using the trained models with the first two settings are denoted by the concatenated abbreviations: EC+BCE, EC+EDL, EC+V+BCE, EC+V+EDL, EC+E+BCE, and EC+E+EDL. EC+V+EDL is the model including the all proposed improvements. Each CO pipeline is dedicated to solving the instances of the corresponding CO problem. These CO pipelines are benchmarked with the default configuration of CPLEX and those deploying the pre-trained MIP-GNN models[6] under a 30-minute time limit per instance, as in the study of Khalil et al. (2022). The pre-trained MIP-GNN models with and without Error Layer are similarly denoted by MIPGNN and MIPGNN+E, respectively. The time limit includes the elapsed time in data preprocessing, prediction, and optimization. Hence, the remaining time limit for the CO pipelines to solve each instance is less than 30 minutes (see Table 6 in Appendix B).

5.1 Benchmarking Datasets

We evaluate our methodology on five CO problem datasets. Three of these includes fundamental CO problems which are *Minimum Set Cover* (MSC), *Combinatorial Auction* (CA), and *Maximal Independent Set* (MIS) (Gasse et al., 2019). The other two datasets are from the study of Khalil et al. (2022) and include CO problems harder than the previous three,

6. The models are available at `https://github.com/lyeskhalil/mipGNN`.

which are *Fixed-Charge Multi-Commodity Network Flow* (FCMNF) (Hewitt et al., 2010), and *Generalized Independent Set* (GIS) (Colombi et al., 2017).

We generated 1000 training, 200 validation, and 50 testing instances for each problem. The problem scale of the instances is the same in these three datasets. To investigate the scalability of our methodology, we generated four transfer datasets for MSC, CA, and MIS problems which contain 2, 4, 8, and 16 times the number of variables in the training instances. Table 5 in Appendix B includes the number of decision variables and constraints in the training dataset of each problem. The problem scales of the testing and four transfer datasets are denoted by 1x, 2x, 4x, 8x, and 16x, respectively[7]. We generated the training, validation, and test datasets (1x) of FCMNF and GIS problems by respectively 16x and 4x downscaling the size of the actual datasets used by Khalil et al.. We call these actual datasets as transfer datasets throughout our evaluation as well. Each transfer dataset of five problems comprises 50 instances. Our models are trained on the downscaled instances of FCMNF and GIS problems and benchmarked with the MIP-GNN models over FCMNF-16x and GIS-4x datasets. Note that the MIP-GNN models were trained on FCMNF-16x and GIS-4x datasets; therefore, this benchmarking measures their testing performance, whereas our models' transfer performance.

Alongside the 30-minute benchmarking runs, we also ran the default configuration of CPLEX (referred as only 'CPLEX' from here on) longer for the transfer datasets so that we could figure out how long we had to run it to achieve the solution quality of the CO pipelines. For these long CPLEX runs, we set the time limit per instance to 1, 2, 4, and 8 hours for 2x, 4x, 8x, and 16x transfer datasets, respectively, of MSC, CA, and MIS problems. Similarly, the time limit of the long CPLEX runs per instance is 8 hours for FCMNF-16x and GIS-4x datasets as these are the largest-scale transfer datasets of FCMNF and GIS problems. Further details of the training data generation are provided in Appendix B.

5.2 Implementation Details

For our models, we use an 8-layered graph embedding block and a 4-layered MLP as the classifier network. We set the hidden dimension to 32 for all layers and the dropout probability to 0.1 for the classifier network. We train additional models that have the same complexity (i.e., 4-layered GNN and `mean` aggregation) with the pre-trained MIP-GNN for fair benchmarking.

All models are trained using ADAM optimizer (Kingma & Ba, 2015) with eight batches in 24 epochs. The learning rate is initialized at 0.0001 and managed using One-cycle learning rate scheduler (Smith & Topin, 2019). 25% of the model update steps are spent to increase the learning rate to 0.001. Other hyperparameters of One-cycle scheduler are set to the default values in its PyTorch implementation[8].

Evidential Deep Learning Regularization We conducted a grid search of the regularizer parameter λ of EDL Loss by observing its effect on solving the largest-scale dataset

7. Size of MIPs can also be quantified w.r.t. $n \times m$ or number of nonzeros in the matrix of constraint coefficients matrix $\mathbf{A}$ (Gleixner et al., 2021), which would indicate a much greater quantification for the scales of transfer datasets.
8. `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html`

of each problem. Accordingly, we set λ to 1, 2, 6, 6, and 12 for the EDL models trained on MSC-1x, CA-1x, MIS-1x, FCMNF-1x, and GIS-1x datasets, respectively.

Uncertainty-based Problem Reduction It is expected that a warm-starting procedure is completed within a short time; therefore, we limited UPR Algorithm to run for five iterations and three minutes in total, for any problem instance. We observed that the first 3-4 iterations of UPR algorithm mostly end in a few seconds since earlier iterations reduce the problem size by large proportions. Hence, we set the solve time limit per iteration is set to 30 seconds for the first four iterations and one minute for the last iteration. Lastly, for each problem type and each model, we set the uncertainty threshold $\bar{u}$ to the median uncertainty value of predictions for the validation dataset of the problem. It is reasoned in Section 6.1.

In every 100 B&B nodes, *Guided node selection* strategy of MIP-GNN framework periodically selects the node that provides the best bound rather than the node with the highest confidence score (Khalil et al., 2022). In this way, the dual bound also can be refined. We also follow this rule in our experiments.

5.3 Evaluation Metrics

Selecting appropriate evaluation metrics is paramount for evaluating the problem-solving capabilities of any ML model. This is especially crucial in our case where an ML model is employed within a MIP solver for a downstream optimization task (Khalil et al., 2017; Nair et al., 2020). In such workflows, loss or accuracy metrics associated with model training are deemed inadequate. Instead, data-driven optimization methods are benchmarked based on how they improve the optimization metrics, which are *Optimality Gap*, *Primal Gap*, and *Primal Integral* (lower is better for each metric).

Optimality Gap B&B Algorithm quantifies a primal-dual gap as a relative measure to certify the optimality of obtained solutions. Hence, it is reported as the optimality gap by MIP solvers throughout the search. For the primal (PB) and dual bounds (DB) computed at time t, it is calculated as

$$\text{gap}(t) = \frac{|PB - DB|}{\max\{|PB|, |DB|, \epsilon\}} \times 100, \tag{14}$$

where $PB = \mathbf{c}^\top \tilde{\mathbf{x}}$, $\tilde{\mathbf{x}}$ is the best integral solution found at t, DB is the objective value of the best LP-relaxed solution at t, and ϵ is a small constant to prevent numerical errors if $|PB| = |DB| = 0$. If $|PB| = |DB|$, then $\tilde{\mathbf{x}}$ is the optimal solution.

The optimality gap is resorted to benchmark algorithms if the optimal solution is known for a given problem, which is unlikely for large-scale optimization domains. Besides, primal heuristics improve PB but do not directly tighten DB. Instead, one can refer to the following metric.

Primal Gap It measures the relative difference between the objective value of a feasible solution $\tilde{\mathbf{x}}$ and that of the optimal or best-known solution $\tilde{\mathbf{x}}^*$. It is calculated as

$$\text{p-gap}(\tilde{\mathbf{x}}) = \frac{|\mathbf{c}^\top \tilde{\mathbf{x}} - c^\top \tilde{\mathbf{x}}^*|}{\max\{|\mathbf{c}^\top \tilde{\mathbf{x}}|, |\mathbf{c}^\top \tilde{\mathbf{x}}^*|, \epsilon\}} \times 100. \tag{15}$$

In this study, primal gap values are calculated with respect to the best objective value attained by the CO pipeline configurations (ran for 30 minutes) and the long CPLEX runs (up to 8 hours). In addition to performance evaluation based on the best solution found in a given time limit, the efficiency of primal heuristics can be measured considering the trade-off between solution speed and quality (Berthold, 2013) with the following metric.

Primal Integral It measures the solution quality of an algorithm regarding elapsed solving time. For a given time limit T and the list of incumbent solutions $[\tilde{\mathbf{x}}^{(i)}, \ldots, \tilde{\mathbf{x}}^{(I)}]$ ordered w.r.t. their timestamps $t_i \in [0, T]$, it is defined as

$$\text{p-integral}(T) = \sum_{i=1}^{I} \text{p-gap}(\tilde{\mathbf{x}}^{(i)}) \cdot (t_i - t_{i-1}), \tag{16}$$

where I is the number of incumbent solutions and $t_0 = 0$. Note that $\text{p-gap}(\tilde{\mathbf{x}}^{(i+1)}) \leq \text{p-gap}(\tilde{\mathbf{x}}^{(i)})$. In our study, we calculate $(t_i - t_{i-1})$ in terms of seconds and normalize p-integral(T) by the solve time limit, 1800 seconds.

We principally refer to the primal gap for benchmarking the models and consider primal integral for evaluating tie-cases. Still, we report all three metrics throughout the sections. Note that primal heuristics do not guarantee refinement of the optimality gap as they only improve the primal bound. Nevertheless, we remark that the optimality gap results in our experimentation generally led to consistent conclusions on ranking the models with respect to the primal gap.

For the benchmarking and ablation study of the models, we also calculate an average improvement rate metric, where a higher value indicates better performance. It quantifies how much a model (or a set of models) improves an optimization metric relative to the performance of another model (or another set of models) or CPLEX on average. For example, the average improvement rate by a model over CPLEX for a metric ρ (e.g., primal gap) is calculated as $100 * \frac{\rho_{cplex} - \rho_{model}}{\rho_{cplex}}$, where ρ_{cplex} and ρ_{model} are average measures of ρ provided by CPLEX and the model, respectively.

For the measuring the accuracy of the predicted solutions, we treat the best solutions obtained through the long CPLEX runs as the ground truth. However, it is worth noting that our models are parameterized with the optimal solutions to the training instances. Hence, the best solutions for the transfer datasets are suboptimal, they do not represent actual target solutions to measure the predictive performance of the models precisely[9].

6. Evaluation

We present and discuss the experimental results covering each design aspect of our study in respective subsections. In Section 6.1, we empirically validate the generalization power of the models trained on downscaled instances to predict solutions for larger-scale instances. Afterward, we benchmark our methodology implemented through different CO pipeline configurations in Section 6.2 with CPLEX as the baseline solver and in Section 6.3 with the pre-trained MIP-GNN models. Moreover, we present an ablation study to analyze the effect

9. Even though the optimal solutions for all datasets were at hand, computing the accuracy of the predicted solutions precisely would require considering that alternate optima can be present.

of each CO pipeline setting in the particular sections 6.4, 6.5, 6.6, and 6.7. These sections investigate how much each pipeline configuration facilities solving the benchmarking CO problems and can highlight the advantage of different model configurations on different problems.

We evaluate our methodology mainly on the largest-scale transfer dataset of each problem since the performance differences of the CO pipelines become more prominent as the problem size increases. Nevertheless, Appendix D includes the figures presenting the performance of all models for each scale of the problems. Since we set the scale of training instances to be easily solvable with CPLEX, all CO pipelines generally achieve the optimal solutions for testing instances and provide optimal gap and primal gap values near zero, like in $[0, 0.005]$. Therefore, we consider only primal integral values for evaluation over testing datasets.

In this section, we present the boxplot distributions of the results by each CO pipeline configuration (referred as only 'configuration' from here on) defined in Section 5.3 for 50 largest-scale transfer instances of each CO problem. Figure 3, 4 and 5 present the boxplot distributions of each metric attained by each configuration with NS and UPR+NS strategies in 30 minutes. Figure 6 present the distributions of primal gap values of the warm-start solutions found with UPR. In all figures, the boxplot distributions of metrics for CPLEX are placed together with the CO pipelines applying only NS strategy. Note that the results of the pre-trained MIP-GNN models are only available for FCMNF and GIS datasets since the study of Khalil et al. covers these datasets.

