GRAPH NEURAL NETWORKS-BASED PRIMAL HEURISTICS FOR COMBINATORIAL OPTIMIZATION

A Thesis

by

Furkan Cantürk

Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in
Department of Artificial Intelligence

Özyeğin University
May 2023

GRAPH NEURAL NETWORKS-BASED PRIMAL HEURISTICS FOR COMBINATORIAL OPTIMIZATION

Approved by:

Asst. Prof. Dr. Reyhan Aydoğan,
Advisor
Dept. of Computer Science
Özyeğin University

Asst. Prof. Dr. Emre Sefer
Dept. of Computer Science
Özyeğin University

Prof. Dr. Okan Örsan Özener,
Coadvisor
Dept. of Industrial Engineering
Özyeğin University

Assoc. Prof. Dr. Ali Haydar Özer
Dept. of Computer Science
Marmara University

Date Approved: May 17, 2023

Prof. Dr. Haluk Topçuoğlu
Dept. of Computer Science
Marmara University

To my lovely family...

ABSTRACT

By examining the patterns of solutions obtained for varying 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 end-to-end 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 the 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 up-scaled 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 a state-of-the-art MIP solver 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 in a computationally efficient modeling approach without sacrificing solution quality.

ÖZETÇE

Farklı örnekler için elde edilen çözümlerin yapısı incelenerek, kombinatoryal optimizasyon (KO) problemlerinin yapısı ve davranışı anlaşılabilir ve bunları çözmek için verimli algoritmalar geliştirilebilmektedir. Özellikle Grafik Sinir Ağları (GSA'lar) gibi Makine Öğrenmesi teknikleri, bu zahmetli tasarım sürecini parametrize etme ve otomatikleştirme için umut vadetmektedir. GSA'ların tümevarımsal yanlılığı, KO problemlerinin karışık tamsayılı programlama (KTP) formülasyonlarındaki karar değişkenleri ve kısıtların ilişkisel temsili üzerinden çözümleri öğrenmeyi mümkün kılmaktadır. Eğitilmiş GSA'lar, CO problemleri için yüksek kaliteli ve uyarlı çözümler oluşturmak için temel sezgisel yöntemlerle birlikte kullanılabilmektedir. Ancak, mevcut GSA tabanlı uçtan uca öğrenme yaklaşımları, ölçeklenebilir model eğitimi ve genelleştirme açısından çeşitli sınırlılıklar nedeniyle genellikle küçük ölçekli problem örnekleri üzerinden değerlendirilmiştir. Bu sorunu ele alan çalışmamız, büyük ölçekli KO problemlerinin küçültülmüş örneklerinin optimal çözümlerinin uçtan uca öğrenmesine dayanmaktadır. KO için kullanılan yeni bir GSA modeli için çeşitli geliştirmeler yaparak eğitimde kullanılanlardan daha büyük ölçekteki örneklerde modelin genelleme yapabilmesi sağlanmaktadır. Ayrıca, karar değeri tahminlerine dayalı olarak çözüm aramanın nasıl yapılandırılacağını otomatikleştirmek için model belirsizliğine dayalı iki aşamalı bir temel sezgisel yöntem önermekteyiz. Modellerimiz, yaygın olarak kıyaslanan beş KO probleminin 16 kat büyütülmüş örneklerinde genelleme yapabilmektedir. Problem büyüklüğü arttıkça mevcut GSA tabanlı KO yaklaşımlarının gitgide düşen performansının aksine, modellerimizi kullanan KO işlem hatları, son teknoloji bir KTP çözücüsü olan CPLEX'e göre gitgide artan bir performans iyileştirmesi sunmaktadır. Önerilen belirsizlik temelli temel sezgisel yöntemler,

yüksek kaliteli çözümleri elde etmek için büyük bir hız artışı sağlamakta ve 16 kat büyütülmüş örneklerde %6 ila %75 daha iyi optimalite aralığı ve %45 ila %99 daha iyi temel aralık değerleri sağlamaktadır. Tüm bu kazanımlar, çözüm kalitesinden ödün vermeden hesaplama açısından verimli bir modelleme yaklaşımıyla elde edilmektedir.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor Assistant Professor Reyhan Aydoğan for her continuous support throughout my master's program and encouraging attitude to pursue my research interests. Her guidance in this study and other projects in our laboratory kept me going with the right steps as an inexperienced master's student. I am also grateful to my co-advisor Professor Okan Örsan Özener for his guidance and invaluable advice in this study. I would not have been able to carry out this study without their wisdom.

I would like to thank the members of my thesis committee, Professor Haluk Topçuoğlu, Assistant Professor Emre Sefer, and Associate Professor Ali Haydar Özer, for their careful assessment of this study and suggestions. I would also like to thank Professor Mehmet Gönen and Dr. Milad Elyasi for their valuable comments on the writing of this study.

Special thanks to my friend Taha Varol for his support in our study. We had lots of meetings and made valuable critiques together that improved the quality of this work.

Finally, I am thankful for the support by The Scientific and Research Council of Türkiye (TÜBİTAK) through 2210 National Scholarship Programme for M.Sc. Students and the project grant 120N680 during my master's program.

TABLE OF CONTENTS

DEDICATION . . . . . **iii**

ABSTRACT . . . . . **iv**

ÖZETÇE . . . . . **vi**

ACKNOWLEDGEMENTS . . . . . **viii**

LIST OF TABLES . . . . . **xi**

LIST OF FIGURES . . . . . **xii**

LIST OF ABBREVIATIONS . . . . . **xiv**

I INTRODUCTION **1**

II BACKGROUND **9**

2.1 Combinatorial Optimization 9

2.2 Branch & Bound Search Algorithm 9

2.3 Graph Neural Networks 11

2.4 MIP Representation as a Bipartite Graph 13

III RELATED WORK **15**

3.1 End-to-End Learning for Primal Heuristics 18

3.2 Training Data Generation in End-to-End Learning for Primal Heuristics 22

IV PROPOSED METHODOLOGY **26**

4.1 GNN Architecture for Combinatorial Optimization 27

4.1.1 Feature Scaling and Normalization Layers 30

4.1.2 Violation Layer 32

4.1.3 Combined Aggregation 32

4.2 Uncertainty-based Primal Heuristics 33

4.2.1 Warm-start Solution Generation by Problem Reduction . . . 35

4.2.2 Uncertainty-Guided B&B Search 39

V EXPERIMENTAL SETTING **40**
5.1 Benchmarking Datasets 41
5.2 Implementation Details 44
5.3 Evaluation Metrics 45
VI EVALUATION **48**
6.1 Generalization on Larger-scale CO Problems 51
6.2 Benchmarking with CPLEX 54
6.3 Benchmarking with MIP-GNN 57
6.4 Effect of Warm-start Solutions 60
6.5 Effect of Violation Layer 61
6.6 Effect of Uncertainty Quantification 62
6.7 Effect of Combined Aggregation 64
VII DISCUSSION **67**
VIII CONCLUSION **69**
8.1 Future Work 69
8.2 Broad Impact 70
APPENDIX A — EXAMPLE INPUT & OUTPUT DATA **73**
APPENDIX B — BENCHMARKING DATASETS **75**
APPENDIX C — ALL RESULTS **78**
REFERENCES **84**
VITA **91**

LIST OF TABLES

1 An overview of the related studies and our study on end-to-end learning for primal heuristics. 23

2 (2a) 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 (2b) the average time efficiency of these pipelines relative to the default CPLEX performance. 58

3 The number of instances in which a feasible sub-MIP is not obtained in UPR algorithm for 50 FCMNF-16x instances. 60

4 The average improvement rates (Imp.) by the models trained with EDL Loss over those trained with BCE Loss for the testing dataset and the largest-scale transfer dataset of each problem. 63

5 The average improvement rates (Imp.) by the models using the combined aggregation (*comb*) over those using *mean* aggregation for the testing dataset of each problem and MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x transfer datasets. 66

6 The improvement rates (Imp.) by the models using the combined aggregation (*comb*) over those using *mean* aggregation for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x transfer datasets and the test dataset of each problem for the primal gap values of warm-start solutions generated in UPR algorithm. 66

7 The Output of EC+V+EDL model (trained on CA instances with 10 variables) for the given CA instance representation in Figure 13. . . . 74

8 The median and standard deviation of the number of variables, the number of constraints, and CPLEX solve time (with a single thread) for the training dataset of each problem. 77

9 The average and standard deviation of the inference time, which is the sum of elapsed time in the data preprocessing and prediction, for each dataset of each problem. 77

10 The average elapsed time (in minutes) of the CO pipelines that use EC+BCE, EC+EDL, EDL+V+BCE, and EC+V+EDL models to obtain individual best objective values for each scale of all problems. . . 81

11 The average elapsed time (in minutes) of CPLEX to obtain the best objective values achieved by the CO pipelines that use EC+BCE, EC+EDL, EDL+V+BCE, and EC+V+EDL models for each scale of all problems. 83

LIST OF FIGURES

1 An Example B&B Tree Representation. The numbers at the top-left corner of the boxes denote the order of node exploration. LB denotes the current lower bound when a B&B node is explored (after the first integral solution). 12

2 MIP Representation as a Bipartite Graph. Image credit to [1]. 14

3 MIP-GNN Architecture with two-layer Graph Embedding 28

4 The Boxplot Distributions of Optimality Gap Values for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets 50

5 The Boxplot Distributions of Primal Gap Values for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets 50

6 The Boxplot Distributions of Primal Integral Values for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets 50

7 The Boxplot Distributions of Primal Gap Values by the Warm-start Solutions Found with UPR for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets 51

8 The Average Primal Gap Curves for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets (NS and UPR+NS results are denoted by solid and dashed lines, respectively.) 51

9 The 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 53

10 The Average Improvement Rates 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.) 55

11 The Average Improvement Rates by the Model Configurations for the FCMNF-16x and GIS-4x Datasets 56

12 The Boxplot Distributions of (12a) Final and (12b) Warm-starting Primal Gap Values by the CO Pipelines Using the Models with 4-layered GNN and *mean* aggregation for FCMNF-16x and GIS-4x Transfer Datasets 59

13 The Bipartite Graph Representation for CA Instance 21 74

14 The Boxplot Distributions of Optimality Gap Values for MSC, CA, MIS, FCMNF, and GIS Datasets 79

15 The Boxplot Distributions of Primal Gap Values for MSC, CA, MIS, FCMNF, and GIS Datasets 80

16 The Boxplot Distributions of Primal Integral Values for MSC, CA, MIS, FCMNF, and GIS Datasets 81

17 The Boxplot Distributions of Primal Gap Values of Warm-start Solutions through UPR algorithm for MSC, CA, MIS, FCMNF, and GIS Datasets 82

18 The Boxplot Distributions Optimality Gap and Primal Integral Values by the Models with 4-layered GNN and *mean* aggregation, and MIP-GNN models for FCMNF-16x and GIS-4x Transfer Datasets 82

LIST OF ABBREVIATIONS

B&B Branch & Bound.

BCE Binary Cross Entropy.

CA Combinatorial Auction.

CO combinatorial optimization.

EC Edge Convolution.

EDL Evidential Deep Learning.

FCMNF Fixed-Charge Multi-Commodity Network Flow.

GIS Generalized Independent Set.

GNN Graph Neural Network.

LP linear programming.

MILP mixed-integer linear programming.

MIP mixed-integer programming.

MIS Maximal Independent Set.

ML machine learning.

MLP multi-layer perceptron.

MPNN Message-Passing Neural Network.

MSC Minimum Set Cover.

UPR Uncertainty-based Problem Reduction.

CHAPTER I

INTRODUCTION

Combinatorial optimization (CO) problems arise in myriad fields and are generally hard to scale due to their exponential complexity [2]. Unfortunately, real-world applications are complex systems of many interacting entities with intrinsic relationships; therefore demand large-scale optimization. Even though CO algorithms have matured, they might not scale up due to the prohibitive combinatorial structure (unless decomposable) and the requirement of massive computation power. Therefore, efficient solution methods for CO problems are persistently needed. On the other hand, many industrial CO problems are recurrently solved, such as fleet routing [3], daily production planning, data center management [4], optimal power flow [5], and energy market auction [6]. Besides, varying scenarios of these problems can occur on the same problem structure and have pronounced characteristics, or their parameter distribution can be extracted from reoccurrences. 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 the accumulation of data on problem characteristics, solving CO problems repetitively produces vast empirical data of solution traces by algorithms. All these presented aspects of CO problems create an opportunity 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 have recently started to develop ML-augmented CO solvers to construct heuristic solutions end-to-end and empower existing generalized approaches like mixed-integer programming (MIP) solvers [7, 8, 9].

CO problems can be traditionally solved by Branch & Bound (B&B) Search, which is the core algorithm of MIP solvers, by formulating them as *mixed-integer programs* (MIPs). B&B algorithm solves MIPs exactly by performing *primal task* of finding feasible and good solutions and *dual task* of certifying the optimality. However, it becomes intractable as problems get larger. MIP solvers have been developed to perform these two challenging tasks of B&B by addressing some inherent decision-making problems like *variable selection, node selection*, and *cut selection* and running advanced subroutines like *primal heuristics* and *tightening cut generation* with a variety of methods developed in years of laborious designs and computational studies [10, 11]. Particularly, primal heuristic methods, which try to find high-quality solutions quickly, are appealing in especially real-time decision-making domains where high-quality solutions are demanded in a short time and periodically rather than prioritizing the proof of solution optimality. Thus, neural networks are promising to scale primal heuristics for CO as they can approximate problem-solution mappings and parametrize algorithms to tailor them for specific problem data [8, 12]. Besides, they could be used for constructing solutions efficiently with matrix-based operations on GPUs, whereas traditional CO algorithms only run on CPUs [13].