The optimality gap and primal gap distributions in Figures 3 and 4 set out the contribution of the CO pipelines to solve each problem at scale by wide margins compared to CPLEX. For MSC, CA, MIS, and FCMNF problems, the stages of the proposed primal heuristics are the major factor in boosting optimization performance, while the impact of model configuration is more dominant for GIS problem. Yet, we observe that EC+V+EDL mostly leads to better optimization performance across five problems. Besides, the benchmarking with the pre-trained MIP-GNN model figures out the advantage of training the models over the downscaled instances. When only NS strategy is applied, the CO pipelines using our models achieve on-par or better optimization performance for FCMNF and GIS datasets, respectively. On the other hand, UPR+NS pipelines cannot benefit from the warm-start solutions generated based on the predictions of the MIP-GNN models. Since UPR algorithm heavily depends on the partial solutions that are feasible and do not hinder optimality, it yields low-quality warm-start solutions using the MIP-GNN models (see Figure 6), which is further detailed in Section 6.3. In addition to the final solution quality, the CO pipelines maintain time-efficient solution quality throughout the run time with fairly lower primal integral values, according to Figure 5.

Lastly, Figure 7 exposes how each configuration proceeds in solving each problem. The primal gap values by most of the CO pipelines quickly drop in earlier minutes, which depicts better the efficiency of the proposed primal heuristics leveraging the trained models within B&B algorithm. We further detail how each configuration is effective for solving each problem in the following subsections.

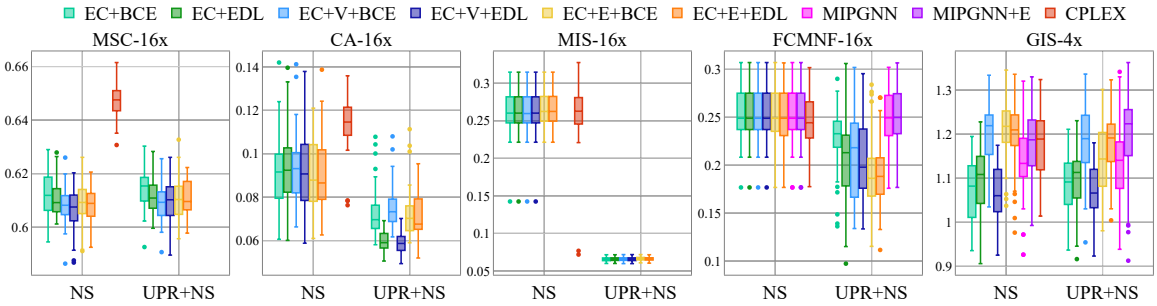


Figure 3: Optimality Gap Distributions for the Largest Scale Transfer Datasets

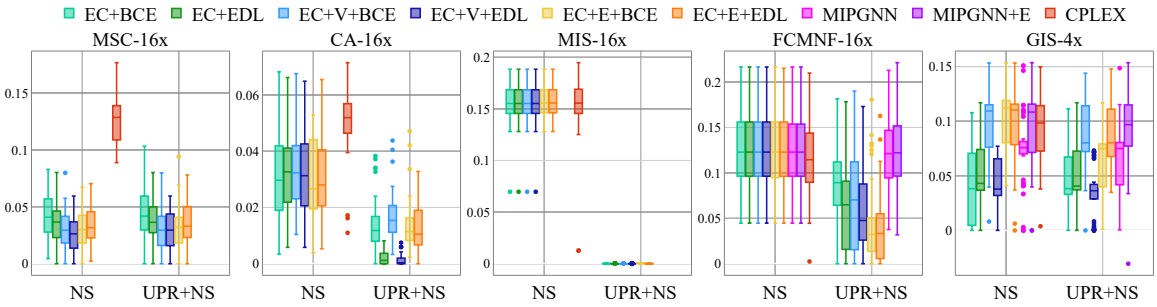


Figure 4: Primal Gap Distributions for the Largest Scale Transfer Datasets

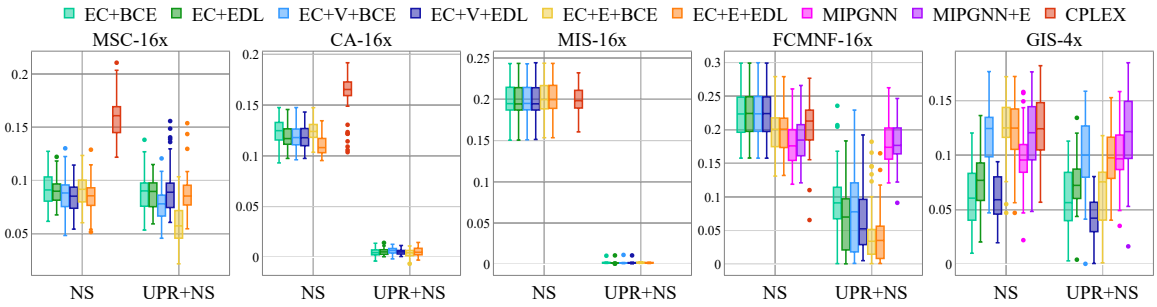


Figure 5: Primal Integral Distributions for the Largest Scale Transfer Datasets

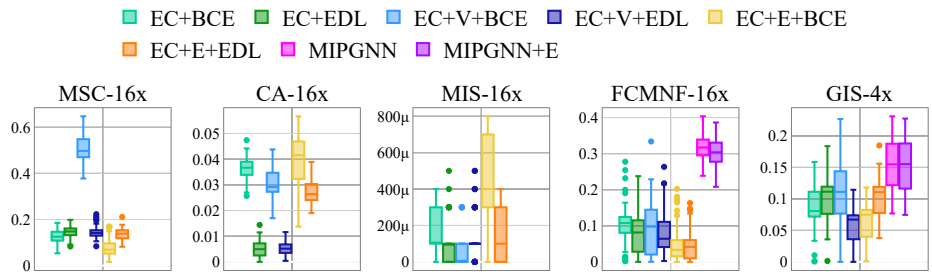


Figure 6: Primal Gap Distributions of the Warm-start Solutions Found with UPR for the Largest Scale Transfer Datasets

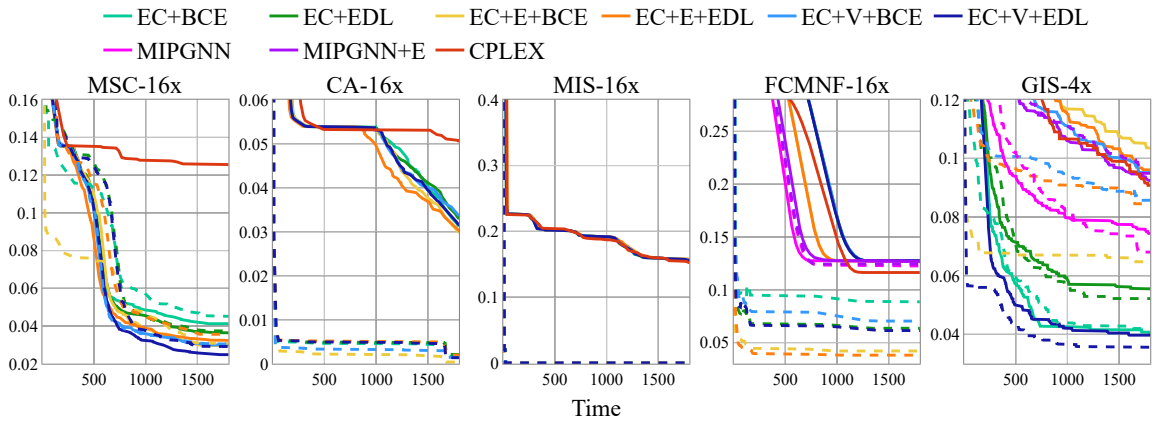


Figure 7: Average Primal Gap Curves for the Largest Scale Transfer Datasets. NS and UPR+NS curves are denoted by solid and dashed lines, respectively. To present the primal gap value averaged across all instances per second by a CO pipeline, primal gap values are linearly interpolated through the time interval $[1, 1800]$.

6.1 Generalization on Larger-scale CO Problems

Before evaluating the optimization performance of the CO pipelines, we investigate whether the proposed modelling approach can achieve a predictive performance on large-scale CO problems as accurately as the testing performance on the downscaled problems. For the evaluation in this section, we additionally trained EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models without the normalization techniques proposed in Section 4.1.1 to reveal how these techniques improve predictive accuracy.

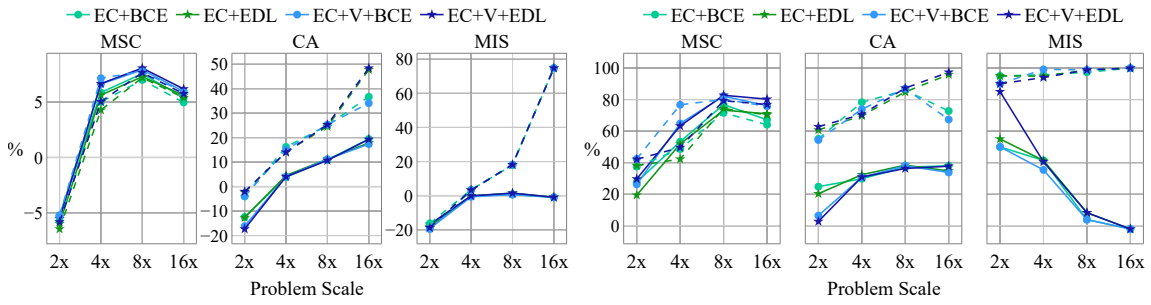
We take into account the accuracy of predictions and their associated uncertainty values together when evaluating the generalization ability of trained models on larger datasets. By sorting the predictions in ascending order of their associated uncertainty values, we assess the accuracy of each percentile of predictions. Figure 12 presents the average accuracy of EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models with and without the proposed normalization techniques (by solid and dashed curves, respectively) in each ten percentile of the associated uncertainty values for all datasets. Particularly, the quality of the best solutions (treated as the ground truth for measuring the prediction accuracy) obtained with CPLEX in eight hours for MIS-16x is worse than that of the warm-start solutions generated with UPR. It indicates that these best solutions are poor to represent the ground truth. Therefore, the prediction accuracy curves for MIS-16x dataset in Figure 12 get lowers slightly.

The models without the normalization layers lead to misleading predictions with high confidence for MIS, FCMNF, and GIS problems; thus, their predictions are much less credible for the proposed primal heuristics for solving these problems compared to our standard models (ones including the normalization layers). Another important motivation for the proposed primal heuristics is that accuracy of the most confident predictions by the models should be near 100% (recall Section 4.2.1). In the following, we will only refer to the standard models and discuss our methodology based on their predictive performance.

Considering the confident predictions, we generally observe that the testing and transfer accuracy of models match. Moreover, the uncertain predictions for the transfer datasets may cause slightly diverging accuracy levels, but not lower than the testing levels mostly. These results confirm that the proposed modeling approach enhances the knowledge transfer capability of GNNs for inferring solutions to larger-scale instances of the benchmarking CO problems. Furthermore, it is observed that the accuracy level within the first 50 percentile of uncertainty distributions is generally near 100% for the testing and the transfer datasets by all models. Hence, we conclude that the uncertainty values are admissible to utilize them in the proposed primal heuristic strategies. This empirical outcome brings an admissible value for the uncertainty threshold $\bar{u}$ to use in UPR algorithm. For each model of each problem, we set $\bar{u}$ to the median of uncertainty values of the predictions for the validation dataset of the problem and use it for both testing and transfer datasets of the problem. On the other hand, we reveal an exceptional result for the admissibility of uncertainty values by EC+V+BCE model for MSC, FCMNF, and GIS problems in Appendix D. Since the predictions of this model are misleading, its CO pipelines underperform in terms decision quality, which is revealed in the next subsections.

6.2 Benchmarking with CPLEX

Figure 8 presents the performance improvement by the CO pipelines for optimality gap and primal gap metrics relative to the performance of CPLEX across the problem scales of MSC, CA, and MIS transfer datasets. Similarly, Figure 9 includes the bar charts of the improvement rates for the FCMNF and GIS transfer datasets.

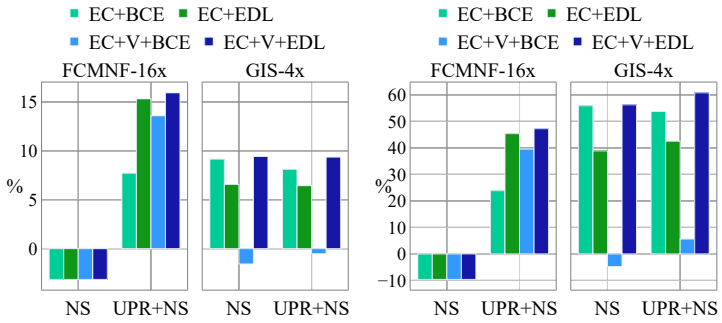


(a) Imp. for Optimality Gap

(b) Imp. for Primal Gap

Figure 8: Average Improvement Rates (Imp.) by the Model Configurations for the Transfer Datasets of MSC, CA, and MIS Problems (NS and UPR+NS results are denoted by solid and dashed lines, respectively.)

Solving MSC and CA problems, both NS and UPR+NS strategies substantially decrease primal gap values by around 80% as the problem size increases. Differently for MIS problem, the benefit of NS strategy fades. It is clarified from Figure 7 where all model configurations with NS strategy reach similar average primal gaps with CPLEX's values while solving MIS-16x problems. On the side of UPR+NS, UPR algorithm boosts solving MIS problem since the warm-start solutions found based on the model predictions lead to immediately reaching around a 6.5% optimality gap. Overall, the non-decreasing pattern of improvement rates



(a) Imp. for Optimality Gap (b) Imp. for Primal Gap

Figure 9: Average Improvement Rates (Imp.) by the Model Configurations for the FCMNF-16x and GIS-4x Datasets

across the scales of MSC, CA, and MIS problems displays that our models offer a scalable performance compared to CPLEX.

The proposed primal heuristics can scale in solving the actual-scale instances of FCMNF and GIS, too, which are considered the more complex CO problems. Khalil et al. report that the testing performance of MIP-GNN framework with their node selection strategy in terms of the optimality gap is slightly worse than CPLEX. Our findings with the NS pipelines are in line with this finding, too. Nevertheless, utilizing the warm-start solutions by UPR algorithm in solving FCMNF-16x instances leads to a much better performance than CPLEX. 40-47% better (i.e., lower) average primal gap for FMCNF-16x can be achieved by deploying UPR+NS strategy with the trained models. For GIS problem, the proposed heuristic strategies decrease the primal gap for GIS-4x instances by 38-60% on average, while EC+V+BCE is the only model showing poor performance improvement since its confident predictions lead to misleading decisions for the transfer instances of GIS problem (see Figure 12).

In addition to benchmarking the performance of the CO pipelines and CPLEX in terms of the evaluation metrics, we analyze the time efficiency of our methodology in Appendix D.1. Consequently, the time savings by the CO pipelines dramatically rise as the problem size increases. The increasing efficiency trends across the scales of all five problems demonstrate that the proposed methodology offers scalable GNN-based primal heuristics.

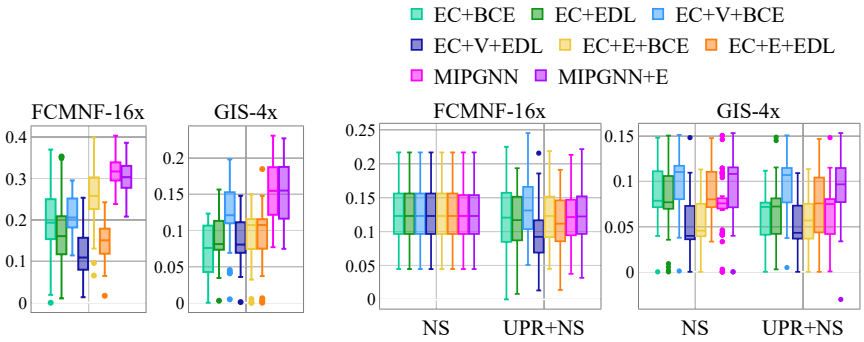
6.3 Benchmarking with MIP-GNN

To benchmark with the pre-trained MIP-GNN models for FCMNF and GIS problems, we additionally trained our models with the same architecture hyperparameters in the study of Khalil et al. (2022). These models use four EC layers, a 4-layered MLP classifier, `mean` aggregation, and have a hidden dimension of 64. Still, the normalization changes to MIP-GNN (proposed in Section 4.1.1) are kept. MIP-GNN models are trained with BCE Loss and use the class probabilities to calculate the confidence scores for B&B nodes in the study of Khalil et al.. Therefore, in this section, we principally consider EC+BCE and EC+E+BCE models used to benchmark with MIPGNN and MIPGNN+E. Note that the instances used for training the MIP-GNN models are at the scale of FCMNF-16x and GIS-

4x instances. Indeed, the transfer performance of our models is compared with the testing performance of MIP-GNN models.

Figure 10b shows that NS pipelines using our models attain median primal gap and optimality gap values on par with the MIP-GNN models for FCMNF problem. For GIS problem, EC+BCE performs much better than the MIP-GNN models, while EC+E+BCE is slightly outperformed by the others. Here, BCE Loss function causes worsening transfer performance for EC+E and EC+V models. Instead, the models trained with EDL offer the top performance among all models. This result lights on the value of uncertainty quantification with EDL for GIS problems.

It is observed from Figure 10a that UPR algorithm cannot reach high-quality solutions based on the predictions of the MIP-GNN predictions. Particularly for FCMNF-16x dataset, any feasible sub-MIP had not been obtained for any instance with UPR algorithm (then, all of them were repaired with Algorithm 2). As discussed in Section 3.2, these findings corroborate that using multiple solutions per instance to set the target space may prevent learning solution patterns. Even the most confident predictions of MIP-GNN models as a whole are not convenient for getting valid partial solutions, which is in line with the conclusions of Han et al. (2023) about the variable fixing strategy of Neural Diving. In contrast, the utilization of UPR algorithm is prominent in finding high-quality warm-start solutions if the predictions of our models are used.

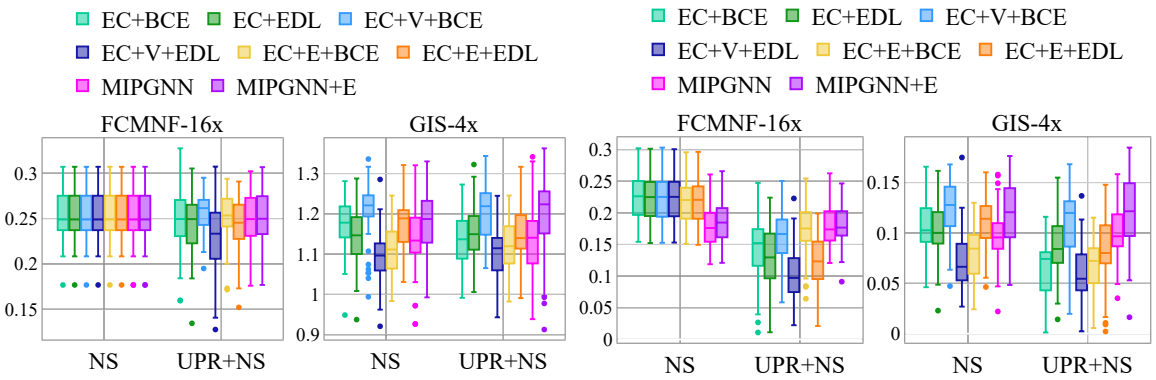


(a) The Primal Gap Distributions of Warmstart Solutions Found with UPR

(b) The Primal Gap Distributions of Final Solutions

Figure 10: Primal Gap Distributions of Warmstart (10a) and Final (10b) Solutions by the CO Pipelines Using 4-layered GNNs and `mean` aggregation for FCMNF-16x and GIS-4x Transfer Datasets

The performance gain through EDL models compared to MIP-GNN models is also reflected in optimality gap and primal integral values presented by Figure 11. Consequently, our models' transfer performance can achieve and even exceed the MIP-GNN models' testing performance. Table 8 in Appendix C shows that the proposed improvements over MIP-GNN framework provide either better or statistically indifferent performance in terms of primal gap. This benchmarking validates that the proposed training regime and architectural improvements can enable GNNs to generalize on the actual-scale instances of FCMNF and GIS problems. Moreover, our methodology requires much less costly training dataset



(a) Optimality Gap Values

(b) Primal Integral Values

Figure 11: Optimality Gap and Primal Integral Distributions by the CO Pipelines Using the Models with 4-layered GNN and `mean` aggregation, and MIP-GNN models for FCMNF-16x and GIS-4x Transfer Datasets

generation and model training[10] as well as less fine-tuning burden (due to having fewer hyperparameters) compared to the MIP-GNN framework, which includes a bias threshold for training and exploiting the models, and several hyperparameters for training data collection (see Section 3.2).

6.4 Effect of Warm-start Solutions

To analyze how the warm-start solutions contribute to solve CO problems at scale, we set separate CO pipelines by ablating UPR algorithm from the proposed primal heuristic strategy. Considering the primal gap distributions in Figure 4, the warm-start solutions obtained with UPR algorithm are highly influential in solving CA, MIS, and FCMNF problems. Figure 7 reveals that UPR algorithm directly leads to finding high-quality solutions by requiring much less effort with B&B for these three problems. This immense time efficiency in reaching high-quality solutions is quantified in Table 9b relative to the default CPLEX performance. Thus, UPR+NS pipelines can also bring substantially lower optimality gaps for these problems by exploiting 30 minutes solve time better, as seen in Figure 3, compared to NS pipelines and the default CPLEX configuration. On the other hand, for MSC and GIS problems, the quality of the warm-start solutions is influential in the stage of global search with only a few CO pipelines. As observed from the primal gap curves for MSC and GIS problems depicted in Figure 7, NS pipelines can immediately achieve a close solution quality with UPR+NS pipelines, which indicates that NS strategy can also be effective in finding high-quality solutions in a short time for these problems. Although UPR+NS

10. We cannot make a direct comparison with the experimentation in the study of Khalil et al. (2022) for the elapsed times in the training dataset generation and the model training since the experiment environments are different. Yet, we can specify that the median time for obtaining the optimal solutions to the downscaled training instances of FCMNF and GIS problems are respectively 142.36 and 1439.2 seconds (see Table 5 at Appendix B) whereas Khalil et al. set one hour time-limit for generating a pool of solutions per training instance of these problems. Besides, considering the number of decision variables and constraints in the training instances, the proposed training regime in our study certainly takes much less time.

pipelines lead to 0.001-0.005 higher primal gap values on average for MSC-16x instances compared to those by NS pipelines, these differences are insignificant ($p > 0.05$) according to the t-Test for each model configuration deployed in NS and UPR+NS pipelines.