ML approaches based on Graph Neural Networks (GNNs) contribute to MIP, CO, constraint satisfaction, and algorithmic reasoning fields extensively [9] since GNNs have useful capabilities, such as permutation invariance and equivariance, allowing varying input size and inductive bias, for learning graph-structured information. As graphs are a generic way of knowledge representation, GNNs have begun to be widely used [14]. Considering this functionality of GNNs, our study aims to scale end-to-end learning-based primal heuristics for CO. The backbone of end-to-end learning for CO is to induce solutions with GNNs to given problem information/representation. A generic way of relational representation of CO problems is based on MIP formulations of CO problems as the bipartite graphs of the decision variables and

constraints [12, 15]. Besides, graph-structured problems like Max-Cut and Minimum Set Cover can be directly modeled by encoding their graph representations [16]. For given such graph representations, one end-to-end learning approach to CO deploys supervised models with solutions to training instances obtained beforehand. Alternatively, autoregressive models could learn a sequential inductive bias, which improves generalization over supervised, non-autoregressive models but are costly in terms of inference time [17, 18]. Supervised models are more amenable to post-hoc search as their output as a whole is likely to be an infeasible solution, while reinforcement learning approaches scale better but require more computation as they perform beam search or sampling [17]. Although Graph Neural Network (GNN)-augmented CO solvers enhance a wide range of applications, scaling them is challenging when the size of targeted problems is large [8, 17].

There exist several challenges related to the nature of CO problems for utilizing ML techniques. Firstly, end-to-end learning approaches to CO struggle to generalize on larger-scale problems, although GNNs provide the flexibility of inference for different-sized problems [15], possibly due to the relations among decision variables getting complicated as the problem size increases. Secondly, end-to-end learning approaches based on supervised models approximate high-quality solutions to CO problems at hand. However, high-quality labeled training data are often expensive to obtain, which limits the practicality of supervised learning models for NP-hard problems [17]. The recent studies [19, 20] report that the previous studies adopting GNNs for branching policy and primal heuristics [15, 21] experiment small-scale CO problems, which SOTA solvers like CPLEX and Gurobi can solve instantly. Besides, in their recent study on supervised learning-based primal heuristics, Khalil *et al.* [19] remark that their proposed framework MIP-GNN is convenient for CO problems in which past instances are available with their near-optimal solutions [19].

Another design issue is how the trained models with supervision are exploited for

the downstream optimization task. Utilizing the predictions of supervised models typically requires ad-hoc processes to acquire feasible solutions for constrained optimization problems and post-hoc strategies for solution search in discrete domains. The common strategies exploiting those trained models are problem reduction by fixing variables to the predictions, guiding B&B search, and local branching around the predicted solution [1, 19, 20, 21, 22]. These strategies typically include post-processes requiring tuning by solving instances of a given problem, thus leading to development overhead increasing with the number of hyperparameters.

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

1. A bottleneck for the scalability of ML models for CO is the training data generation procedure. A commonly adopted approach depends on the collection of high-quality solutions by solving a set of instances of the target problem with a MIP solver. As the required time to obtain high-quality solutions for large instances increases w.r.t. exponential space complexity of CO problems, the scale of solved instances to generate a training dataset is limited.

2. Most training data generation approaches use multiple solutions per problem instance to set a target space to be learned, which might limit models to capture patterns of feasibility and optimality of solutions.

3. Training GNN models require a massive amount of GPU memory depending on the scale of training instances. This causes a handicap for the computational achievability of supervised GNN models for large-scale CO problems.

4. Several studies [15, 21, 22, 23] examine the possibility of generalization to solve larger instances of given CO problems than the ones used in training. Yet, the evaluations of these studies do not draw an explicit conclusion about the

generalization on larger-scale instances. It is still open to producing evidence of scaling end-to-end learning approaches to CO.

5. The primal heuristic strategies proposed in the current studies rely on class probabilities or coverage rates to get up a partial solution. Class probabilities might be misleading for unusual samples (e.g., larger-scale) compared to ones in the training dataset, causing 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-to 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 [19], to infer solutions successfully for larger-scale instances of CO problems and builds on the following methods, each addressing the corresponding challenges (listed above) denoted in parentheses.

- Rather than generating a solution pool per training instance, 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. So, we can obtain models capturing solution patterns that might be useful even for much larger instances with a much lower training cost (Challenges 1, 2, and 3).

- For generalization on larger-scale instances than the training instances, we endow the MIP-GNN architecture with several normalization techniques applied to raw features and hidden embeddings of the nodes in the bipartite graph.

Thus, we can maintain predictive performance steady in inference for larger-scale instances (Challenge 4).

- We train the models with an *uncertainty-quantification* approach, Evidential Deep Learning [24], to calibrate their prediction uncertainty and form our primal heuristics based on utilization of predictions w.r.t. associated uncertainty values. In this way, our models are tuned during the training regarding the performance in the downstream optimization task, and the proposed primal heuristic procedures are automatically adapted to changing solution patterns (if existing) based on the problem scale (Challenges 4 and 5).

- Our primal heuristics strategy utilizing the predicted values for decision variables of CO problems builds on two methods which are *Warm-start solution generation*, and *Uncertainty-based guided B&B node selection*. These methods fully exploit the trained models by considering the predicted values and associated uncertainty values for all decision variables in the problem to be solved (Challenges 4 and 5).

For five different CO problems commonly benchmarked in the related ML for CO literature, we conduct an extensive experiment that covers benchmarking with a state-of-the-art solver CPLEX and the pre-trained MIP-GNN models. Then, we present the ablation study of the proposed architectural improvements. Our models trained over small-scale (1x) instances are evaluated for large-scale (2x, 4x, 8x, and 16x) transfer instances of those problems. The trained GNNs can generalize on the transfer instances as accurately as the testing performance. The GNN-augmented CPLEX pipelines deploying the proposed primal heuristics strategy (shortly, 'the CO pipelines' from here on) exhibit an increasing trend of performance gain relative to default CPLEX performance as the scale of problem instances increases. This result is unlike the deteriorating performance by GNN-based CO approaches [15, 17, 21, 25]. It is

noteworthy that our experiments aimed to efficiently solve the 16x-transfer instances that were well above the scale of those attempted in those studies. In addition to targeting to solve large-scale problems, our study exposes the challenge of real-world applications by benchmarking with an industry-level solver software CPLEX, which is harder to be outperformed [19] compared to the commonly benchmarked academic-purpose solver SCIP. Compared to the performance of CPLEX, the CO pipelines approximately achieve 80-99% better primal gap values for 16x-transfer instances of three fundamental CO problems and 45% and 60% better primal gap values for the other two problems, which are presented in [19] as more challenging datasets.

As significant evidence of offering a generalized primal heuristics approach, our CO pipelines can boost the efficacy of B&B algorithm for all benchmarked CO problems, each showing different solution progress, and yield high orders of time efficiency to obtain high-quality solutions. Secondly, our models present close or better optimization performance in solving the actual-scale instances of two CO problems, which MIP-GNN models are trained on, compared to the testing performance of MIP-GNN models. It corroborates the benefit of learning patterns with the proposed architecture over the optimal solutions to the downscaled instances.Overall, our experimentation substantiates that the proposed methodology can enable end-to-end learning to solve each problem at scale. Last but not least, our methodology is also cost-efficient in terms of required computation for the training dataset generation and GNN model training, which are the processes causing intensive CPU and GPU workload in the development of the end-to-end learning methods for CO. The source code of our study will be accessible[1] once it is published.

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 end-to-end learning for primal heuristics comprehensively. Section

[1] https://github.com/furkancanturk/scalable_gnn_4_co

4 proposes our end-to-end learning approach to solve CO at scale with GNN-based primal heuristics. Sections 5 and 6 present the experimental evaluation of our study. Sections 7 and 8 discuss in detail how the proposed methodology can scale for CO problems and envision further research directions. Lastly, we finalize our paper with a discussion of how it can affect applications in NP-hard domains.

CHAPTER II

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 among a finite set of possible solutions for a given optimization problem. It involves selecting a combination of elements from a given set that satisfies certain constraints and minimizes or maximizes a specific objective function [2]. In general, CO problems are formulated as mixed-integer linear programming (MILP) or Integer Linear Programming (ILP), which is a mathematical modeling of an optimization problem to minimize (or maximize) a linear objective function of continuous and/or integer decision variables 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 grows by dividing solution space with

constraining non-integer solutions. In each B&B node, a linear problem is obtained by relaxing (removing) integrality constraints and solved with linear programming (LP), which typically results much faster than MIP. If 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* methods applied in MIP solvers try to achieve feasible solutions quickly to improve *primal bound* while *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. This policy can manage the search in a depth-first, breadth-first, or best-first manner. These 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 [26]. In the following, we describe how B&B search proceeds for the given example MILP 2, which includes

only binary decision values for simplicity.

$$\text{max.} \quad z = 16x_1 + 12x_2 + 8x_3 \tag{2a}$$

$$\text{s.t.} \quad 5x_1 + 4x_2 + 3x_3 \leq 7 \tag{2b}$$

$$x_j \in \{0, 1\}, \quad \forall j \in \{1, 2, 3\} \tag{2c}$$

We provide an example B&B tree representation in 1 for solving MILP 2. Firstly, relaxing the integrality constraints (2c) yields an LP-relaxed solution having the objective value of 22 (being the dual bound (DB)) for the root node (0), which includes fractional value for only x_2. Therefore, two new B&B nodes are generated by adding cuts $x_2 = 0$ and $x_2 = 1$ to the root node. After solving these two subproblems with LP, LP-relaxed solutions with objective values of 21.3 and 21.6, respectively, are obtained. Accordingly, DB is updated as $DB \leftarrow \max\{\lfloor 21.3 \rfloor, \lfloor 21.6 \rfloor\}$. Assume an arbitrary node selection policy is applied, and it expands the tree over Node 1, in which LP-relaxation includes a fractional decision value, $x_1 = 0.6$. One of the new children nodes of Node 1 includes the first integral solution while the other one (3) is infeasible since its cuts $x_2 = 1$ and $x_1 = 1$ violate Constraint 2b. Hence, Node 3 is pruned from the tree, and the lower bound is set to the objective value of Node 2, 20. Then, the tree is expanded by branching Node 4 with the cuts $x_3 = 0$ and $x_3 = 1$. Node 5 includes a new integral solution with $z = 16$, but it is lower than $LB = 20$; therefore, LB is not updated. The LP-relaxed solution value of Node 6 is 20.8. Since the current LB is still 20, Node 6 is not promising for providing a solution with $z > 20$, thus being pruned from the tree. Consequently, an open B&B node does not remain, and the optimal solution to MILP 2 is the solution of Node 2 with $z = 20$.

2.3 Graph Neural Networks

GNNs are a generic way of modeling knowledge through relations within data [27, 28]. The fundamental concept underlying GNNs involves the computation of a vectorial

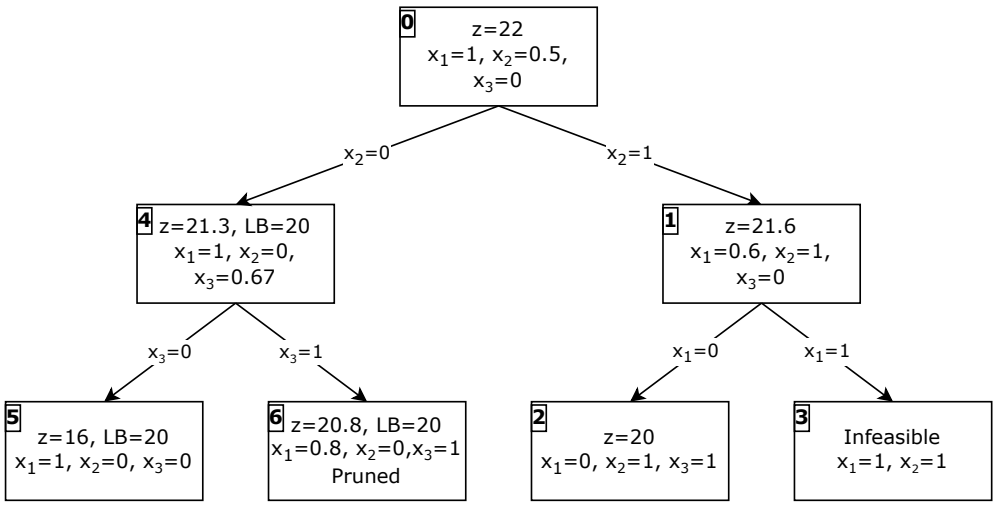


Figure 1: An Example B&B Tree Representation. The numbers at the top-left corner of the boxes denote the order of node exploration. LB denotes the current lower bound when a B&B node is explored (after the first integral solution).

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 is that the learned vector representation 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) [29], 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.

$$\mathbf{h}_v^{(k)} = \gamma^{(k)}\Big(\mathbf{h}_v^{(k-1)}, \bigoplus_{u \in N(v)} \big(\phi^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)})\big)\Big) \tag{3}$$

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 respective nonlinear function, e.g., Rectifier Linear Unit (ReLU), Perceptron, multi-layer perceptron (MLP), and attention networks. $\bigoplus$ can be a permutation-invariant operator such as *sum*, *mean*, *max*, *min*, and *standard deviation (std)*. 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 are dependent on each other through the constraints and the objective function of the problem. By representing decision variables and constraints as nodes, MIPs can be represented as undirected graphs and the relations among the nodes can be encoded with GNNs. Then solutions can be induced with GNNs learning how decision variables in the problem take values to form a solution that meets a set of constraints and is optimal (or high-quality) concerning an objective function. Gasse et al. [15] introduce a weighted bipartite graph to represent MIPs and use GNNs to model interrelations of the decision variables within a set of (in)equalities and to induce their values regarding the 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 2. A multiset of edges (E) connects the constraint nodes with corresponding variable nodes. Variable nodes are featured with and their domain types (binary, continuous, or integer). If decision variables are integral domain, $\mathbf{l}$ and $\mathbf{u}$ are also used in the feature set of variable nodes. Constraint nodes are featured with and their types ($\leq$ and $=$). Types of variables and constraints are one-hot encoded in the nodes. Lastly, edges are featured with $a_{i,j} \in \mathbb{A}$. A bipartite graph with the mentioned features fully embodies information in MIP problems, but more features can be extracted from the root and intermediate nodes of B&B tree as long as B&B algorithm processes [15, 21, 13]. 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 [12] on the investigation of the expressive power of GNNs to approximate solutions to given bipartite graph representations of MILPs.