6.5 Effect of Violation Layer

We measure how Violation Layer improves MIP-GNN model for solving larger-scale instances by comparing it with the models using Error Layer (EC+E) and the bare models (EC). Compared to the results of EC and EC+E models in Figure 4, the primal gap values by the EC+V models are slightly lower for MSC-16x with UPR+NS strategy, significantly lower for CA-16x, and similar for MIS-16x instances. On the other hand, the impact of Violation Layer and Error Layer are divergent for solving GIS-4x and FCMNF-16x datasets. While EC+E models offer much better performance with UPR+NS strategy for FCMNF-16x instances, they significantly underperform by a wide margin for GIS-4x instances. Both NS and UPR+NS pipelines using the EC+V models trained with BCE Loss show a degrading performance in solving GIS-4x instances. This particular result among all datasets indicates that using the class probabilities of BCE models might present non-eligible quantification of prediction uncertainty, which was pointed out in Section 4.2. Lastly, it is noteworthy that the explicit performance gain through Violation Layer depends on the warm-start solutions acquired in UPR algorithm using EC+V models (especially EC+V+EDL models), which are much better for CA-16x, MIS-16x, and GIS-4x datasets, whereas EC+E models are slightly more beneficial for MSC-16x and FCMNF-16x datasets (see Figure 6). These results indicate that Violation Layer drives capturing solution patterns better for generalization on larger-scale instances and leads to more robust performance in the warm-starting.

6.6 Effect of Uncertainty Quantification

To assess the impact of uncertainty quantification through EDL on the performance of the proposed primal heuristic strategy, Table 2 presents the improvement rates for each metric achieved by models trained with EDL Loss compared to those trained with BCE Loss. While some CO pipelines employing BCE models yield slightly lower primal integral values in comparison to those using EDL models for test instances of CA, MIS, FCMNF, and GIS problems, the gains of uncertainty quantification become particularly notable for transfer datasets of MSC, CA, FCMNF, and GIS problems.

Solving the transfer datasets of MSC and GIS problems results in much higher optimality gaps (with the median values of 64.8% and 118.9% by CPLEX) compared to other problems, which shows finding tight primal and dual bounds is harder for these two problems. A notable observation is that EDL models lead to reducing the primal gap explicitly for these two problems compared to BCE models. This finding corroborates that quantification of prediction uncertainties produces eligible node confidence scores to improve the primal bound. Moreover, the performance improvement by EDL models in solving transfer-16x instances of CA and FCMNF problems is more remarkable with UPR algorithm for the proposed primal heuristics. The improvement rates for the primal gap of the warm-start solutions underlie that the ranking of predictions with respect to associated uncertainty values is more useful to get better partial solutions within UPR algorithm when EDL

models are deployed. Accordingly, confident predictions of EDL models are more credible for retaining the optimality, thus driving the algorithm to find better solutions.

Table 2: Average improvement rates (Imp.) by the EDL models over the BCE models for the testing dataset and the largest-scale transfer dataset of each problem. The improvement rates are averaged across EC+V and EC models. The optimality gap and primal gap improvement rates for the test datasets are excluded since all models achieve (near) zero gap values. The negative percentages indicate that the CO pipelines of EDL models underperformed.

Problem	Strategy	(Test) Primal Integral Imp. (%)	(Transfer) Primal Integral Imp. (%)	(Transfer) Primal Gap Imp. (%)	(Transfer) Optimality Gap Imp. (%)	(Transfer) Warm-starting Primal Gap Imp. (%)
MSC	NS	44.64	2.57	13.99	0.24	-
	UPR+NS	43.11	-8.05	11.96	0.17	54.23
CA	NS	-18.74	1.94	0.55	0.21	-
	UPR+NS	16.70	9.58	88.57	19.66	84.50
MIS	NS	-1.46	0.05	0.00	0.02	-
	UPR+NS	3.86	4.84	-2.05	-0.03	-3.07
FCMNF	NS	-4.34	0.07	0.00	0.00	-
	UPR+NS	-0.76	21.56	21.39	5.57	19.98
GIS	NS	-0.73	25.07	29.54	4.37	-
	UPR+NS	2.43	27.99	31.24	4.26	18.09

6.7 Effect of Combined Aggregation

This section presents the ablation study of the aggregation operator used in EC layers, which is the combined aggregation (`comb`) introduced in Section 4.1.3. To measure the effect of `comb` on generalization to larger-scale instances, we also trained separate EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models that use `mean` aggregation as the commonly used one in the related studies. We benchmarked these two groups of models over the testing and the largest-scale transfer datasets of each problem. Table 3 presents the improvement rates by the models using `comb` over those using `mean` in terms of each metric.

The `comb` models generally demonstrate superior generalization in solving CA, MIS, and FCMNF problems compared to the `mean` models. The positive the improvement rates with NS strategy present evidence that the models using `comb` predicts values for decision variables by exposing better-calibrated uncertainty values with EDL Loss or well-grounded class probabilities with BCE Loss. Upon applying UPR+NS strategy, the optimization performance by the `comb` pipelines increases by a wide margin. This result indicates that these models can learn graph properties that are useful for solving CO problems; therefore, high-quality warm-start solutions can be obtained in UPR with the predictions of the `comb` models (see Table 7 at Appendix C).

Using `comb` degrades the generalization performance for MSC-16x and GIS-4x datasets, unlike the positive improvement rates for the other three datasets. Although the `comb` pipelines provide better testing performance, the `mean` pipelines generalize on solving larger-scale instances of MSC better. The CO pipelines using BCE models cannot benefit `comb` for solving GIS-4x while those with EDL models can. It indicates that EDL can lead the regularization of the models that use `comb` for GIS problem. However, training models on the downscaled instances of MSC and GIS problems with complex aggregations to require

Table 3: Average improvement rates (Imp.) by the models using `comb` over those using `mean` for the testing and the largest-scale transfer datasets of each problems. The improvement rates are averaged across EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models. The optimality gap and primal gap improvement rates for the test datasets are excluded since all models achieve (near) zero gap values. The negative percentages indicate that the CO pipelines of the models using `comb` underperformed.

Problem	Strategy	Loss Function	(Test) Primal Integral Imp. (%)	(Transfer) Primal Integral Imp. (%)	(Transfer) Primal Gap Imp. (%)	(Transfer) Optimality Gap Imp. (%)
MSC	NS	BCE	3.69	-7.37	-44.98	-0.46
		EDL	44.21	-14.85	-70.14	-0.72
	UPR+NS	BCE	0.52	-42.09	-82.89	-0.93
		EDL	43.09	-19.68	-68.58	-0.73
CA	NS	BCE	31.74	-12.93	-20.77	-7.24
		EDL	7.40	11.60	15.82	6.99
	UPR+NS	BCE	-12.81	81.46	-0.18	0.19
		EDL	-0.21	83.96	90.21	22.76
MIS	NS	BCE	0.86	0.13	1.26	1.09
		EDL	6.90	0.20	1.26	1.09
	UPR+NS	BCE	51.90	-1.41	53.37	0.33
		EDL	51.82	2.22	-21.97	0.17
FCMNF	NS	BCE	3.34	-5.29	0.00	0.00
		EDL	-9.60	0.30	-0.56	-0.18
	UPR+NS	BCE	20.87	46.77	34.76	12.15
		EDL	17.61	21.36	16.26	4.07
GIS	NS	BCE	1.72	-66.62	-80.51	-9.61
		EDL	4.48	8.62	8.41	-1.80
	UPR+NS	BCE	9.01	-93.49	-98.97	-9.90
		EDL	19.04	5.00	7.22	-1.94

additional regularization. This could be resolved by narrower hidden layers, dropout layers in MPNNs, or increasing the dropout probability for the classifier network layers. In summary, our ablation study in this section reveals that GNNs combining various aggregation operators can learn valuable patterns for solving CO problems. Nonetheless, it suggests that stronger regularization may be necessary to ensure performance improvement.

7. Discussion

In this section, we discuss the generalization capability and limitations of the proposed methodology for scaling CO problem solutions. Determining the required model complexity for generalizing any CO problem is challenging due to the dependence on the problem structure, making precise complexity estimates difficult. For generating the training datasets, we scaled the instances to a level that CPLEX can solve exactly within a reasonable time (refer to Table 5 in the Appendix for the median solve time for the training instances). We did not conduct an experiment to analyze the effect of magnitude of downscaling training instances on solving larger scale instances. However, our experimentation demonstrates exemplary optimization performance for FCMNF and GIS problems when downscaled by 16 and 4 times, respectively, in benchmarking with MIP-GNN models. A more advanced design of the proposed CO scaling methodology could determine the most efficient level of the downscaling by fine-tuning it for each problem separately through end-to-end optimization.

Two factors affect the learning of solution patterns in downscaled instances, determining the benefit of inferring solutions for larger instances. Firstly, the interdependency and interrelation of decision variables in a problem may extend as the problem size increases[11]. The benefit of learned solution patterns for solving a given problem can weaken as the scale of training instances reduces. Therefore, it is important to tune the magnitude of downscaling by analyzing its effect on the downstream optimization task, considering the cost of collecting optimal solutions to training instances for a given problem. Another factor is the depth of GNN: How deep should GNN models be so that they can also capture long-range relations among decision variables of larger instances? Deeper GNNs would exploit more comprehensive relations among a high number of decision variables. On the other hand, training deeper GNNs on small-scale CO problem instances can be useful to capture such relations up to a certain extent since the relations among a small number of decision variables are likely to fade out of a certain neighborhood depth. So, the depth of GNN can be fine-tuned considering the tension between the extend of relations within decision variables of a small-scale instance and the benefit of capturing wider patterns to extract useful information for solving larger problems.

8. Conclusion

This study presents a comprehensive overview of end-to-end learning with GNNs to efficiently address CO problems using primal heuristics and particularly focuses on the scalability of this methodology. The proposed training approach and enhancements to MIP-GNN model facilitate knowledge transfer for solving larger-scale instances. The proposed primal heuristics leverage the trained models by generating warm-start solutions and guiding B&B search based on predictions and calibrated model uncertainty through EDL. This methodology provides substantially improved performance in solving five CO problems compared to CPLEX as the problem size increases. The benchmarking with the pre-trained MIP-GNN models demonstrates that the proposed training regime and architectural changes drive GNNs to capture patterns that are more useful in solving larger scale instances. Besides, the proposed methodology requires less costly training data generation and comes with less burden of hyperparameter tuning. Nevertheless, one could do further fine-tuning by using different evidence functions and proposing different regularization approaches or resort to other uncertainty quantification techniques (Abdar et al., 2021). Also, a dynamic approach to compute the confidence scores for B&B nodes could be adopted to control the exploitation of the predicted decision values and the exploration of other promising regions according to the estimated bounds by B&B algorithm.

As a future work, the proposed methodology can be optimized for a given problem by an analysis regarding (i) the extent of relations and interdependency among decision variables regarding problem size, (ii) the cost of collecting optimal solutions to the downscaled instances, (iii) the cost of training deeper GNNs, and (iv) the benefit of the learned prob-

11. This condition is not generalized for all problem domains. Some problems can pose a local nature or decomposable structure which displays interdependency within only subgroups of decision variables. Therefore, learned narrow-ranged patterns can be stably valuable for solving arbitrarily large instances of a problem. The experimentation for MIS problem gives results in line with it. The warm-start solutions generated based on the predictions of the trained models lead to an immediate convergence in the primal gap for even 16x larger instances (see Figure 7).

lem/solution patterns in depth by the models for the downstream optimization of actual-scale instances of the problem. Considering the terms and characteristics of application domains, this exhaustive analysis could make the best use of the proposed methodology to solve CO problems at scale. Aligned with the motivation of our study, a potential research direction could be promising by advancing curriculum learning approaches, which aim to train models to transfer knowledge from solving easy CO problems to harder ones (Zhou et al., 2023; Lisicki et al., 2020). Last but not least, our methodology could be extended for solving mixed-integer linear programming (MILP) problems. Nair et al. (2020) and Khalil et al. (2022) present different approaches to how the proposed GNNs can be generalized to predict values of non-binary integer variables. Nevertheless, these approaches require further development for scalability, given that an arbitrary MILP can involve complex constraint structures and large domains for both integer and continuous decision variables. These problem characteristics could make capturing solution patterns within complex problem structures more challenging; therefore, devising different learning approaches and more complex GNN architectures are needed.

8.1 Broad Impact

CO algorithms serve numerous industries, such as energy, logistics, supply chain, and finance. However, designing efficient, scalable, and robust CO algorithms requires substantial time and domain knowledge. By automatizing algorithm designs by learning problem structures and solution patterns and adapting better to given specific problem data, GNN-based CO approaches could enable the democratization of high-performing algorithms, leading to improved efficiency in these industrial domains.

Our proposed methodology demonstrates significant potential both in improved scalability and solution quality, which show a potential path to achieve superior GNN-augmented solvers. Embedding pre-trained GNNs into fully-fledged MIP solvers could enhance the solving of CO problems across various scientific fields, including applied mathematics, operations research, computer science, and economics. Moreover, empowering next-generation MIP solvers with GPU-based matrix computations could lead to widespread use of them in large-scale optimization applications, particularly in real-time decision domains. Besides, the proposed methodology demand much less CPU and GPU workloads while scaling. It can reduce maintenance costs of ML-augmented real-world applications considering dynamic real-world problems since they would require the continual collection of solutions for retraining/fine-tuning the predictive models in order to keep the generalization power steady. Lastly, such intelligent systems crafted for optimization tasks should provide descriptive analytics and explainability techniques alongside their predictive models. Otherwise, the output of the system could drive decisions lacking trust and reasoning behind them, which brings the risk of extreme sub-optimality or even infeasibility in the nature of evolving real-world problems. The proposed uncertainty quantification-based approach intrinsically works with the sense of trust toward the trained models (Soleimany et al., 2021). Nevertheless, one could refer to emerging research on GNNs developing further interpretable methods for different aspects of explainability like robustness, accountability, and fairness (Dai et al., 2022; Yuan et al., 2020). Thus, adopting explainable intelligent optimization approaches would help in understanding system dynamics and the reasoning

behind automated decisions, which is solicited in high-stakes domains such as chemistry, health, security, and finance.

Acknowledgments

The authors would like to thank Mehmet Gönen, Milad Elyasi, and reviewers for their valuable feedback and comments. Furkan Cantürk was partially supported by The Scientific and Research Council of Türkiye (TÜBİTAK) through 2210 - National Scholarship Programme for M.Sc. Students and the project grant 120N680. Taha Varol was supported by TÜBİTAK with the project grant 120E488.

Appendix A. A Repair Algorithm for Infeasible MIP

We propose Algorithm 2 to restore the infeasible sub-MIP M' at the last iteration of UPR algorithm. It is based on *Conflict Analysis* method (Achterberg, 2007)[12] , which diagnoses conflicts among the constraints in an infeasible MIP. We use this method to select which of the added cuts in UPR algorithm causing the infeasibility.

Algorithm 2: Conflict Analysis-based Repair for Infeasible MIP

Input : Infeasible MIP instance M' with cuts Ω, repair time limit T
Output: Feasible MIP instance M_f

1 $repaired \leftarrow false$ // Initialize a predicate indicating if M' is feasible or not
2 $t = 0$ // Initialize elapsed repair time
3 **while** $t < T$ & $||\Omega|| > 0$ **do**
4 $(\Omega_c, t) \leftarrow \text{ConflictAnalysis}(M', T - t)$
5 **if** $|\Omega_c| = 0$ **then**
6 $repaired \leftarrow true$
7 $M_f \leftarrow M'$
8 **break** // ends the loop
9 **end**
10 $\Omega' \leftarrow \text{GetConflictingCuts}(\Omega_c, \Omega)$
11 $M' \leftarrow RemoveCuts(M', \Omega')$
12 $\Omega \leftarrow \Omega \setminus \Omega'$
13 **end**
14 **if** $repaired = false$ **then**
15 $M_f \leftarrow \text{RemoveCuts}(M', \Omega)$
16 **end**
17 **return** M_f

In each iteration of Algorithm 2, the conflicted constraints in a given MIP are diagnosed by *Conflict Refiner* module of CPLEX. Conflict Analysis procedure in Line 5 is invoked to find a minimum subset of conflicting constraints in M' and returns a tuple of the set of diagnosed constraints Ω_c and the elapsed time t within a given time $T - t$. If M' is

12. Conflict analysis is managed in the off-the-shelf MIP solvers such as Conflict Refiner in CPLEX (IBM ILOG CPLEX, 2021), Conflict Analysis in SCIP (SCIP Optimization Suite, 2024a), and Irreducible Inconsistent Subsystem in Gurobi (Gurobi Optimization, 2024).

feasible, then $\Omega_c = \emptyset$. The variables that appear in both Ω' and Ω_c are detected (Line 10), and then the cuts that fix these variables to the predictions are removed from Ω' (Line 11). This procedure continues within a time-conditioned loop until a feasible MIP instance M_f is obtained (Line 4) or all of the variable fixing cuts are eliminated ($|\Omega| = 0$). If M_f cannot be obtained (the worst case), all the remaining cuts are removed from M_f (Line 15), which results in recovering the original MIP. After repairing an infeasible sub-MIP using Algorithm 2, UPR algorithm once again runs for one minute to obtain a warm-start solution from the restored sub-MIP M_f .

In our experimentation, UPR algorithm achieved feasible sub-MIPs for all instances of all datasets except FCMNF-16x. Table 4 presents the numbers of last iteration sub-MIPs ($M_{sub}^{(N)}$) that are infeasible for FCMNF-16x instances. For all these instances, Algorithm 2 successfully restores each sub-MIP in one minute; afterwards, UPR algorithm achieves a feasible solution for it.

Table 4: The number of instances in which a feasible sub-MIP is not obtained in UPR algorithm for 50 FCMNF-16x instances. The model configurations are classified based on the loss functions, aggregation types (`comb` and `mean`), and the number of GNN layers (k).

Model	**Loss Function**	`comb`		`mean`	
		k=4	**k=8**	**k=4**	**k=8**
EC	BCE	29	0	13	13
	EDL	3	3	2	11
EC+E	BCE	50	0	17	31
	EDL	11	0	0	1
EC+V	BCE	35	1	34	9
	EDL	0	4	0	0
MIPGNN	BCE	-	-	50	-
MIPGNN+E	BCE	-	-	50	-

Appendix B. Benchmarking Datasets and Training Data Generation

This section detail training data generation procedures for the benchmarking datasets in this study. Readers can refer to the studies of Ding et al. (2020) and Khalil et al. (2022) for the definitions of the problems. We used the dataset generators released by Gasse et al. (2019) for MSC, CA, and MIS problems[13] and Khalil et al. for FCMNF and GIS problems[14]. The original MSC dataset includes instances with the objective coefficients that are randomly distributed in $[0, 100)$. Due to the range and the distribution type, CPLEX can solve any instance of the dataset within seconds, even if the number of variables and constraints in the problem is increased. Therefore, after generating MSC instances, we modified them by assigning the objective coefficients to be in $[5, 100)$ at random, which makes the problem harder and not trivially solved with CPLEX. A fourth dataset in the study of Gasse et al. (2019) includes *Capacitated Facility Location Problem* instances which are also trivially solved by CPLEX for any combination of the problem parameters (number of customers, number of facilities, and the ratio of facility capacities to demand of customers) and problem size. Thus, we did not include it in our experimentation.

13. `https://github.com/ds4dm/learn2branch-ecole/tree/dev`
14. `https://github.com/lyeskhalil/mipGNN`

The scale of MSC, CA, and MIS problems experimented by Gasse et al. (2019), Ding et al. (2020) and Han et al. (2023) can be solved in a few seconds with CPLEX. Instead, we set the scale of problems at larger magnitudes, which are presented in Table 5 as the number of decision variables. Recall that the scale of instances in the training, validation, and testing datasets is denoted by 1x. The number of decision variables and constraints in the transfer instances of each problem is certain multiples of the values in the table. For example, each MSC-16x instance has 16000 decision variables and 8000 constraints.

Table 5: The median and standard deviation of the number of variables, the number of constraints, and CPLEX solve time per instance (with a single thread) for the training dataset (1x) of each problem.

	MSC	**CA**	**MIS**	**FCMNF**	**GIS**
# Variables	1000.0 ± 0.0	1000.0 ± 0.0	1000.0 ± 0.0	5200.0 ± 0.0	5341.0 ± 36.59
# Constraints	500.0 ± 0.0	388.0 ± 4.61	4223.0 ± 13.96	1375.0 ± 0.0	6963.0 ± 0.0
Solve Time (sec)	468.52 ± 713.53	74.89 ± 167.53	9.8 ± 197.96	142.36 ± 237.1	1439.2 ± 644.23

Table 6: The average and standard deviation of the inference time (in seconds), which is the sum of elapsed time in the data preprocessing and prediction, for each dataset.

	Test (1x)	**Transfer (2x)**	**Transfer (4x)**	**Transfer (8x)**	**Transfer (16x)**
MSC	1.94 ± 0.85	3.06 ± 1.21	11.62 ± 5.09	26.7 ± 9.53	71.16 ± 52.41
CA	1.47 ± 0.51	2.05 ± 0.71	2.58 ± 0.85	3.17 ± 1.08	5.21 ± 1.54
MIS	1.58 ± 0.51	2.2 ± 0.71	2.99 ± 0.97	5.42 ± 1.81	9.43 ± 2.68
FCMNF	2.38 ± 0.92	-	-	-	15.14 ± 6.37
GIS	2.58 ± 1.09	-	6.47 ± 2.67	-	-

To obtain downscaled instances of FCMNF problem for our training dataset, we set the number of nodes in the randomly generated Erdos-Renyi graph to 50 and the number of commodities to 25. To acquire a similar ratio of the number of edges to the nodes with the actual FCMNF dataset, we set the Erdos-Renyi edge probability to 0.075. Other problem parameters are kept the same. As a result, our training FCMNF instances includes approximately 1/16th the number of decision variables in the actual FCMNF problems.

10 different GIS datasets are covered in the study of Khalil et al.. The GIS datasets named 'C125.9.clq' and 'C250.9.clq' follow a common distribution of problem parameters, but the latter dataset includes 4x the number of variables of the first one. These datasets are, respectively the easiest and most challenging datasets according to their optimality gap results in the study. Using these two datasets in our study, we train our models only on 'C125.9.clq' instances and also benchmark them on 'C250.9.clq' instances with the MIP-GNN models separately pre-trained on 'C125.9.clq' and 'C250.9.clq' datasets and CPLEX.

We experimented with the pre-trained MIP-GNN models using a bias threshold of 0, which proved to be better than those with a bias threshold of 0.1 according to the empirical findings of Khalil et al..

We used CPLEX 22.1.0 to obtain the optimal solutions to the instances in training, validation, and test datasets. Each CPLEX run for the data collection, and benchmarking was carried out with a single thread; hence all runs were parallelized. All tasks and experiments were conducted on computers with an NVIDIA Quadro RTX 5000 GPU, an Intel Xeon Silver 4215R CPU, and 32 GB RAM.