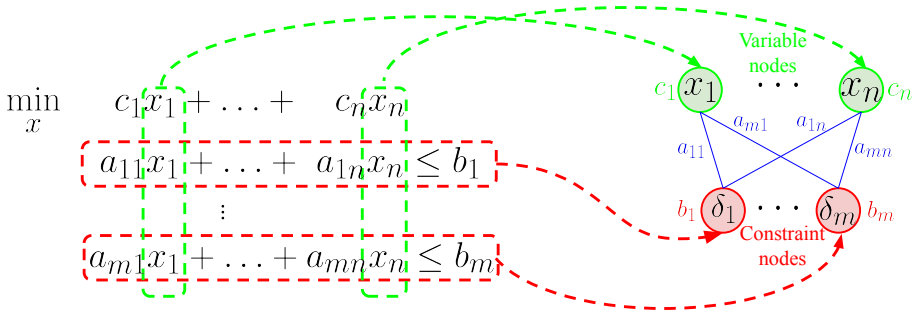


Figure 2: MIP Representation as a Bipartite Graph. Image credit to [1].

CHAPTER III

RELATED WORK

CO problems are addressed by different optimization approaches such as exact, approximation, and heuristic algorithms [2]. Exact methods like Branch and Bound (B&B) and Dynamic Programming (DP) iteratively partition a finite decision space and eventually find a provably optimal solution. However, these methods are intractable for domains with non-polynomial space complexity. Alternatively, approximation methods can find an ϵ-optimal solution assuring the quality of the solution is not worse than the optimal solution by ϵ rate. They are mostly used in domains where a tolerance level for solution quality is admissible or needed. If high-quality solutions are needed with a concern of solution time, one can resort to heuristic methods by exploiting problem structure and expert knowledge to obtain (desirably) good solutions quickly. Heuristics typically cannot guarantee a certain worst-case performance, but their efficiency can be validated empirically.

CO problems can be generally handled with fully-fledged MIP solvers like CPLEX, Gurobi, and SCIP. Existing methods accompanying B&B algorithm in MIP solvers have been advanced to make node selection and variable selection procedures in B&B efficient and incorporate advanced features like *cutting plane generation* and *heuristic method selection* [30]. Developing general MIP solvers and problem-specific solution designs requires significant time and expert knowledge [10, 11]. On the other hand, ML paradigms applied in optimization domains can bring automated heuristics and learning-driven decision-making methods into MIP solvers such as learning variable selection to branch [31, 15, 13, 1, 32], learning node selection [33, 34, 35], learning to cut generation or selection [36, 37], and learning neigbourhood search [38, 39, 40].

Data-driven optimization approaches utilize ML models that can expertise on a specific problem by exploiting data of past instances. In general, ML-augmented MIP solvers exploit data of solutions or trajectories of search/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. Other than learning search policies, ML architectures, such as GNNs, can automatically learn to identify and leverage the problem features and solutions patterns from raw data (without requiring handcrafted pre-processing or feature engineering) to generate solutions end-to-end.

On the side of end-to-end learning for CO, mainly two types of methodologies adopting ML emerge, *learning constructive solutions on graphs*, and *end-to-end learning for primal heuristics* [9, 41]. Learning constructive solution policies for CO problems encode graph-structured CO problems, which are typically unconstrained or loosely constrained like TSP and Minimum Vertex Cover, and train GNNs for a constructive solution policy using reinforcement learning or tree search [16, 18, 17, 42]. The latter methodology trains GNNs with supervision that predicts 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 [30].

GNNs can map different-sized graphs to their target vectors or scaler values, which paves the development of learning methodologies to scale CO problems by training GNNs with small-scale instances. Therefore, these models are intensively exploited in real-time decision-making domains like Travelling Salesman Problem (TSP), Vehicle Routing Problem [43], and energy areas [44]. In [45], a GNN model is trained on grid topologies of fixed size to predict power flow and achieve better accuracy than a standard benchmarking method on larger real power grids, but with increasing errors as the grid scale increases. Ma *et al.* [46] propose a GNN model trained on small-scale

instances of the traveling salesman with time windows problem using reinforcement learning. It can generalize to larger-scale instances better than previous RL-based TSP solvers. As distinct from RL-based approaches, [47] trains a GNN model with small-scale instances in a supervised manner, which is repetitively used to build heat maps for larger-scale TSP instances. Then, a Monte Carlo tree search over the heat maps guides the search for high-quality solutions. Although a considerable research effort on ML techniques for TSP emerges, the computational studies [17, 25] enounce that evaluation of those techniques commonly tends to overshadow the limited ability of models to generalize, as their efficacy is solely measured on predetermined or small-scale instances. Thus, these studies report the deteriorating performance of ML approaches to generalize on larger TSP instances.

Other than domain-specific ML-based approaches, GNN-embedded MIP solvers address a wider spectrum of CO problems. The prior works leveraging GNNs within MIP solvers are done in [15] for dual branching and in [21] for primal heuristics. Gasse *et al.* [15] report that the performance improvement of learned variable selection policy in solving CO problems decreases as the size of problems increases. The authors can mitigate this issue by training the proposed GNN model with larger instances. Similarly, it is observed in [21] that the generalization ability of the proposed model is not explicit while solving larger-scale instances of eight benchmarking CO problems. In the benchmarking with the methodology in [21] (called ML-Split), the recent studies [23, 22] train GNNs on problems with instances of graph problems with 500-1000 decision variables and evaluate them on solving the problems with 1500-3000 decision variables. These models can lead to finding high-quality solutions faster compared to the default configuration of SCIP and ML-Split. However, the solution quality improvement rates by the models in these three studies [21, 23, 22] are less than 1% on average compared to the best objective values obtained with SCIP. Another related

study [35] proposes a Siamese GNN model to imitate an expert oracle that prioritizes B&B nodes for exploration. The oracle accesses the optimal solution obtained before and compares each pair of open B&B nodes accordingly. The trained GNN model benchmarked with SCIP to rank B&B nodes for solving GIS, FCMNF, and MAXSAT problems. The study provides a limited assessment for generalization on larger instances which include only 15-50% more graph nodes than those trained on.

Even though GNNs make data-driven CO approaches flexible in solving different-sized problems, they provide either a limited improvement or declining optimization performance on larger-scale instances due to a lack of predictive generalization and/or unscalable training regimes. Considering the gap between the scale of industrial applications, the scalability of the proposed GNNs in the literature for CO is still questionable. Thus, our study positions to scale CO solvers by exploiting GNNs within primal heuristics. The following subsections detail recent studies on end-to-end learning for primal heuristics and discuss the training data generation approaches mostly adopted in these studies from a supervised learning perspective for optimization tasks.

3.1 End-to-End Learning for 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 [48, 49]. Primal heuristics methods construct a solution by virtue of heuristic information about solution pieces or problem structure. In this way, these methods aim 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 [10, 49] on MIP solvers like CPLEX, a state-of-the-art MIP solver [50], and SCIP, an open-source and academic-purpose MIP solver [11]. 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 [49]. Primal heuristics have been developed over decades by experts [10]. For example, 65 primal heuristic methods have been implemented in SCIP[1]. Moreover, ascertaining effective ones among several heuristics for a given problem requires domain knowledge and empirical validation. Alternatively, metaheuristics (see a recent survey [51]) and learning-driven approaches [48] can accompany B&B algorithm to run heuristics in more sophisticated and automated ways.

This section presents related studies on GNN-based supervised learning approaches to solve constrained CO problems with primal heuristics methods in a general-purpose (not problem-specific) way. All presented studies except [1] propose GNN models predicting values of decision variables in CO problems which can be formulated as (mixed) binary linear programs. The predictions of trained GNN models are harnessed as heuristic information to solve problems with MIP solvers. A standard workflow of these solvers consists of (i) training data generation, (ii) training a GNN model end-to-end w.r.t. a loss function based on target solutions, (iii) tuning the strategy of using model predictions for the downstream optimization task, and (iv) running a MIP solver[2] with primal heuristics utilizing the model predictions. We describe the previous studies on end-to-end learning for primal heuristics to solve CO problems considering these four phases.

In a pioneer study [21], *stable* decision variables, whose values remain the same in all of the local solutions around an initial solution, are labeled and included in the training dataset. A GNN model is trained to classify values of *stable* binary decision variables for the proposed tripartite graph representations of MIP formulations. Then, $\eta\%$ of the most confident predictions are used to generate a partial solution, and a local branching cut is generated to search around this solution up to a predefined

[1] https://scipopt.org/doc/html/group__PRIMALHEURISTICS.php

[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.

Hamming distance. Alternatively, a global cut is added to the root node of B&B tree, in which one child node covers the local cut, and the other child node covers the rest decision space. After all, CO problems are solved with the generated cut.

For the proposed approach in [21], finding an initial feasible solution can be challenging for an arbitrary MIP problem. The quality of the initial solution should be assured so that it can lead to good solutions during the local search. On the other hand, the stable variables represent local information about the structure of found solutions, so this approach might prevent learning the global structure of MIP decision space. Instead, Nair *et al.* [1] follow an energy-based generative modeling approach and weighten the effect of collected solutions in the proposed loss function (with a MIP solver per MIP instance) based on their objective values. For given bipartite graph representations of MIPs, a GNN is trained to predict the probability of being 1 for each binary decision variable in the collected solutions and is used to generate multiple solutions by sampling. Then, a subset of variables from each sampled solution is selected by a secondary binary classifier that is trained separately based on a predefined coverage rate. Finally, different sub-problems are obtained by fixing the selected variables to their values in sampled solutions and are solved by SCIP in parallel or sequentially. The feasible solution having the best objective value among all solved sub-problems is considered the final solution of the pipeline. Rather than fixing variables in the partial solutions generated in Neural Diving, Han *et al.* [20] propose a local search around a partial solution up to a predefined distance, like the local branching in [21]. The partial solution is generated consisting of predefined amounts (k_0, k_1) of zeros and ones. This approach might be less costly than the training selective classifier but requires finding the best combination of k_0 and k_1. Another similar modeling approach [19] is based on the computation of a bias vector per instance by averaging the decision values in the collected solutions with a MIP solver. A GNN classifier is trained to learn a binary vector obtained from this

bias vector for a given bias threshold. Afterwards, B&B search is guided by selecting open B&B nodes in the order of the sum of fixed variables' class probability (softmax output) of the trained GNN model.

Apart from the aforementioned studies, Shen *et al.* [23] generate a training dataset by obtaining an optimal solution per small-scale instance of a given CO problem. A GNN model is trained to learn how likely a binary decision variable is 1 in the optimal solution. The predicted probabilities by the 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. Later, Huang *et al.* [22] propose a bi-level framework to assign the decision variable scores in the end-to-end learning approach in [23] For a given coverage rate, a subset of the decision variables with the highest class probabilities is fixed to the predictions. Then, a secondary GNN model predicts the values of the unfixed decision variables. The experimentation over four CO problems shows that this framework enables the primal heuristics approach in [23] to reach high-quality solutions much shorter time for two CO problems. After obtaining the first feasible solution, both of these approaches do not continue on B&B search neither with the proposed variable selection guidance nor the default configuration of SCIP.

All presented studies in this section except [19] build on problem reduction or local search methods, which make them incomplete solution approaches as the typical characteristic of primal heuristics methods. Therefore the proposed methodologies obstruct the global search for better solutions and obtaining optimal solutions. Our methodology also benefits from a problem-reduction strategy in a warm-starting stage but includes a global search stage adopting the node selection strategy proposed in [19] which makes our approach a complete optimization method. Moreover, our study propound learning over the optimal solutions to the small-scale instances rather than the formerly presented approach, learning over multiple solutions per instance. We

provide insights into this modeling approach in Section 3.2. Another prominent design aspect differing from the presented studies is that our primal heuristics strategy to utilize the predictions of trained GNNs adopts an uncertainty-quantification approach. We use model uncertainty to drive both warm-start solution generation and B&B search. In this way, our methodology could be sounder in the utilization of the predicted decision values compared to using the prediction probabilities, which all presented methods rely on. Above all, our study comes out to design a scalable and generalized approach to solving CO problems at scale. On the other hand, only three studies [21, 22, 23] cover evaluation of the proposed methods over instances of scales larger than the training instances. Yet, these evaluations either do not include predictive accuracy results over larger-scale instances or do not demonstrate the generalization ability of the proposed modeling for all benchmarked problems. These studies reach a slightly better performance in solving some of the experimented larger-scale CO problems in the benchmarking with 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 compared to SCIP [19] (see the performance gain comparisons over Gurobi and SCIP in the evaluations of [1] and [20]. In our study, we use CPLEX as the baseline solver and achieve substantial performance improvement with our proposed methodology of end-to-end learning for primal heuristics.

To sum up, we provide a comparative overview of the presented studies in this section alongside our study as shown in Table 1. In the next subsection, we discuss the adopted training data generation approaches, considering how they influence ML tasks and the downstream optimization task.