Appendix C. Supporting Results

Table 7: Average improvement rates (Imp.) for the primal gap values of warm-start solutions generated in UPR algorithm by EC+V and EC models using `comb` over those using `mean` aggregation for the testing and the largest-scale transfer datasets of each problem. The negative percentages indicate that the `comb` models underperformed.

Problem	Loss Function	(Test) Warm-start Primal Gap Imp. (%)	(Transfer) Warm-start Primal Gap Imp. (%)
MSC	BCE	0.13	-259.24
	EDL	11.03	-13.56
CA	BCE	24.64	7.30
	EDL	-16.80	86.14
MIS	BCE	10.96	54.03
	EDL	22.46	18.58
FCMNF	BCE	85.67	54.55
	EDL	49.13	26.56
GIS	BCE	16.11	-60.07
	EDL	20.33	-0.72

Table 8: The comparison of our models that use 4-layered GNNs and `mean` aggregation, with MIP-GNN models. The negative average improvement rates (Imp.) indicate that Model 1 provides worse average primal gap. p-values are for the t-test results of the model pairs. The results of NS strategy for FCMNF-16x are excluded since the distribution pairs are almost the same (thus no statistical difference) as seen in Figure 10b.

Problem	Strategy	Model 1	Model 2	Avg. Primal Gap by Model 1	Avg. Primal Gap by Model 2	Imp. (%) by Model 1	p-value
FCMNF-16x	UPR+NS	EC+BCE	MIPGNN	0.1187	0.1226	3.1491	6.05e-01
		EC+EDL	MIPGNN	0.1161	0.1226	5.2605	3.49e-01
		EC+E+BCE	MIPGNN+E	0.1255	0.1244	-0.9318	8.66e-01
		EC+E+EDL	MIPGNN+E	0.1145	0.1244	7.9200	1.54e-01
		EC+V+BCE	MIPGNN+E	0.1333	0.1244	-7.1664	2.05e-01
		EC+V+EDL	MIPGNN+E	0.0942	0.1244	24.2659	2.35e-05
GIS-4x	NS	EC+BCE	MIPGNN	0.0847	0.0744	-13.8769	5.03e-02
		EC+EDL	MIPGNN	0.0752	0.0744	-1.0897	8.81e-01
		EC+E+BCE	MIPGNN+E	0.0581	0.0923	37.0493	1.48e-08
		EC+E+EDL	MIPGNN+E	0.0855	0.0923	7.3524	2.45e-01
		EC+V+BCE	MIPGNN+E	0.0989	0.0923	-7.1571	2.79e-01
		EC+V+EDL	MIPGNN+E	0.0489	0.0923	47.0047	1.85e-11
	UPR+NS	EC+BCE	MIPGNN	0.0618	0.0680	9.1961	2.70e-01
		EC+EDL	MIPGNN	0.0717	0.0680	-5.2946	5.59e-01
		EC+E+BCE	MIPGNN+E	0.0571	0.0949	39.8881	2.36e-07
		EC+E+EDL	MIPGNN+E	0.0715	0.0949	24.6874	2.23e-03
		EC+V+BCE	MIPGNN+E	0.0939	0.0949	1.0452	8.90e-01
		EC+V+EDL	MIPGNN+E	0.0495	0.0949	47.8931	1.34e-09

Appendix D. All Results

We conducted experiments on all CO pipeline configurations for all datasets except EC+E models, which were only experimented with on the testing and largest-scale transfer dataset of each problem. Consequently, the results of those pipelines are not available in the figures and tables for 2x, 4x, and 8x-transfer datasets of MSC, CA, and MIS problems.

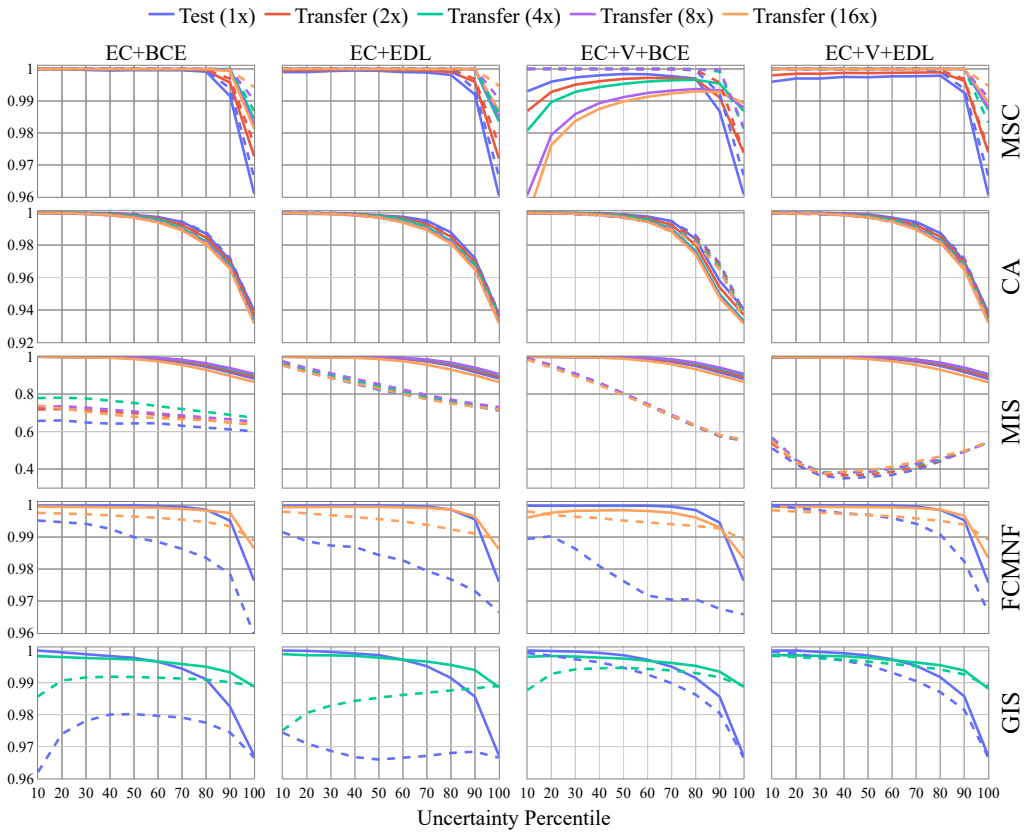
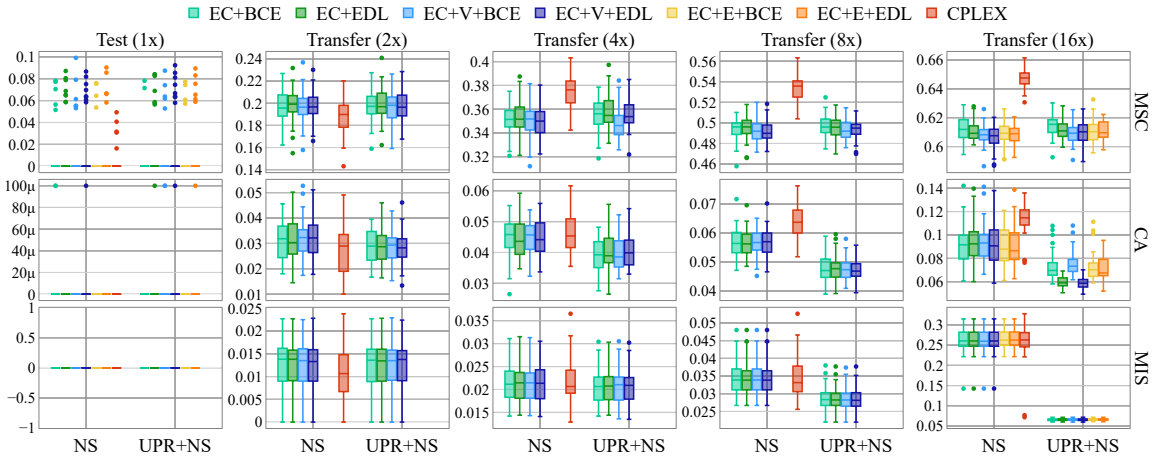


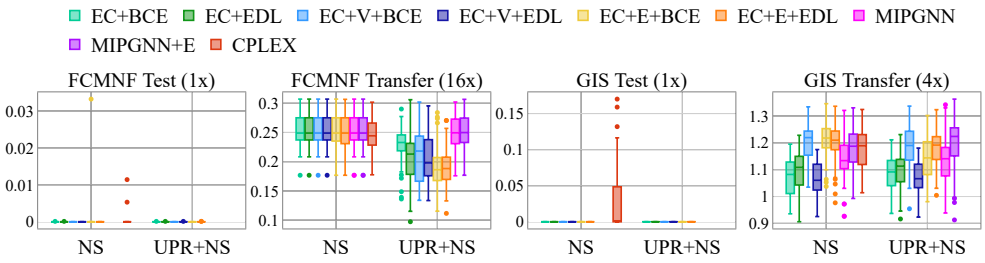
Figure 12: Average Accuracy of the Predicted Decision Values by the Models with (solid curves) and without (dashed curves) the Proposed Normalization Techniques in Each Uncertainty Percentile for All Datasets

Figure 12 presents the average accuracy of EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models in each column for their predictions in each ten percentile of the associated uncertainty values for all datasets in the rows. An exceptional finding is that the uncertainty values of the EC+V+BCE model for MSC transfer datasets can be misleading about the accuracy of highly confident predictions, causing the uncertainty percentile-accuracy curve not to start with the highest accuracy level. This also occurs slightly for the FCMNF transfer dataset. During the earlier phase of development, it was observed that models without any dropout layer in the classifier network make highly confident but misleading predictions, particularly for MSC datasets. This was attributed to overfitting to input features. For instance, these MSC models were over-biased towards objective coefficients and predicted 1's for almost all variables with small objective coefficients. To address this issue, all models were regularized by applying a 0.1 dropout probability in each layer of their classifier neural network. Although EC+V+BCE for the MSC problem would benefit from a higher dropout rate or further regularization, it was intentionally kept at a 0.1 dropout rate to demonstrate its negative impact on downstream optimization performance. As a result, the EC+V+BCE model produces inferior warm-start solutions with the UPR algorithm as the scale of MSC problem increases (see Figure 16a) and the CO pipelines using EC+V+BCE for FCMNF problem prominently underperform (See Figures 3-5).

Figure 13, 14, and 15 present boxplots of the optimality gap, the primal gap, and the primal integral results for all models with NS and UPR+NS strategies. Each boxplot represents the distribution of a corresponding metric for 50 instances.



(a) Optimality Gap Distributions for MSC, CA, and MIS Datasets

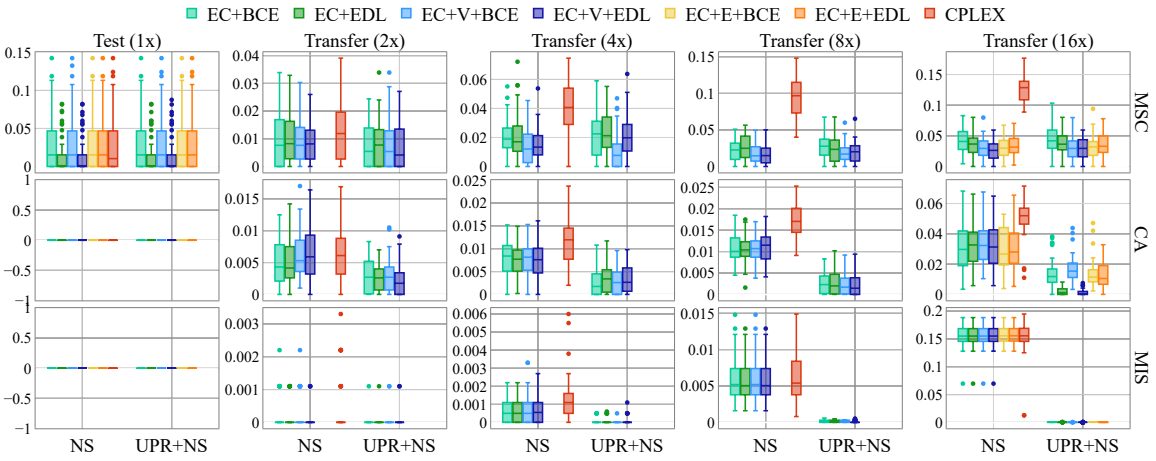


(b) Optimality Gap Distributions for FCMNF and GIS Datasets

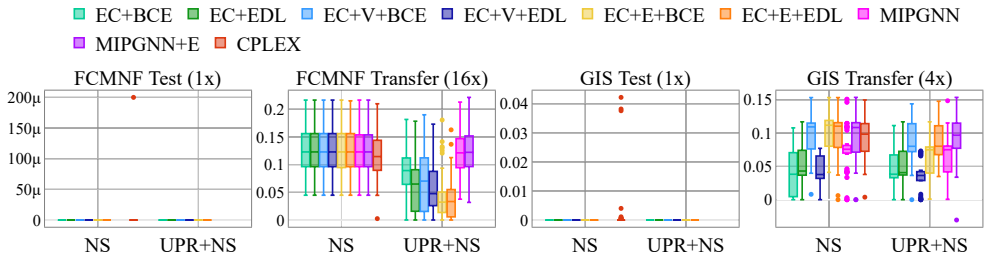
Figure 13: Optimality Gap Distributions for All Datasets

D.1 Time Efficiency of the Proposed Methodology

In this section, we first explain how we calculate the time efficiency gained by the CO pipelines using EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL over the default configuration of CPLEX. Afterwards, we present the time efficiency results. Assume that for a given MSC-2x instance, the NS pipelines deploying four models achieve their individual best objective values (lower better) 435, 430, 420, and 415 in 1, 2, 3, and 4 minutes, respectively. Assume CPLEX achieves those objective values in 2, 4, 6, and 16 minutes; then, the average time efficiency of four CO pipelines is $(2/1+4/2+6/3+16/4)/4 = 2.5$ for the given MSC instance. As an another case, assume that CPLEX cannot find a solution having the objective value of 415 in the given time limit (one hour), so it cannot be included in the calculation of the average time efficiency, which then $(2/1+4/2+6/3)/3 = 2$. The second case shows that although CPLEX fails to achieve the best objective value of EC+V+EDL pipeline, the average efficiency is less compared to the first case. Accordingly, the average success rate of CPLEX to reach the solution quality of the CO pipelines and the average time efficiency of the CO pipelines should be interpreted together.



(a) Primal Gap Distributions for MSC, CA, and MIS Datasets



(b) Primal Gap Distributions for FCMNF and GIS Datasets

Figure 14: Primal Gap Distributions for All Datasets

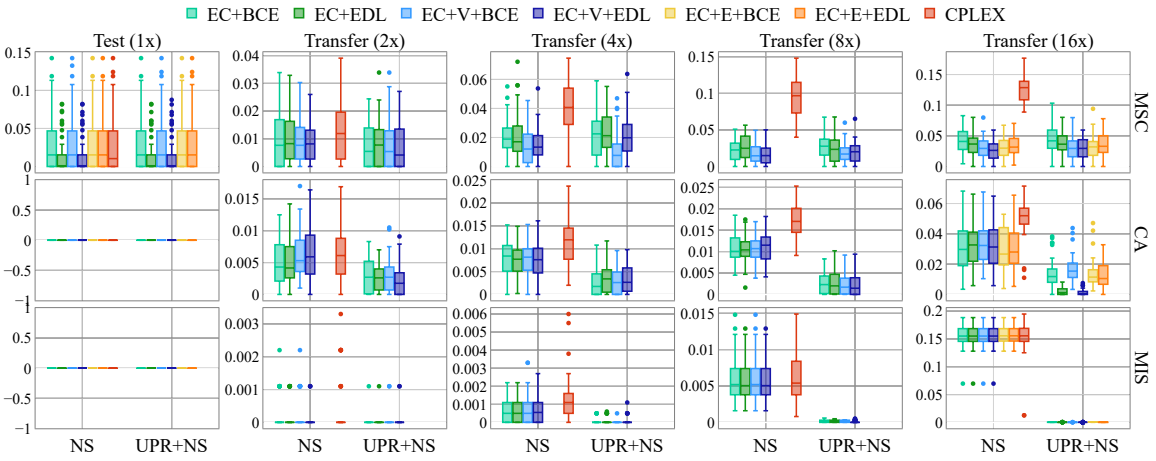
Table 9: (9a) The long-run CPLEX's success rate to achieve solution quality of the CO pipelines using EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models, and (9b) the average time efficiency of these pipelines relative to the default CPLEX performance for each scale of all problems.

(a) Average success rate (%) of CPLEX for long runs in finding same quality solutions with those obtained by the four CO pipelines.

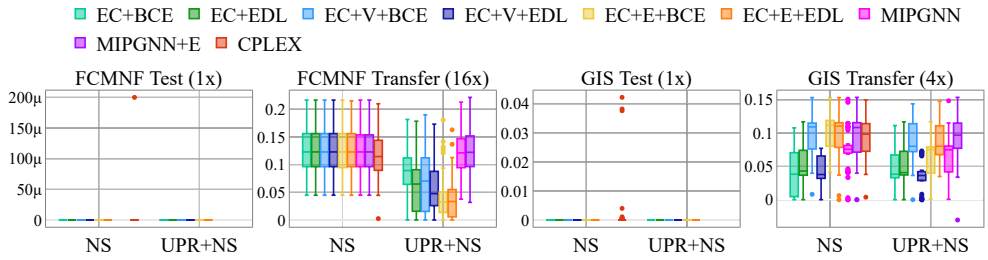
Problem	Strategy	1x	2x	4x	8x	16x
MSC	NS	72.5	63.5	52.0	9.5	7.5
	UPR+NS	72.5	52.5	52.0	9.5	7.5
CA	NS	88.0	53.5	63.0	66.5	99.0
	UPR+NS	88.0	30.5	22.5	5.0	24.5
MIS	NS	98.0	76.5	54.5	98.0	100.0
	UPR+NS	98.0	75.5	26.5	0.0	0.0
FCMNF	NS	87.0	-	-	-	84.0
	UPR+NS	77.0	-	-	-	49.5
GIS	NS	98.0	-	65.5	-	-
	UPR+NS	98.0	-	63.0	-	-

(b) Average time efficiency (better if higher than 1.0) of the four CO pipelines relative to the long-run CPLEX performance.

Problem	Strategy	1x	2x	4x	8x	16x
MSC	NS	0.82	1.35	3.69	12.41	19.97
	UPR+NS	0.91	1.52	3.16	10.47	18.82
CA	NS	0.62	1.01	2.54	4.74	1.77
	UPR+NS	1.44	1.34	3.40	8.02	11.94
MIS	NS	2.71	1.26	2.19	2.60	2.37
	UPR+NS	3.71	7.77	61.26	NA	NA
FCMNF	NS	0.84	-	-	-	3.71
	UPR+NS	1.96	-	-	-	7.33
GIS	NS	2.86	-	9.98	-	-
	UPR+NS	3.60	-	14.82	-	-



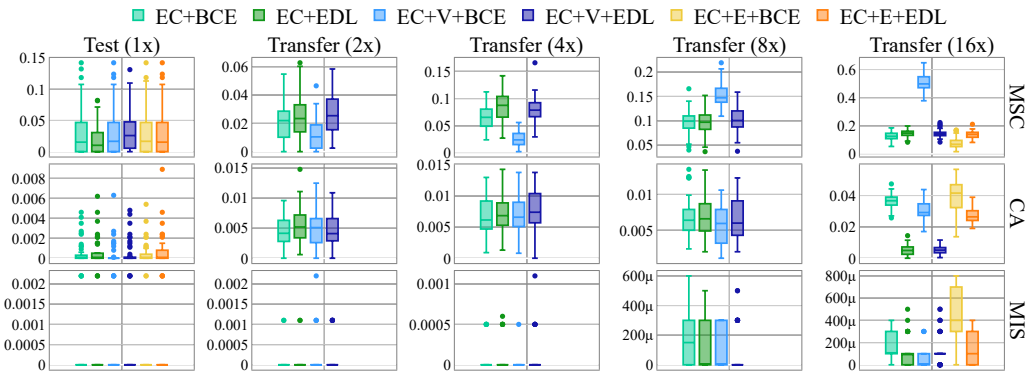
(a) Primal Integral Distributions for MSC, CA, and MIS Datasets



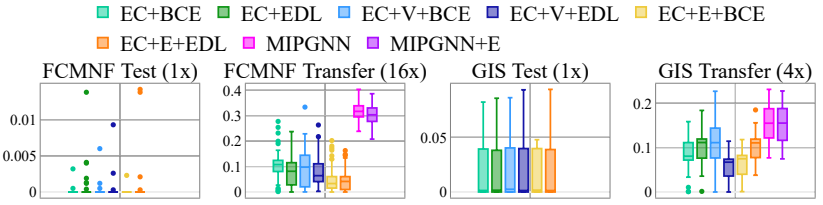
(b) Primal Integral Distributions for FCMNF and GIS Datasets

Figure 15: Primal Integral Distributions for All Datasets

The default configuration of CPLEX was also experimented with longer time limits (recall Section 5). For MSC, CA, and MIS problems, the time limits (per instance) are 1, 2, 4, and 8 hours for the transfer-2x, 4x, 8x, and 16x datasets, respectively. For GIS-2x and FCMFN-16x, the time limit is 8 hours per instance. Yet, CPLEX cannot always reach the solution quality of the CO pipelines, even its time limit is much longer. It is observed from Table 9a that the 30-minute solution quality of the CO pipelines for the most of instances cannot be obtained with CPLEX. Particularly, the success rates are zero for MIS-8x and MIS-16 datasets since a solution that is at least as good as that of the CO pipelines cannot be found with CPLEX for any instance of those datasets in respectively 4 and 8 hours. For the instances in which CPLEX can achieve solutions (with the longer time limits) that are at least good as that of the CO pipelines, Table 9b presents the average time efficiency through the CO pipelines relative to the CPLEX performance. The values in Table 9b are the proportion of the corresponding values in Table 10b and Table 10a. The values are averaged across the four CO pipelines and 50 instances of each dataset. Since FCMNF and GIS problems only have two datasets, the cells of the remaining problem scales are filled by '-' in these tables. 'NA' denotes that CPLEX cannot obtain reach solution quality of any CO pipelines for any problem instance in the given long runtime. For example, the NS pipelines using those four models obtain their individual best objective values for MSC-16x instances in 17.21 minutes on average and then do not find better solutions in the remaining average



(a) Warm-start Primal Gap Distributions for MSC, CA, and MIS Datasets



(b) Warm-start Primal Gap Distributions for FCMNF and GIS Datasets

Figure 16: Primal Gap Distributions of Warm-start Solutions through UPR Algorithm for All Datasets

Table 10: Average elapsed time (in minutes) results of the CO pipelines and CPLEX. The results are averaged over EC+BCE, EC+EDL, EDL+V+BCE, and EC+V+EDL models and 50 instances of each dataset.

(a) The average elapsed time of the CO pipelines to obtain individual best objective values.