3.2 Training Data Generation in End-to-End Learning for 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

Table 1: An overview of the related studies and our study on end-to-end learning for primal heuristics.

Study	Training Data Generation	Problem Representation	ML Task	Primal Heuristics Strategy	Benchmarking
[21]	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 [1]	A set of solutions per problem instance	The bipartite graph with features extracted from the root node of B&B tree [15]	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: Tuned SCIP and Gurobi Problems: CORLAT, MIPLIB, NNV, GP Pack, GP Plan, and EG
[23, 22]	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 [21] Problems: CA, DS, MIS, and MVC
MIP-GNN [19]	A set of 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
[20]	A set of solutions collected per problem instance	The bipartite graph with features extracted from the root node of B&B tree [15]	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 [52]
- 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 [53]
- 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

to be learned by a supervised model is its optimal solution vector. However, CO problems hinder the traceability of training data generation for supervised models by achieving (near-)optimal solutions, as they are NP-hard [17, 19]. Therefore, a possible training data generation procedure is collecting a set of feasible solutions per

instance without necessitating an optimal solution to be in the set, as done in the recent studies on learning primal solutions for CO. However, training data generation procedures prescribed in those studies lead ML models to different learning tasks (detailed in Section 3.1) 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 of end-to-end learning for primal heuristics 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, the aforementioned training data generation without considering the presence of the listed solution types might make the model avert learning 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. Kotary *et al.* [54] 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, adopting the end-to-end learning approach for MIP, Han *et al.* [20] show that fixing variables to their predicted values to get a partial solution can form a subproblem that barriers reaching optimal solutions during B&B search.

If multiple solutions are used as a target space for a problem instance to be learned by a model, the presence of the listed solution characteristics makes the model approximate obscure solutions. In other words, the more different the multiple solutions, the more predicted values for the decision variables represent a fuzzy solution as a whole. Moreover, setting up a target solution space to be learned from a pool of solutions per problem instance is subject to several hyperparameters. Considering MIP-GNN framework [19], one needs to determine the size of the solution pool, 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 improving 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. Tuning these parameters requires analyzing their effect on learning and optimization tasks. Similarly, the approach proposed in [21] requires several design aspects to be considered. The proposed training dataset generation is based on a local search around an initial feasible solution, and it requires to be tuned by finding the best scope size and stopping criteria. Additionally, the size of the partial solution constructed from the most confident predictions and the scope size of the local branching are 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.

CHAPTER IV

PROPOSED METHODOLOGY

The nature of optimization problems forms the design of data-driven optimization methods. A design aspect of such methods requires considering that the decision space of a constrained optimization problem can include different high-quality solutions having close or equal objective values. Accordingly, an adopted end-to-end learning approach for CO is training a non-autoregressive supervised or generative model to predict how likely each decision variable takes a value (zero or one) across a set of feasible solutions to a given problem instance. Ensuring the predictions by such models as a whole to be a feasible and high-quality solution is challenging since only prior embedded into these non-autoregressive models is the given problem information. These end-to-end learning approaches form a solution pool as a target space to be learned for each problem instance in the training set by collecting feasible solutions with an oracle, e.g., a MIP solver. It is not stipulated that the optimum solution must be in the solution pool. Nevertheless, the quality of the solution pool is consequential in end-to-end learning performance and inferencing good solutions for the downstream optimization task.

In this study, we propose a supervised learning of optimal solutions to downscaled instances of a given CO problem. We 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. So, instead of D, we generate small-scale problem (M') instances with n' variables by sampling 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.

Accordingly, our learning approach requires a GNN model that can successfully infer solutions to instances of M after training it on D'. Therefore, we employ MIP-GNN architecture [19], which is an advanced GNN model for end-to-end learning solutions to given MIP formulations as the bipartite graphs. In the following subsections, we first introduce MIP-GNN and then propose several supplementations and changes for it (Sections 4.1.1, 4.1.2, and 4.1.3). Lastly, we propose an uncertainty-based two-stage primal heuristic strategy utilizing the trained models to solve CO problems at scale (Section 4.2).

4.1 GNN Architecture for Combinatorial Optimization

MIP-GNN architecture depicted in Figure 3 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 [15] by using different graph convolutions and an additional layer named *Error*

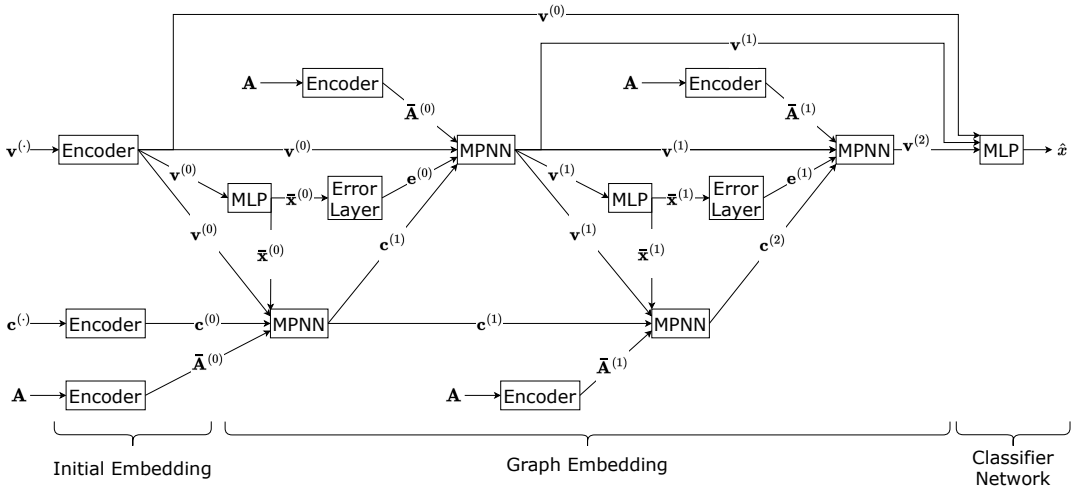


Figure 3: MIP-GNN Architecture with two-layer Graph Embedding

Layer, which propagates error messages between consecutive variable node embedding layers. In this section, we describe the MIP-GNN that uses Edge Convolution (EC)[1] [55] 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 unless specified.

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 $\mathbf{t}^{(v)}_j$ 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[2]. Let $\mathbf{d}^{(v)}_i \in \mathbb{Z}^n$, and $\mathbf{d}^{(c)}_i \in \mathbb{Z}^m$ be the vectors of constraint and variable node degrees, respectively. So,

[1]Definition of MIP-GNN in [19] 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.

[2]Note that $<$, $>$, and $\geq$ expressions can be converted to $\leq$ expressions. For example, $>$ expression can be converted to $\leq$ expression by adding ϵ quantity to the right-hand side and then multiplying the expression by -1. Therefore, $\mathbf{t}^{(c)}$ is a sufficient encoding to represent the type of any equality/inequality expression.

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 $\mathbf{c}_i^{(\cdot)} = [\mathbf{b}_i, \mathbf{t}_i^{(c)}, \mathbf{d}_i^{(c)}]$.

MIP-GNN builds on three blocks: (i) an initial embedding layer to map raw features of nodes to hidden embedding, (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, one each 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

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

where $\bigoplus$ is *mean* function, $\phi_c^{(k)}$ 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

$$\mathbf{v}_j^{(k)} = \mathrm{ReLU}\Big(\bigoplus_{i \in N(j)} \phi_v^{(k)}\big([\mathbf{v}_j^{(k-1)}, \mathbf{c}_i^{(k)}, \bar{\mathbf{A}}_{i,j}^{(k-1)}, \mathbf{e}_i^{(k-1)}]\big)\Big),\ \forall j \in V, \tag{5}$$

where $\mathbf{e}_i^{(k-1)}$ is the error signal computed by *Error Layer* defined in Equation 6a and 6b. 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 6a). 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 6b). 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.

$$\Delta = \mathbf{A}\bar{\mathbf{x}}^{(k)\top} - \mathbf{b} \tag{6a}$$

$$\mathbf{e}^{(k)} = f_e^{(k)}(\Delta) \tag{6b}$$

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 of 0 and 1 classes for each 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 in the optimal solution to the instance.

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 w.r.t. 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 7a 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 7b scales constraint bound $\mathbf{b_i}$ likewise. Equation 7c scales the objective coefficients by the maximum of their absolute values.

$$\mathbf{A}_{i,j} \leftarrow \frac{\mathbf{A}_{i,j}}{\max_{i \in C, j \in V} \left\{ |\mathbf{A}_{i,j}|, |\mathbf{b}_i| \right\}} \; \forall i \in C, \; \forall j \in V \tag{7a}$$

$$\mathbf{b}_i \leftarrow \frac{\mathbf{b}_i}{\max_{i \in C, j \in V} \{ |\mathbf{A}_{i,j}|, |\mathbf{b}_i| \}} \; \forall i \in C \tag{7b}$$

$$\mathbf{c}_j \leftarrow \frac{\mathbf{c}_j}{\max_{j \in V} \{ |\mathbf{c}_j| \}} \; \forall j \in V \tag{7c}$$

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

$$\mathbf{d}_j^{(v)} \leftarrow \frac{\mathbf{d}_j^{(v)}}{\max_{j \in V}\{\mathbf{d}_j^{(v)}\}} \; \forall j \in V \tag{8a}$$

$$\mathbf{d}_i^{(c)} \leftarrow \frac{\mathbf{d}_i^{(c)}}{\max_{i \in C}(\{\mathbf{d}_i^{(c)}\}} \; \forall i \in C \tag{8b}$$

PreNorm [15] 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 [56] We use it at the end of each MPNN layer before the node update function (ReLU) 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 $\mathbf{h}_i$, *GraphNorm* is defined as

$$\text{GraphNorm}(h_{i,d}) = \gamma_d \cdot \frac{\mathbf{h}_{i,d} - \alpha_d \cdot \mu_d}{\hat{\sigma}_d} + \beta_d, \tag{9}$$

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.

By computing representations for the nodes of a graph relative to each other with GraphNorm, the flow of information across layers of a neural network can follow a distribution during the inference for the target problem instance M, similarly with the inference for the downscaled instance M'. So, the message flows through the graph embedding block, and the node representations through the classifier network trained on small-scale CO instances can remain stable when inferring values of decision variables in larger CO instances.

4.1.2 Violation Layer

Considering inference 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 10 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 subsequently 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.

$$\Delta = \mathbf{A}\bar{\mathbf{x}}^\top - \mathbf{b} \tag{10a}$$

$$\bar{\Delta} = \mathbf{q}^{(c)}\Delta \tag{10b}$$

$$\mathbf{e} = \tanh\Big(\text{GraphNorm}\big(\text{MLP}([\bar{\Delta},\ \mathbf{t}^{(c)}])\big)\Big) \tag{10c}$$

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 [57]. Therefore, we opt for a combined aggregation [57] defined in Equation 11 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* operators are concatenated column-wise. Then, the concatenated message is linearly projected into a combined message $\mathbf{m}_{1\times h}$ 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 [17].

$$\mathbf{m}_{1\times h} = \Big[\mathbf{m}_{\mu}, \mathbf{m}_{\sigma}, \mathbf{m}_{\max}, \mathbf{m}_{\min}\Big]_{1\times 4h} \otimes \mathbf{W}_{4h\times h} \tag{11}$$

4.2 Uncertainty-based Primal Heuristics

As neural networks are approximation models, it is challenging that their outputs are guaranteed to satisfy a system of constraints [58, 5, 59]. Although the outputs as a whole are not guaranteed to be a feasible solution, fragments of them can be used to accelerate CO. Distinguishing which predictions are more reliable for constructing partial solutions typically depends on thresholding class probabilities or setting a coverage rate to accept the most confident predictions. The presented studies in Section 3.1 methods resort to a post-process depending on class/prediction probabilities for tuning the utilization of the predicted decision values to get up a partial solution. Moreover, some of them [19, 22, 23] also use the predicted probabilities for guiding B&B search. Class probabilities of classifier neural networks are traditionally derived by the softmax function, but it can cause unreliable class probabilities for predicting unusual samples due to its exponential quantification [24]. Considering this issue and the tuning procedure, 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 with primal heuristics.

Uncertainty quantification approaches such as Bayesian Neural Networks [60], Monte-Carlo Dropout [61], and Evidential Deep Learning (EDL) [24] 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. With this motivation, we resort to the model uncertainty as information on how predicted decision variable values are credible in solving CO problems. As EDL is a closed-form solution to

estimate model uncertainty accurately, we employ it for training GNN models in our methodology. In this way, our models can be configured considering their efficacy on the downstream optimization task while training them, we can directly use the model predictions in our primal heuristic strategy without a post-training process. We describe EDL Loss in the following.

A classification problem can be 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

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \sum_{k=1}^{K} \mathbf{y}_{jk} \big(\psi(S_j) - \psi(\hat{\alpha}_{jk}) \big) + \lambda \mathrm{KL}(\mathbf{y}_j, \hat{\alpha}_j), \tag{12}$$

where $\mathbf{y}_j$ is the one-hot encoded target value of decision variable j, ψ is digamma function, $\lambda \in \mathbb{R}$ is the regularization coefficient, and $S_j = \sum_{k=1}^{K} \hat{\alpha}_{jk}$ [24]. 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-based 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

An effective way to accelerate solving CO problems is by generating an initial feasible solution and continuing to search for the optimal solution by exploiting this solution. It is challenging that a predicted solution by an end-to-end learning model for CO is a feasible and high-quality solution. Instead, one can fix a portion of decision variables to their predicted values by the model, which each variable fixing reduces the space complexity by half. The best solution in the reduced 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 can be challenging. Adopting this heuristic method, we propose an uncertainty-based approach to select which decision variables are fixed to their predicted values. As model predictions with less uncertainty are more likely to be accurate, the predictions to be fixed are selected based on their associated uncertainty values.