Problem	Strategy	1x	2x	4x	8x	16x
MSC	NS	4.01	14.17	17.19	16.40	17.21
	UPR+NS	3.75	12.30	17.55	18.18	18.28
CA	NS	2.79	29.97	29.97	29.95	29.92
	UPR+NS	1.22	28.26	28.06	28.29	28.05
MIS	NS	0.17	3.40	8.78	25.61	20.72
	UPR+NS	0.04	0.61	0.34	0.09	0.47
FCMNF	NS	3.49	-	-	-	20.34
	UPR+NS	1.42	-	-	-	15.81
GIS	NS	6.63	-	11.88	-	-
	UPR+NS	5.27	-	10.10	-	-

(b) Average elapsed time of CPLEX to obtain the best objective values provided by the CO pipelines.

Problem	Strategy	1x	2x	4x	8x	16x
MSC	NS	3.27	19.17	63.42	203.50	352.32
	UPR+NS	3.41	18.73	55.48	190.39	351.77
CA	NS	1.75	30.22	76.22	142.06	52.97
	UPR+NS	1.75	37.82	95.46	226.85	334.99
MIS	NS	0.47	3.70	19.17	64.73	49.46
	UPR+NS	0.47	4.76	25.44	NA	NA
FCMNF	NS	2.95	-	-	-	75.50
	UPR+NS	2.79	-	-	-	115.94
GIS	NS	19.00	-	118.56	-	-
	UPR+NS	19.00	-	149.73	-	-

12.79 minutes. On the other hand, CPLEX spends average 352.32 minutes to reach the individual best objective values by the NS pipelines for MSC-16x instances. Accordingly, CPLEX finds solutions as good as those achieved by the NS pipelines only if it runs for 19.97x their elapsed time on average.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., & Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, *76*, 243–297.

Achterberg, T. (2007). Conflict analysis in mixed integer programming. *Discrete Optimization*, *4*(1), 4–20. Mixed Integer Programming.

Achterberg, T., Berthold, T., Koch, T., & Wolter, K. (2008). Constraint integer programming: A new approach to integrate CP and MIP. In Perron, L., & Trick, M. A. (Eds.), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 6–20, Berlin, Heidelberg. Springer.

Achterberg, T., & Wunderling, R. (2013). *Mixed Integer Programming: Analyzing 12 Years of Progress*, pp. 449–481. Springer, Berlin, Heidelberg.

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

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

Berthold, T. (2018). A computational study of primal heuristics inside an mi(nl)p solver. *J. of Global Optimization*, *70*(1), 189–206.

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., & Witzig, J. (2023). Enabling research through the SCIP optimization suite 8.0. *ACM Trans. Math. Softw.*, *49*(2).

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., & Friedrich, T. (2022). What's wrong with deep learning in tree search for combinatorial optimization. In *The Tenth International Conference on Learning Representations*. OpenReview.net.

Cai, T., Luo, S., Xu, K., He, D., Liu, T., & Wang, L. (2021). Graphnorm: A principled approach to accelerating graph neural network training. In Meila, M., & Zhang, T. (Eds.), *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139, pp. 1204–1215. PMLR.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., & Veličković, P. (2021). Combinatorial optimization and reasoning with graph neural networks. In Zhou, Z.-H. (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4348–4355. Survey Track.

Chen, Z., Liu, J., Wang, X., Lu, J., & Yin, W. (2023). On representing mixed-integer linear programs by graph neural networks. In *International Conference on Learning Representations*.

Choudhury, S., Solovey, K., Kochenderfer, M. J., & Pavone, M. (2021). Efficient large-scale multi-drone delivery using transit networks. *J. Artif. Int. Res.*, *70*, 757–788.

Colombi, M., Mansini, R., & Savelsbergh, M. (2017). The generalized independent set problem: Polyhedral analysis and solution approaches. *European Journal of Operational Research*, *260*(1), 41–55.

Conrad, J., Gomes, C. P., van Hoeve, W.-J., Sabharwal, A., & Suter, J. (2007). Connections in networks: Hardness of feasibility versus optimality. In Van Hentenryck, P., & Wolsey, L. (Eds.), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 16–28, Berlin, Heidelberg. Springer.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., & Velickovic, P. (2020). Principal neighbourhood aggregation for graph nets. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., & Lin, H. (Eds.), *Advances in Neural Information Processing Systems*.

Dai, E., Zhao, T., Zhu, H., Xu, J., Guo, Z., Liu, H., Tang, J., & Wang, S. (2022). A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *ArXiv*, *abs/2204.08570*.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6351–6361, Red Hook, NY, USA. Curran Associates Inc.

Derinkuyu, K., Tanrisever, F., Kurt, N., & Ceyhan, G. (2020). Optimizing day-ahead electricity market prices: Increasing the total surplus for energy exchange istanbul. *Manufacturing & Service Operations Management*, *22*(4), 700–716.

Ding, J., Zhang, C., Shen, L., Li, S., Wang, B., Xu, Y., & Song, L. (2020). Accelerating primal solution findings for mixed integer programs based on solution prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA*, pp. 1452–1459. AAAI Press.

Donti, P. L., Rolnick, D., & Kolter, J. Z. (2021). DC3: A learning method for optimization with hard constraints. In *9th International Conference on Learning Representations*. OpenReview.net.

Fioretto, F., Van Hentenryck, P., Mak, T. W. K., Tran, C., Baldo, F., & Lombardi, M. (2021). Lagrangian duality for constrained deep learning. In Dong, Y., Ifrim, G., Mladenić, D., Saunders, C., & Van Hoecke, S. (Eds.), *Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track*, pp. 118–135, Cham. Springer International Publishing.

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 1050–1059. JMLR.org.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., & Lodi, A. (2019). Exact combinatorial optimization with graph convolutional neural networks. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems, Vancouver, BC, Canada*, pp. 15554–15566.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 1263–1272. JMLR.org.

Gleixner, A., et al. (2021). MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, *13*(1).

Gupta, P., Gasse, M., Khalil, E. B., Kumar, M. P., Lodi, A., & Bengio, Y. (2020). Hybrid models for learning to branch. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Gurobi Optimization (2024). Irreducible inconsistent subsystem in gurobi. `https://www.gurobi.com/documentation/10.0/refman/py_model_computeiis.html`. [Online; accessed 01-May-2024].

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 1025–1035, Red Hook, NY, USA. Curran Associates Inc.

Han, Q., Yang, L., Chen, Q., Zhou, X., Zhang, D., Wang, A., Sun, R., & Luo, X. (2023). A GNN-guided predict-and-search framework for mixed-integer linear programming. In *International Conference on Machine Learning*.

Hewitt, M., Nemhauser, G. L., & Savelsbergh, M. W. P. (2010). Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, *22*(2), 314–325.

Huang, L., Chen, X., Huo, W., Wang, J., Zhang, F., Bai, B., & Shi, L. (2022). Improving primal heuristics for mixed integer programming problems based on problem reduction: A learning-based approach. In *17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 181–186.

IBM ILOG CPLEX (2021). Conflict refiner in CPLEX. `https://www.ibm.com/docs/en/icos/20.1.0?topic=conflicts-more-about-conflict-refiner`. [Online; accessed 01-May-2024].

Joshi, C. K., Cappart, Q., Rousseau, L.-M., & Laurent, T. (2021). Learning TSP Requires Rethinking Generalization. In Michel, L. D. (Ed.), *27th International Conference on Principles and Practice of Constraint Programming*, Vol. 210, pp. 33:1–33:21, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*(2), 393–422.

Khalil, E., Morris, C., & Lodi, A. (2022). MIP-GNN: A data-driven framework for guiding combinatorial solvers. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*, 10219–10227.

Khalil, E. B., Dilkina, B., Nemhauser, G. L., Ahmed, S., & Shao, Y. (2017). Learning to run heuristics in tree search. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, p. 659–666. AAAI Press.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, Toulon, France, Conference Track Proceedings*. OpenReview.net.

Korte, B., & Vygen, J. (2012). *Combinatorial Optimization: Theory and Algorithms* (5th edition). Springer Publishing Company, Incorporated.

Kotary, J., Fioretto, F., & Hentenryck, P. V. (2021a). Learning hard optimization problems: A data generation perspective. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., & Vaughan, J. W. (Eds.), *Advances in Neural Information Processing Systems*, pp. 24981–24992.

Kotary, J., Fioretto, F., Hentenryck, P. V., & Wilder, B. (2021b). End-to-end constrained optimization learning: A survey. In Zhou, Z. (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 4475–4482.

Labassi, A. G., Chételat, D., & Lodi, A. (2022). Learning to compare nodes in branch and bound with graph neural networks. In *NeurIPS*.

Lisicki, M., Afkanpour, A., & Taylor, G. W. (2020). Evaluating curriculum learning strategies in neural combinatorial optimization..

Liu, S., Zhang, Y., Tang, K., & Yao, X. (2022). How good is neural combinatorial optimization?. *CoRR*, *abs/2209.10913*.

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, p. 270–288, New York, NY, USA. Association for Computing Machinery.

Nair, V., et al. (2020). Solving mixed integer programs using neural networks. *CoRR*, *abs/2012.13349*.

Neal, R. M. (2012). *Bayesian learning for neural networks*, Vol. 118. Springer Science & Business Media.

Peng, Y., Choi, B., & Xu, J. (2021). Graph learning for combinatorial optimization: A survey of state-of-the-art. *Data Science and Engineering*, *6*, 119–141.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*, 61–80.

SCIP Optimization Suite (2024a). How to use conflict analysis. `https://www.scipopt.org/doc-8.0.3/html/CONF.php`. [Online; accessed 01-May-2024].

SCIP Optimization Suite (2024b). SCIP doxygen documentation: Primal heuristics. `https://scipopt.org/doc/html/group__PRIMALHEURISTICS.php`. [Online; accessed 01-May-2024].

Sensoy, M., Kaplan, L., & Kandemir, M. (2018). Evidential deep learning to quantify classification uncertainty. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 3183–3193, Red Hook, NY, USA. Curran Associates Inc.

Shen, Y., Sun, Y., Eberhard, A., & Li, X. (2021). Learning primal heuristics for mixed integer programs. In *International Joint Conference on Neural Networks*, pp. 1–8. IEEE.

Simonovsky, M., & Komodakis, N. (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29–38, Los Alamitos, CA, USA. IEEE Computer Society.

Smith, L. N., & Topin, N. (2019). Super-convergence: very fast training of neural networks using large learning rates. In Pham, T. (Ed.), *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Vol. 11006, p. 1100612. International Society for Optics and Photonics, SPIE.

Soleimany, A. P., Amini, A., Goldman, S., Rus, D., Bhatia, S. N., & Coley, C. W. (2021). Evidential deep learning for guided molecular property prediction and discovery. *ACS Central Science*, *7*(8), 1356–1367.

Wilder, B., Ewing, E., Dilkina, B., & Tambe, M. (2019). End to end learning and optimization on graphs. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems, Vancouver, BC, Canada*, pp. 4674–4685.

Yuan, H., Yu, H., Gui, S., & Ji, S. (2020). Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*, 5782–5799.

Zhang, J., Liu, C., Li, X., Zhen, H.-L., Yuan, M., Li, Y., & Yan, J. (2023). A survey for solving mixed integer programming via machine learning. *Neurocomputing*, *519*, 205–217.

Zhao, T., Pan, X., Chen, M., & Low, S. (2023). Ensuring DNN solution feasibility for optimization problems with linear constraints. In *The Eleventh International Conference on Learning Representations*.

Zhou, R., He, Z., Tian, Y., Wu, Y., & Du, S. S. (2023). Understanding curriculum learning in policy optimization for online combinatorial optimization. *Transactions on Machine Learning Research*, *N/A*.