Let $\mathbf{u}$ be the vector of uncertainties 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 Binary Cross Entropy (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 [19]. Accordingly, a set of cuts Ω is generated (Equation 13) 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 $\bar{u}$.

$$\Omega = \{x_j := \text{round}(\hat{\mathbf{x}}_j) \mid \mathbf{u}_j \leq \bar{u},\ j \in \mathcal{V}\} \tag{13}$$

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 the search 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 of a trained model for a larger-scale instance are as accurate as the inference for the testing instances at the scale of training instances, one can determine what percentile of predictions are totally/almost accurate based on the validation set. We empirically validate this claim by presenting the predictive performance results in Section 6.1. Besides, setting $\bar{u}$ accepts 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 by a model 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 end-to-end learning 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 first iteration. Unless M_{sub} is infeasible or a feasible solution cannot be found in t, 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.

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

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. Note that if $M_{sub}^{(i+1)}$ is infeasible due to $\Omega^{(i+1)}$, then $M_{sub}^{(i)}$ 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. Our repair algorithm (2) to restore the infeasible sub-MIP M' at the last iteration of UPR algorithm is based on *Conflict Analysis* [62], which has been implemented in off-the-shelf MIP solvers[3]. *Conflict Analysis* methods diagnose conflicts among the constraints in MIP.

[3]Conflict analysis is managed in the off-the-shelf MIP solvers such as Conflict Refiner in CPLEX, Conflict Analysis in SCIP, and Irreducible Inconsistent Subsystem in Gurobi.

MIPs can be refined by removing the set of constraints that are identified as causing infeasibility.

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 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, all 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.

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 ← false          // Initialize a predicate indicating if M' is feasible or not
2  t = 0                                                  // Initialize elapsed repair time
3  while t < T  &  ||Ω|| > 0 do
4  |  (Ω_c, t) ← ConflictAnalysis(M', T − t)
5  |  if |Ω_c| = 0 then
6  |  |  repaired ← true
7  |  |  M_f ← M'
8  |  |  break                                                          // ends the loop
9  |  end
10 |  Ω' ← GetConflictingCuts(Ω_c, Ω)
11 |  M' ← RemoveCuts(M', Ω')
12 |  Ω ← Ω \ Ω'
13 end
14 if repaired = false then
15 |  M_f ← RemoveCuts(M', Ω)
16 end
17 return M_f
```

4.2.2 Uncertainty-Guided B&B Search

We adopt *Guided node selection* technique proposed in [19] where B&B nodes are prioritized based on the predictions of the trained MIP-GNN model. Khalil et al. [19] 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 BCE models, we propose the utilization of evidence values for decision classes computed by EDL models using $\mathbf{u}_{j,k} = 1/\alpha_{j,k}$ values 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 14, 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.

$$s_b = \sum_{j \in \mathcal{J}_b,\, k = x_j^{(b)}} (1 - \mathbf{u}_{j,k}) \tag{14}$$

Since s_b increases as the depth of nodes increases, B&B search starts diving toward the predicted solution vector. After selecting 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 uncertainty. Henceforth, partial solutions (i.e., intermediate B&B nodes) that include the predicted decision values with high confidence are retained until the B&B nodes fixing the decision variables predicted with high uncertainty are explored. This exploration continues by flipping the values of decision variables with high prediction uncertainty at first, posing oscillations around the decision space where the model predictions are precise. Using this uncertainty-based node selection strategy, B&B algorithm manages the trade-off between exploitation and exploration by following a search policy that is configured with the predictions and associated uncertainty values computed by the trained model.

CHAPTER V

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 given trained MIP-GNN with EC into CPLEX and uses their predictions and associated uncertainty values within the proposed primal heuristics to solve CO problems. Our experimentation covers mainly 12 different CO pipelines per CO problem, thus, 60 pipelines in total. These pipelines are configured with the following three settings.

- **Model Architecure**: Models with Violation Layer and Error Layer are denoted by EC+V and EC+E, respectively. The bare models denoted by EC.
- **Loss Function**: Models trained with Binary Cross Entropy and Evidential Deep Learning loss functions are denoted by BCE and EDL, respectively.
- **Primal Heuristics Strategy**: CO pipelines that apply Uncertainty-based Problem Reduction (UPR) for warm-starting before the global search with Uncertainty 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. 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 the CO pipelines that deploy the pre-trained

MIP-GNN models using EC layers released in [19] under a 30-minute solve time limit per instance. The pre-trained MIP-GNN models with and without Error Layer are similarly denoted by MIPGNN and MIPGNN+E, respectively. The solve time limit includes the elapsed time in data preprocessing and inference time to generate predictions for decision variable values. Hence, the remaining time limit for the CO pipelines to solve each instance is less than 30 minutes.

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) [15][1]. The other two datasets are from [19] and include CO problems harder than the previous three, which are Fixed-Charge Multi-Commodity Network Flow (FCMNF) [63], and Generalized Independent Set (GIS) [64][2]. Table 8 in Appendix B includes the number of decision variables and constraints in the training datasets we generated for five CO problems.

Each CO problem dataset contains 1000 training, 200 validation, and 50 test instances which are at the same problem scale. To investigate the capability of our methodology to scale CO, we generated four transfer datasets of MSC, CA, and MIS problems which contain 2, 4, 8, and 16 times the number of variables in the training instances. The problem scales of the testing and four transfer datasets are denoted by 1x, 2x, 4x, 8x, and 16x, respectively[3]. 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 [19]. We call these actual datasets as

[1]We used the codebase in https://github.com/ds4dm/learn2branch-ecole/tree/dev to generate the benchmark datasets.

[2]The pre-trained MIP-GNN models are available in https://github.com/lyeskhalil/mipGNN, which also includes FCMNF and GIS dataset generator coding we used.

[3]Size of MIPs can also be w.r.t. $n \times m$ or number of nonzeros in $\mathbf{A}$ [53], which would indicate a much greater quantification for the scales of transfer datasets.

transfer datasets throughout our evaluation. Our models are trained on the down-scaled 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 these actual datasets; therefore, this benchmarking measures their testing performance, whereas our models' transfer performance.

Each transfer dataset of MSC, CA, MIS, FCMNF, and GIS comprises 50 instances. All results provided throughout the evaluation are presented based on the statistics of evaluation metrics for these 50 instances. Size of the problems, CPLEX solve time for collecting the optimal solutions for training and validation instances and elapsed time in the data preprocessing are given in detail at Appendix B. Alongside the 30-minute benchmarking runs, we also ran the default configuration of CPLEX 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 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. In the following, we provide semi-formal definitions of the benchmarking problems.

Minimum Set Cover (MSC) We define MSC with an example MILP representation. The objective (Equation 15) is to minimize the total cost associated with the selection of a set of items, which is represented through binary variables x_j (17). This optimization problem is subject to the constraint (16) that every required set S_i must contain at least one item that has been selected. The parameter c_j denotes the cost of selecting the j^{th} item, whereas m and n correspond to the number of mandatory sets and available items, respectively.

$$\text{min.} \quad \sum_{j=1}^{n} c_j x_j \tag{15}$$

$$\text{s.t.} \quad \sum_{j \in S_i} x_j \geq 1 \quad \forall i \in \{1, 2, \ldots, m\} \tag{16}$$

$$x_j \in \{0, 1\} \quad \forall j \in \{1, 2, \ldots, n\} \tag{17}$$

Combinatorial Auction It is a type of auction in which bidders can bid for bundles or combinations of items rather than just individual items. The goal is to determine the optimal allocation of items among the bidders to maximize the total value of the items assigned, subject to various constraints. In CA, each bidder submits a bid for each possible bundle or combination of items that they are interested in. The auctioneer then determines the allocation of items among the bidders to maximize the total value of the items assigned, subject to constraints such as each item can only be assigned to one bidder and each bidder can only be assigned one bundle of items.

Maximal Independent Set It is a widely studied problem in graph theory that aims to identify the largest possible set of vertices in a given graph such that no two vertices in the set share an edge. Formally, given an undirected graph $G = (V, E)$, an independent set $S \subseteq V$ is defined as a set of vertices such that no two vertices in S are adjacent (i.e., there is no edge connecting them). A maximal independent set S is an independent set that is not a proper subset of any other independent set of G. In other words, we cannot add any vertex to S while maintaining its independence property. MIS problem is of practical importance in many fields, including computer science, network analysis, and social network analysis.

Fixed-Charge Multi-Commodity Network Flow It is the problem of optimal routing of each commodity from its origin to its destination, ensuring that flow

conservation and capacity constraints are satisfied. Specifically, in FCMNF, each commodity is associated with a unique source and sink node. A fixed cost is incurred for each network link, regardless of the quantity of flow that traverses it. The objective of FCMNF is to minimize the total cost of routing all commodities from their respective origins to their destinations.

Generalized Independent Set It is a CO problem that deals with a given graph with vertex revenues and a set of removable edges with associated removal costs. The objective is to determine an independent set that maximizes the net benefit, which is defined as the difference between the total revenues collected from the vertices included in the independent set and the costs incurred for the removal of any edges that have both endpoints in the independent set. To solve the GIS problem, one must identify the independent set that yields the maximum net benefit. This is a complex problem that requires a thorough exploration of the feasible solution space, considering both the benefits and the costs involved in removing edges from the graph.

5.2 Implementation Details

For our MIP-GNN 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 architecture size of the pre-trained MIP-GNN for fair benchmarking.

All models are trained using ADAM optimizer [65] with eight batches in 24 epochs. The learning rate is initialized at 0.0001 and managed using *One-cycle* learning rate scheduler [66]. 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 PyTorch implementation of it[4].

[4] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html

Evidential Deep Learning Regularization We conducted a grid search of the regularizer parameter λ of EDL Loss by observing its effect on the downstream optimization task for the largest-scale dataset 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 in a short time; therefore, we limited UPR algorithm to run for three minutes in total and set its iteration limit to five for all datasets. We observed that the first 3-4 iterations of UPR 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 [19]. 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 true in cases where a model is employed within a MIP solver for a downstream optimization task [48, 1]. In such workflows, loss 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

$$\text{gap}(t) = \frac{|PB - DB|}{\max\{|PB|, |DB|, \epsilon\}} \times 100, \tag{18}$$

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

$$\text{p-gap}(\tilde{\mathbf{x}}) = \frac{|\mathbf{c}^\top \tilde{\mathbf{x}} - c^\top \tilde{\mathbf{x}}^*|}{\max\{|\mathbf{c}^\top \tilde{\mathbf{x}}|, |\mathbf{c}^\top \tilde{\mathbf{x}}^*|, \epsilon\}} \times 100. \tag{19}$$

In our study, primal gap values are calculated w.r.t. the best objective value attained by 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 methods can be measured considering the trade-off between solution speed and quality [67] 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

$$\text{p-integral}(T) = \sum_{i=1}^{I} \text{p-gap}(\tilde{\mathbf{x}}^{(i)}) \cdot (t_i - t_{i-1}), \tag{20}$$

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 PB. Nevertheless, we remark that the optimality gap results generally led to consistent conclusions on ranking the models with respect to the primal gap in our experimentation.

For the benchmarking and ablation study of the models, we also calculate an improvement rate (better if higher) to quantify how much a model improves an optimization metric relative to the performance of another model or CPLEX. For example, the 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 given model configuration, respectively.

CHAPTER VI

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 [19]. Moreover, we present an ablation study to analyze the effect of each CO pipeline setting in the particular sections 6.4, 6.5, 6.6, and 6.7. These sections investigate how much each model configuration facilities solving the benchmarking CO problems and can highlight the advantage of different model configurations on different problems.

We evaluate our study mainly on the largest-scale instances of each problem since the performance differences of the CO pipelines become more prominent as the problem size increases. Nevertheless, we provide the figures presenting the performance of all models for each scale of the datasets in Appendix C to exhibit how our methodology can scale compared to the performance of the default configuration of CPLEX (we refer to it as only 'CPLEX' from now on). Since we set the scale of training instances to be easily solvable with CPLEX, all CO pipelines generally achieve the optimal solutions to testing instances and provide the evaluation metrics 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 experimental results in terms of the evaluation metrics defined in Section 5.3 for 50 largest-scale transfer

instances of each CO problem. Figure 4 presents the boxplot distributions of the optimality gap values attained by each CO pipeline configuration with NS and UPR+NS strategies in 30 minutes. Likewise, Figure 5 and 6 present the distributions of primal gap and primal integral values, respectively. In all figures, the boxplot distributions of metrics for CPLEX are placed together with the CO pipelines applying only NS strategy. Figure 7 present the distributions of primal gap values of the warms-start solutions found with UPR.

The optimality gap and primal gap results in Figures 4 and 5 set out the contribution of the CO pipelines to solve each CO problem at scale by wide margins compared to CPLEX. For MSC, CA, MIS, and FCMNF problems, the stages of the proposed primal heuristics strategy are the major factor in boosting optimization performance, while the effect of the model configurations is more dominant for GIS problem. Yet, we observe that EC+V+EDL is the model that mostly leads to better optimization performance across the CO problems. Besides, the advantage of training the models over the downscaled instances is figured out considering the benchmark with the pretrained MIP-GNN models.When only NS strategy is applied, the CO pipelines using our models achieve on-par or better optimization performance for FCMNF and GIS datasets compared to the MIP-GNN framework, 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 heavily depends on using partial solutions that are feasible and do not hinder optimality, it yields low-quality warm-start solutions using the MIP-GNN models (see Figure 7), which is further detailed in Section 6.3. In addition to the final solution quality, the CO pipelines offer time-efficient solution quality throughout the run time with fairly lower primal integral values when Figure 6 is considered.

Lastly, Figure 8 exposes how each CO pipeline configuration proceeds in solving each problem. To present the primal gap value averaged across all instances per

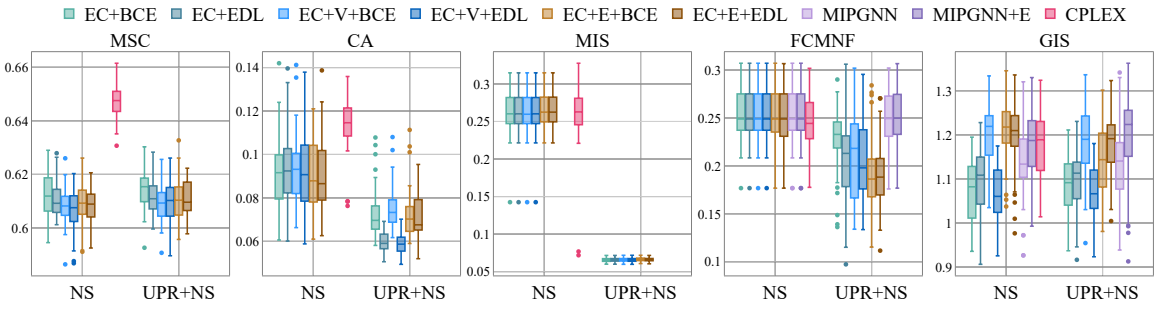


Figure 4: The Boxplot Distributions of Optimality Gap Values for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets

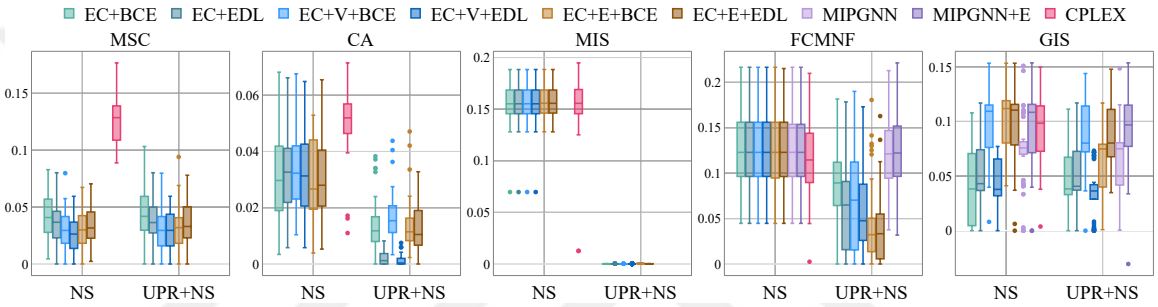


Figure 5: The Boxplot Distributions of Primal Gap Values for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets

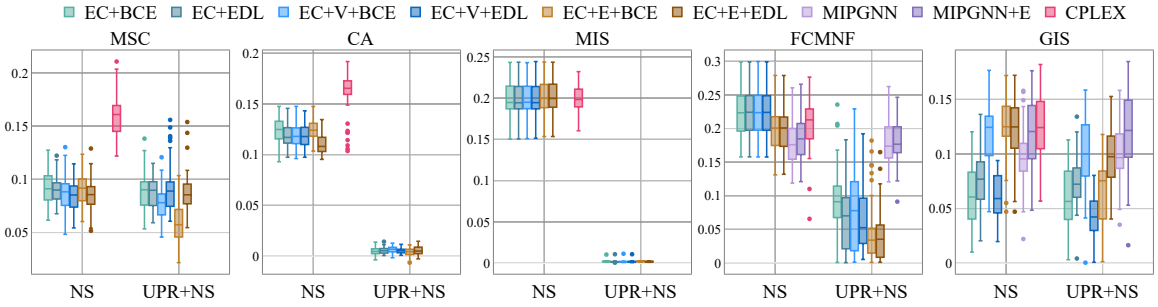


Figure 6: The Boxplot Distributions of Primal Integral Values for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets

second by a CO pipeline, primal gap values are linearly interpolated through the time interval $[1, 1800]$ in the figure. 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 CO pipeline configuration is effective for solving each problem in the following subsections.

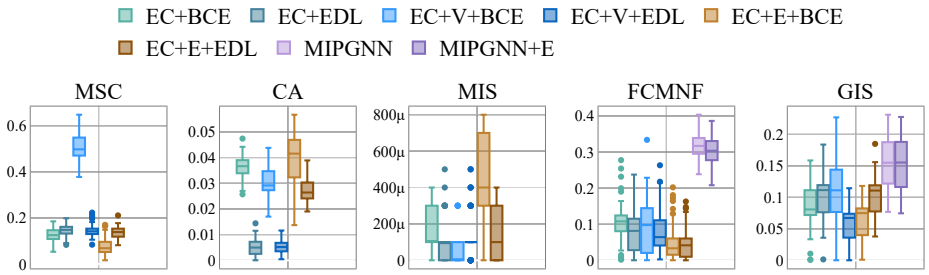


Figure 7: The Boxplot Distributions of Primal Gap Values by the Warm-start Solutions Found with UPR for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets

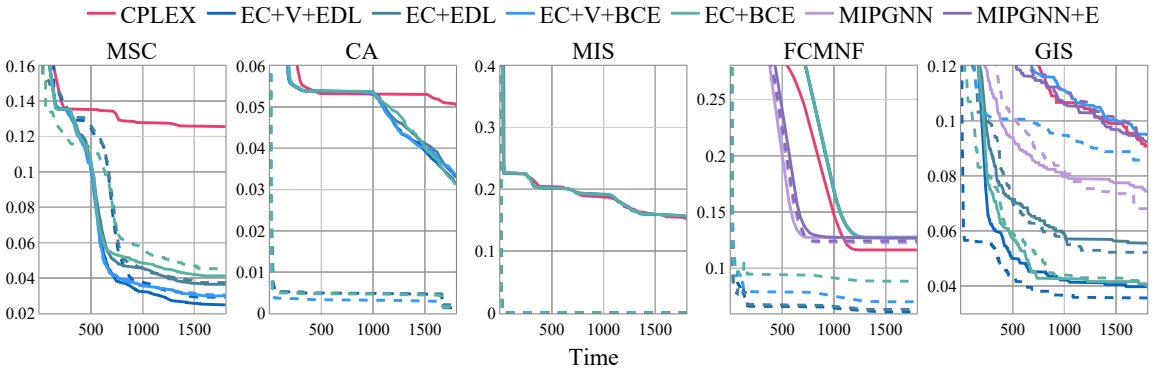


Figure 8: The Average Primal Gap Curves for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x Transfer Datasets (NS and UPR+NS results are denoted by solid and dashed lines, respectively.)

6.1 Generalization on Larger-scale CO Problems

Before evaluating the performance of the CO pipelines regarding optimization performance, we investigate whether the proposed end-to-end learning for CO can exhibit 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 that do not include the normalization techniques proposed in Section 4.1.1 to reveal how the normalization layers improve predictive accuracy. Considering the generalization ability of the models, the proposed primal heuristic methods rely on the uncertainty of predictions. Therefore, we take into account the accuracy of the predictions with the

associated uncertainty values together while evaluating the generalization ability of the trained models on larger instances. Sorting the predictions in ascending order of associated uncertainty values, we measure how accurate each percentile of predictions are. Figure 9 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.

We use the best solutions obtained through the long CPLEX runs to compute the accuracy of the predicted solutions. However, it is worth noting that our models are parameterized with the optimal solutions to the training (downscaled) instances; hence, the best solutions for the transfer datasets do not represent actual target solutions to measure the predictive performance of the models precisely[1] if they are suboptimal. Particularly, the quality of the target solutions for MIS-16x obtained with CPLEX in eight hours is worse than that of the warm-start solutions generated with UPR, so the curves for MIS-16x dataset in Figure 9 get lowers slightly, and poorly represent the actual accuracy.

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 accuracy

[1]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.

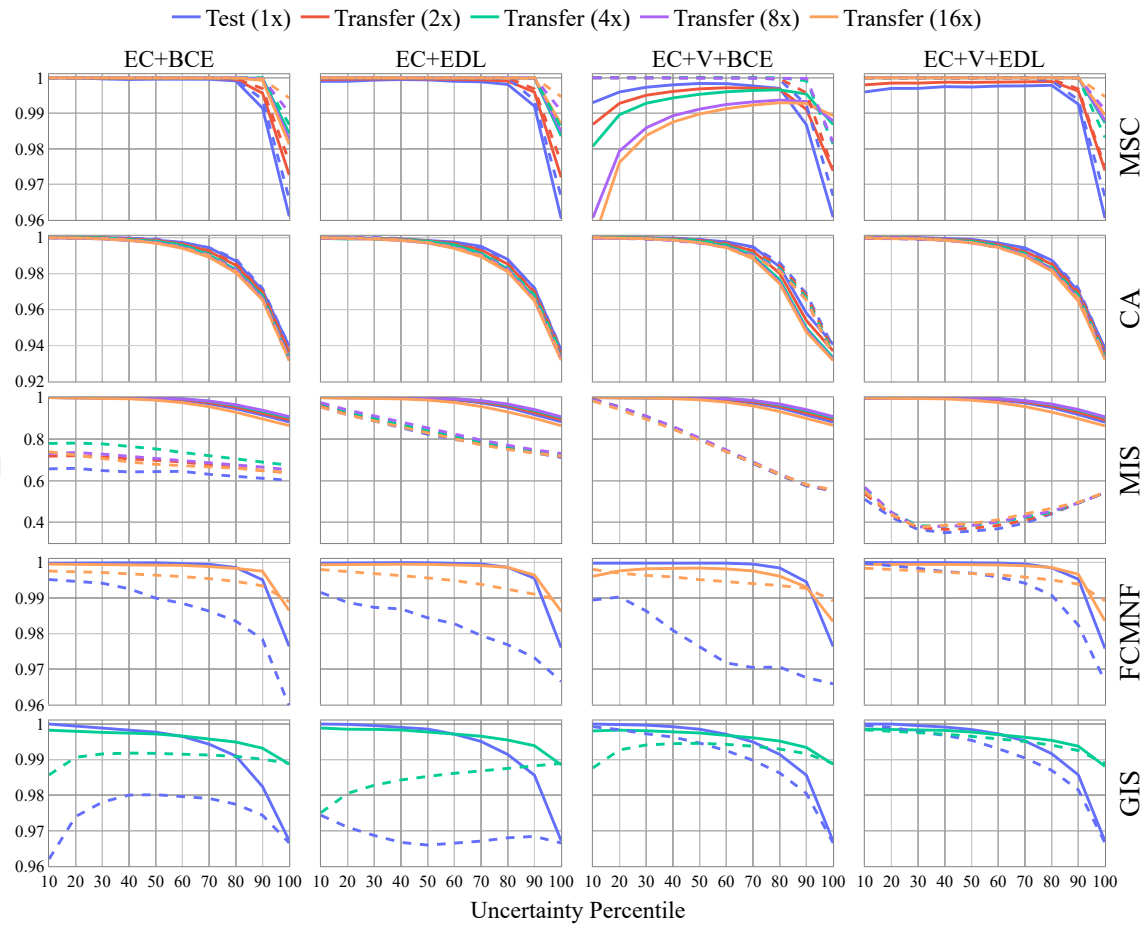


Figure 9: The Average Accuracy of the Predicted Decision Values by the Models with (Solid Curves) and without (Dashed Curves) the Proposed Normalization Techniques in Each Uncertainty Percentile for All Datasets

of the models for testing datasets matches with the transfer datasets. Moreover, the uncertain predictions by the models for the transfer datasets may cause slightly diverging accuracy levels, but not lower than the testing levels mostly. We hereby infer that the proposed modeling approach enables the models to generalize on larger-scale instances of the benchmarking CO problems. Furthermore, it is observed that the accuracy of the predictions within the first 50 percentile of uncertainty distributions is generally near 100% for the testing and the transfer datasets by all models. This empirical result brings an admissible uncertainty threshold to use in Uncertainty-based Problem Reduction. For each model of each problem, we set the uncertainty

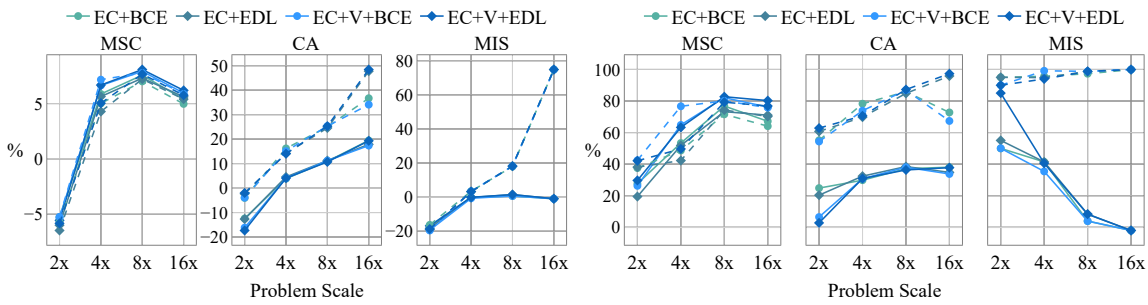
threshold to the median uncertainty value of the predictions for the validation dataset of the problem, which only includes the downscaled instances at size 1x, and use it for both testing and transfer datasets of the problem. Hence, we conclude that the uncertainty values are admissible to utilize them in the proposed primal heuristic strategies. On the other hand, it is an exceptional result that the uncertainty values of EC+V+BCE models for MSC and GIS problems are misleading about the accuracy of highly confident predictions, so its uncertainty percentile-accuracy curve does not start with the highest accuracy level. This also occurs slightly for the FCMNF transfer dataset. In the earlier phase of the development, it is observed such highly confident but misleading predictions (especially for MSC datasets) by the models that do not include any dropout layer in the classifier network. Then this is attributed to the fact that these models overfit to initial node features. For example, MSC models were over-biased to the objective coefficients and predicted the decision value of one for almost all of the decision variables that have low objective coefficients, so the predicted solutions include values of one much more than the optimal solutions. Therefore, we regularized all models by applying 0.1 dropout probability in each layer of their classifier neural network. Although EC+V+BCE model for MSC problem requires a higher dropout rate or further regularization, we still keep it with a 0.1 dropout on purpose to demonstrate how it negatively affects the downstream optimization performance. EC+V+BCE model leads to generating worse warm-start solutions with UPR algorithm as the scale of MSC problem increases (See Figure 17a), and the CO pipelines using EC+V+BCE for FCMNF and GIS problems prominently underperform (See Figures 4 - 6).

6.2 Benchmarking with CPLEX

We quantify the performance improvement by the CO pipelines with the improvement rates for optimality gap and primal gap metrics relative to the performance of CPLEX

across scales for transfer datasets of MSC, CA, and MIS problems in Figure 10. Similarly, Figure 11 includes the bar charts of the improvement rates for the transfer datasets of FCMNF and GIS problems.

Solving MSC and CA problems with both NS and UPR+NS strategies substantially decreases primal gap values by around 80% as the scale of problems increases. Differently for MIS problem, the benefit of NS fades as the problem scales. It is clarified from Figure 8 where all model configurations with NS strategy reach similar average primal gaps with CPLEX 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. Generally, the non-decreasing pattern of improvement rates across the scales of MSC, CA, and MIS problems displays that our models offer a scalable performance compared to CPLEX.

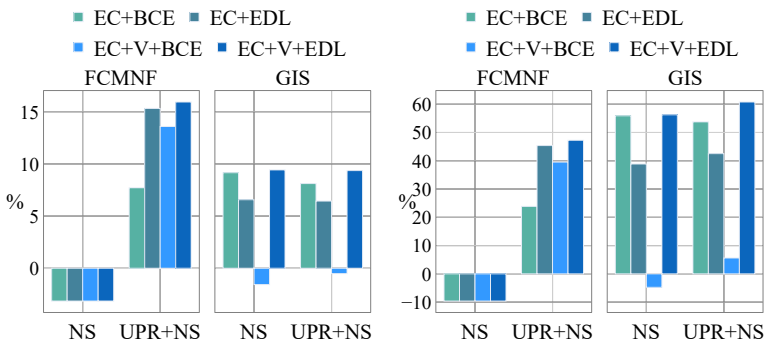


(a) The Average Improvement Rates for Optimality Gap

(b) The Average Improvement Rates for Primal Gap

Figure 10: The Average Improvement Rates 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.)

Moreover, the proposed primal heuristics can scale in solving the actual-scale instances of FCMNF and GIS as the more complex CO problems. Khalil et al. [19] report that the testing performance of MIP-GNN framework with the 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



(a) Average Improvement Rates for Optimality Gap

(b) Average Improvement Rates for Primal Gap

Figure 11: The Average Improvement Rates by the Model Configurations for the FCMNF-16x and GIS-4x Datasets

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 9).

In addition to quantifying the performance of the CO pipelines and CPLEX in terms of the presented metrics, we analyze the time efficiency of our methodology (in Table 2) by measuring when CPLEX can find solutions that have the best objective values obtained through the CO pipelines. The last five columns of Table 2a and 2b for UPR+NS and NS pipelines include the values averaged over the four models for each scale of all problems. Since FCMNF and GIS problems only have two datasets, the cells of the remaining problem scales are filled by '-'. 'NA' denotes that CPLEX cannot obtain reach the solution quality of any CO pipelines for any problem instance in the given long runtime. It is observed from Table 2a that even if the time limit of the default CPLEX configuration runs is doubled at each problem scale, the 30-minute solution quality of the CO pipelines for most instances cannot be obtained.

Particularly, a solution that is at least as good as that of the CO pipelines cannot be found with CPLEX for any instance of MIS-8x and MIS-16 datasets in respectively 4 and 8 hours (it is why zero success rates are in the table).

For the instances in which CPLEX can achieve solutions (with longer time limits) that are at least good as that of the CO pipelines, Table 2b presents the average time efficiency through the CO pipelines relative to CPLEX performance. The presented values are the proportion of the average elapsed time with CPLEX to obtain the individual best objective values of the CO pipelines (see Table 11 at Appendix C.1) to the average elapsed time of the CO pipelines to obtain their individual best objective values (see Table 10 at Appendix C.1). For example, for MSC-16x instances, CPLEX finds solutions as good as those achieved by the NS pipelines only when running 19.97x (on average) the elapsed time with these pipelines. See an example of how the time efficiency is calculated in Appendix C.1. As the scale of each problem increases, the time savings by the CO pipelines dramatically rises. The increasing efficiency trends across problem scales support that the proposed methodology offers scalable and generalizable GNN-based end-to-end learning for solving CO problems.

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 [19]. 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). MIP-GNN models are trained with BCE Loss and use the class probabilities to calculate the confidence scores for B&B nodes in [19]. 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. So, the

Table 2: (2a) 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 (2b) the average time efficiency of these pipelines relative to the default CPLEX performance.

(a) The average success rate (%) of CPLEX for long runs in finding solutions that have equal objective values with ones obtained by the four CO pipelines for each scale of all problems.

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) The average time efficiency (better if higher than 1.0) of the four CO pipelines relative to the long-run CPLEX performance for each scale of all problems.

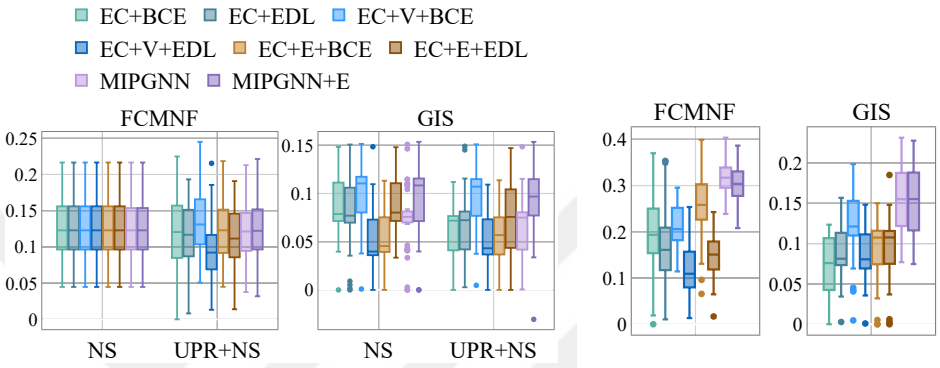
Problem	Strategy	1x	2x	4x	8x	16x
MSC	**NS**	0.82	1.35	3.69	12.41	19.97
	UPR+NS	0.91	1.52	3.16	10.47	18.82
CA	**NS**	0.62	1.01	2.54	4.74	1.77
	UPR+NS	1.44	1.34	3.40	8.02	11.94
MIS	**NS**	2.71	1.26	2.19	2.60	2.37
	UPR+NS	3.71	7.77	61.26	NA	NA
FCMNF	**NS**	0.84	-	-	-	3.71
	UPR+NS	1.96	-	-	-	7.33
GIS	**NS**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

transfer performance of our models is actually compared with the testing performance of MIP-GNN models.

Figure 12a 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 12b 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 in [19] about the variable fixing strategy of Neural Diving (see Section 3.2). 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) Primal Gap Values by the CO Pipelines

(b) Primal Gap Values of Warmstart Solutions Found with UPR

Figure 12: The Boxplot Distributions of (12a) Final and (12b) Warm-starting Primal Gap Values by the CO Pipelines Using the Models with 4-layered GNN and *mean* aggregation for FCMNF-16x and GIS-4x Transfer Datasets

Consequently, our models' transfer performance can achieve and even exceed the MIP-GNN models' testing performance. 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 generation and model training[2]as well as less tuning burden (due to having fewer hyperparameters) compared to the

[2]We cannot make a direct comparison with the experimentation in [19] for the elapsed times in the training dataset generation and the model training since the experiment environments are different. Yet, we can enounce 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 8 at Appendix B) whereas Khalil *et al.* [19] 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 leads to certainly takes much less time.

MIP-GNN framework, which includes a bias threshold for training and exploiting the models and several hyperparameters for data collection (see Section 3.2).

6.4 Effect of Warm-start Solutions

To analyze how the warm-start solutions contribute to scale CO problems, we set separate CO pipelines by ablating UPR algorithm from the proposed primal heuristic strategy. In our experimentation, UPR algorithm achieved feasible sub-MIPs for all instances of all datasets except FCMNF-16x. Table 3 presents the numbers of the last iteration $M_{sub}^{(N)}$ sub-MIPs that are infeasible for FCMNF-16x instances. In the table, the model configurations are classified based on the loss functions, aggregations types, the combined aggregation (*comb*) and *mean* aggregation, and the number of GNN layers (k). For all these instances, Algorithm 2 successfully restores the sub-MIPs in one minute and returns a final sub-MIP, and UPR algorithm achieves a feasible solution for it.

Table 3: The number of instances in which a feasible sub-MIP is not obtained in UPR algorithm for 50 FCMNF-16x instances.

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	-

Considering the primal gap distributions in Figure 5, the warm-start solutions obtained with UPR algorithm are highly influential in solving CA, MIS, and FCMNF problems. Figure 8 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 2b

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 4, 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 8, 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 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 5, primal gap values by the CO pipelines using 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 effect of Violation Layer and Error Layer on solving are divergent. While EC+V is superior to EC, EC+E models offer much better performance with UPR+NS strategy for FCMNF-16x instances. But EC+E models significantly underperform by a wide margin for GIS-4x instances. On the side of EC+V for GIS, both NS and UPR+NS pipelines using the 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 the predictions of EC+V models (especially EC+V+EDL models), which are much better for CA-16x, MIS-16x, and GIS-4x instances, whereas using EC+E models is slightly more beneficial for MSC-16x, and FCMNF-16x instances (see Figure 7). 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 measure how the uncertainty quantification through EDL contributes to the performance of the proposed primal heuristic strategy, we present Table 4, which includes the average improvement rates in terms of each metric by the models trained with EDL Loss over those trained with BCE Loss. The improvement rates are calculated based on the primal integral, the primal gap, the optimality gap, and the warm-starting primal gap values averaged across EC+V and EC models. The last column includes Imp. for the primal gap values of the warm-start solutions found with UPR algorithm. The optimality gap and primal gap improvement rates for the test datasets are excluded since all models achieve (near) zero gap values. Negative percentages indicate that the CO pipelines using EDL models underperformed. Although some of the CO pipelines using BCE models provide marginally lower primal integral values compared to those using EDL models for the test instances of CA, MIS, FCMNF, and GIS problems, utilization of uncertainty-quantification becomes notable for the transfer datasets. The contribution of the EDL models is significant in solving 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 datasets 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 4: The average improvement rates (Imp.) by the models trained with EDL Loss over those trained with BCE Loss for the testing dataset and the largest-scale transfer dataset of each problem.

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 by the related studies. We benchmarked these two groups of models over the testing and the largest-scale transfer datasets of each problem. Table 5 includes the improvement rates by the models using *comb* over those using *mean* aggregation. These rates are calculated based on the primal integral, the primal gap and the optimality gap values 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 in the table indicate that the CO pipelines using the models with *comb* underperformed. Similarly, Table 6 presents the improvement rates calculated based on the warm-start primal gap values averaged over EC+V and EC models.

The *comb* models generally provide better generalization for solving CA, MIS, and FCMNF problems compared to the models using *mean* aggregation. The positive 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. When UPR+NS strategy is applied, the optimization performance by the pipelines using *comb* models 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 by the models using *comb*.

Unlike the improvement rates for the other three datasets, *comb* degrades generalization to MSC-16x and GIS-4x instances. Although the CO pipelines using the

models with *comb* provide better testing performance, the models using *mean* aggregation generalize on larger-scale instances of MSC better. For solving GIS-4x, only those trained with BCE Loss perform better, while EDL Loss improves the generalization of the models with *comb*. It indicates that EDL can lead the regularization of the models that use *comb* for GIS problem. Nevertheless, training models on the downscaled instances of MSC and GIS problems with a complex aggregation seem to require further regularization, such as using narrower hidden layers, applying dropout in the used MPNNs, or a higher probability of dropout for the classifier network layers. Overall, our ablation study in this section figures out that GNNs using different aggregation operators can learn useful patterns for solving CO problems, but stronger regularization could be needed to ensure performance improvement.

Table 5: The average improvement rates (Imp.) by the models using the combined aggregation (*comb*) over those using *mean* aggregation for the testing dataset of each problem and MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x transfer datasets.

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

Table 6: The improvement rates (Imp.) by the models using the combined aggregation (*comb*) over those using *mean* aggregation for MSC-16x, CA-16x, MIS-16x, FCMNF-16x, and GIS-4x transfer datasets and the test dataset of each problem for the primal gap values of warm-start solutions generated in UPR algorithm.

Problem Type	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

CHAPTER VII

DISCUSSION

In this section, beyond the results discussed up to now, we aim to explore the generalization capability and limitation of the proposed methodology for solving CO at scale by underlining what factors determine its efficacy. Since the required model complexity for generalizing any CO problem is likely very dependent on the problem structure, it is challenging to give any precise quantitative estimates a priori.

For generating the training datasets, we determined the scale of the training instances at a level that CPLEX can solve exactly in a reasonable amount of time. In this study, we do not draw an analysis for the relation between the magnitude of downscaling and its effect on the downstream task. Yet, our experimentation presents an exemplary optimization performance for FCMNF and GIS problems by downscaling them 16 and 4 times, respectively, in the benchmarking with MIP-GNN models. However, a more advanced design of the proposed CO scaling methodology could consider how much a CO problem is downscaled to generate a training dataset for efficient end-to-end tuning problem-specific model configurations. There are two factors of learning solution patterns in the downscaled instances, which set the benefit of inferring solutions to larger instances and solving them in a data-driven manner. Firstly, the interdependency and/or interrelation of decision variables in a problem may extend as the problem size increases[1]. The benefit of learned solution patterns in solving a given problem can weaken as the scale of training instances reduces.

[1]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 ever-increasing scales 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 16 times larger instances (see Figure 8)

Therefore, one can tune the magnitude of the downscaling by an analysis of its effect on downstream optimization task with the cost of collecting the optimal solutions to training instances for a given problem.

Another related factor is the depth of GNN: How deep should GNN models (that are trained on small-scale instances of a given problem) 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. But 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 less number of decision variables are likely to fade out of a certain neighborhood depth. So, the depth of GNN can be 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.

CHAPTER VIII

CONCLUSION

This study provides an extensive overview of end-to-end learning with GNNs for primal heuristics to solve CO problems efficiently and particularly addresses the scalability of these modeling approaches. Accordingly, this study builds on supervised GNNs for learning the optimal solutions to downscaled instances of a given large-scale CO problem. The proposed training approach and improvements on a recent GNN model for CO allow generalization on larger-scale instances. The proposed primal heuristics exploits the trained models by generating warm-start solutions and guiding B&B search based on the model predictions and calibrated uncertainty values with Evidential Deep Learning. This methodology provides substantially improved performance in solving MSC, CA, MIS, FCMNF, and GIS problems compared to the performance of default CPLEX configuration as the problem size increases. The benchmarking with 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 large-scale instances.

8.1 Future Work

The proposed methodology requires less costly training data generation and comes with less burden of the hyperparameter tuning compared to the presented related studies. Nevertheless, one could do further tuning of the methodology by using different evidence functions and proposing different regularization approaches or resort to other uncertainty quantification techniques [68]. Also, a dynamic approach to compute the confidence scores for B&B nodes can 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 problem/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 can make the best use of the proposed methodology to solve CO problems at scale. Last but not least, our methodology can be extended to solve MILPs. [1] and [19] present different approaches to how the proposed GNNs can be generalized to predict values of non-binary integer variables. Nevertheless, these approaches need further development for scalability, considering that an arbitrary MILP can include well-complicated 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 developing more complex GNN architectures could be required.

8.2 Broad Impact

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

This paper presents an end-to-end learning method with the goal of acquiring

scalable and cost-efficient GNN-augmented solvers for commonly studied CO problems. Considering the inherent scalability issues with CO problems, our proposed method demonstrates significant potential both in improved scalability and solution quality. Our findings show a potential path to achieve superior GNN-based solvers. Just like the breakthroughs achieved by large pre-trained deep learning models as general-purpose and backbone architectures for intelligent systems, embedding pre-trained GNNs into fully-fledged MIP solvers could also boost solving CO problems in a variety of fields, such as applied mathematics, operations research, computer science, and economics. Moreover, next-generation MIP solvers could be empowered with the efficiency of matrix-based computations on GPUs [13, 1], while the current MIP solvers depend fully on CPUs and RAM. Thus, the time cost of MIP subroutines could be eventually reduced, which could lead to the widespread use of large-scale optimization applications, notably for real-time decision domains.

Another issue worth considering is that the maintenance cost of ML-augmented real-world applications for optimization tasks should not be demanding. However, training deep GNNs requires high GPU memory as their computational complexity linearly scales by the number of nodes and edges in graphs. In addition, a training dataset consisting of large-scale CO problem instances generally requires a relatively higher number of model parameters. Our study devising GNNs learning the optimal solutions to the downscaled instances represents a way of managing both CPU and GPU workloads efficiently for CO without compromising the solution quality. It is also advantageous for adaptive optimization systems 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. Our uncertainty-based approach to utilize decision variable predictions intrinsically works with the sense of trust toward the trained models [69]. Nevertheless, one could refer to emerging research on GNNs developing further interpretable methods for different aspects of explainability like robustness, accountability, and fairness [70] (see [71] on a systematic survey of methods for GNN explainability). 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.

APPENDIX A

EXAMPLE INPUT & OUTPUT DATA

An instance of Combinatorial Auction (CA) Problem (see Section 5.1 for its definition) is given in the following binary linear program defined in Equation 21. This CA instance is represented as the bipartite graph in Figure 13, which depicts the feature vectors for the decision variable and the constraint nodes and corresponding edges with feature values of 1. For this given input graph, Table 7 includes the model output for each decision variable in Equation 21. The predicted values are the decision classes with the highest model outputs. The class probabilities are obtained by passing the outputs of the last hidden layer in the model through softmax function. The class evidence values are calculated from the model outputs as described in Section 4.2. The confidence scores are calculated as described in Section 4.2.2.

$$
\begin{aligned}
\text{max.} \quad & 88.014x_0 + 13.286x_1 + 38.256x_2 + 94.777x_3 + 59.176x_4 \\
& + 281.99x_5 + 83.878x_6 + 93.47x7 + 251.224x_8 + 29.757x_9 \\
\text{s.t.} \quad & x_0 + x_4 + x_5 + x_8 \leq 1 \\
& x_3 + x_4 + x_5 + x_6 + x_8 + x_9 \leq 1 \\
& x_1 + x_3 + x_4 + x_5 + x_7 + x_8 \leq 1 \\
& x_2 + x_3 + x_5 + x_8 + x_9 \leq 1 \\
& x_5 + x_6 + x_7 + x_8 \leq 1 \\
& x_j \in \{0, 1\}, \quad \forall j \in \{0, 1, \ldots, 9\}
\end{aligned}
\tag{21}
$$

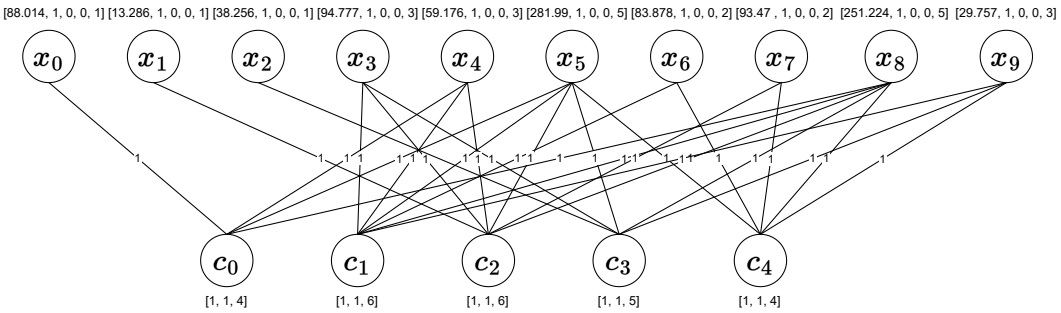


Figure 13: The Bipartite Graph Representation for CA Instance 21

Table 7: The Output of EC+V+EDL model (trained on CA instances with 10 variables) for the given CA instance representation in Figure 13.

Decision Variable	Predicted Value	Class Probabilities	Class Evidence	Confidence Values
x_0	0	[1, 0]	[8.25, 0.02]	[0.892, 0.016]
x_1	0	[1, 0]	[10.52 ,0.01]	[0.913, 0.007]
x_2	1	[1, 0]	[9.96, 0.01]	[0.909, 0.009]
x_3	0	[1, 0]	[7.98, 0.02]	[0.889, 0.018]
x_4	1	[1, 0]	[9.6, 0.01]	[0.906, 0.01]
x_5	0	[0.003, 0.997]	[0.01, 1.24]	[0.008, 0.554]
x_6	1	[1, 0]	[8.78, 0.01]	[0.898, 0.013]
x_7	0	[1, 0]	[8.24, 0.02]	[0.892, 0.017]
x_8	0	[0.006, 0.994]	[0.01, 1.13]	[0.013, 0.531]
x_9	1	[1, 0]	[10.34, 0.01]	[0.912, 0.008]

APPENDIX B

BENCHMARKING DATASETS

This paper studies a set of well-known CO problems consisting of *Minimum Set Cover* (MSC) [72], *Combinatorial Auction* [73], *Maximal Independent Set* [74], *Fixed-Charge Multicommodity Network Flow* [63], and *Generalized Independent Set* [64]. In this section, we provide semi-formal definitions of these CO problems. Then, we detail training data generation procedures for these problems.

MSC dataset released by [15] includes instances with the objective coefficients that are randomly distributed in $[0, 100)$. Due to the range and the type of distribution, 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 by CPLEX.

A fourth dataset used by [15] 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). Since much larger instances of this dataset can be solved by CPLEX within a minute, we did not include it in our experimentation.

MSC, CA, and MIS datasets contain instances that can be solved in a few seconds with CPLEX. Therefore, we generate training, validation, and testing instances at larger scales than those in [15] and other related studies [21, 20]. We generated instances for these problems by setting different random state seeds for each of the training, validation, testing, and transfer datasets. The problem scale of instances in the training, validation, and testing datasets is denoted by 1x based on the number

of decision variables presented in Table 8. The number of decision variables in the transfer instances of each problem is certain multiples of the values in the first row of the table. The scale of transfer instances is denoted accordingly. The number of constraints in the transfer instances is in the same proportion in the table. For example, each MSC-16x instance has 16000 decision variables and 8000 constraints.

To obtain downscaled instances of FCMNF dataset [19] 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 [19]. 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 the easiest and the most challenging datasets, respectively, according to their optimality gap results in [19]. 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 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.

Table 8: The median and standard deviation of the number of variables, the number of constraints, and CPLEX solve time (with a single thread) for the training dataset 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 9: The average and standard deviation of the inference time, which is the sum of elapsed time in the data preprocessing and prediction, for each dataset of each problem.

	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	-	-

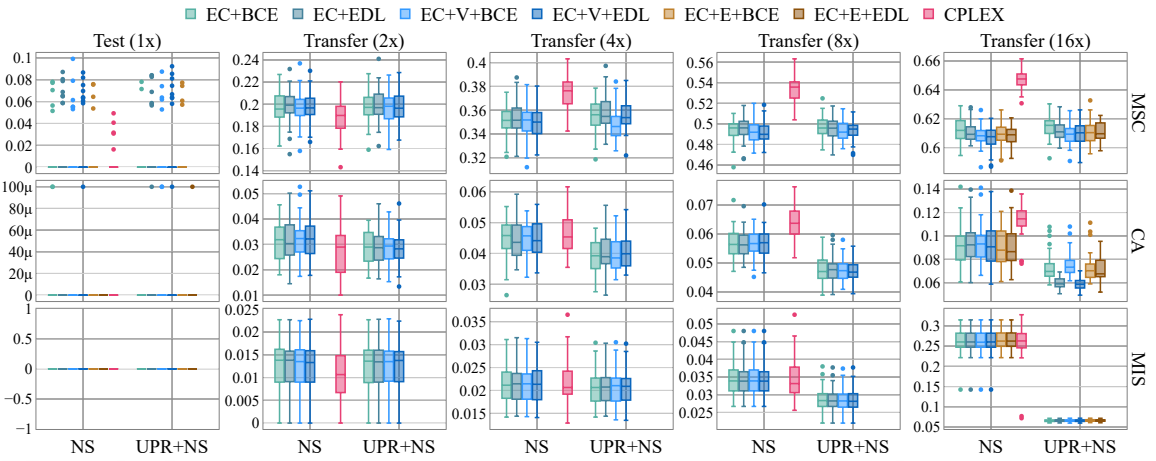
APPENDIX C

ALL RESULTS

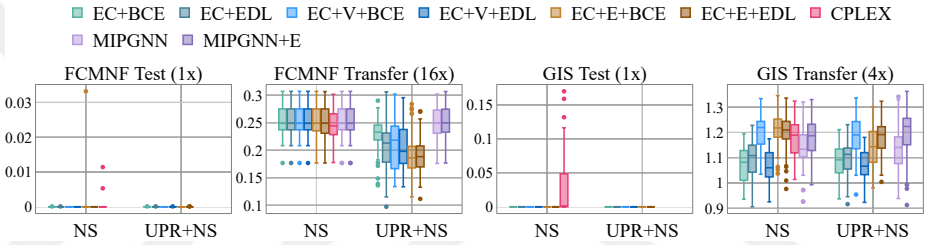
Figure 14, 15, and 16 present boxplots of the optimality gap, the primal gap, and the primal integral results for EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models with NS and UPR+NS search strategies. In the figures, rows correspond to the results for MSC, CA, MIS, FCMNF, and GIS datasets, respectively, and columns correspond to the results for the datasets with 1x, 2x, 4x, 8x, and 16x problem scales, respectively. FCMNF and GIS rows include two subfigures as we evaluate our models over the datasets of the downscaled problems (1x) and the actual problems in [19] (4x and 2x, respectively). Each boxplot represents the distribution of a corresponding metric for 50 instances. We conducted the experiments on the CO pipelines that use EC+E for only the testing and the largest-scale transfer dataset of each problem. So, the results of those pipelines are not available in the figures and the tables for 2x, 4x, and 8x transfer datasets of MSC, CA, and MIS problems.

C.1 Calculating the Time Efficiency of the Proposed Methodology

Let us assume that for a given MSC-2x instance, the NS pipelines deploying EC+BCE, EC+EDL, EC+V+BCE, and EC+V+EDL models achieve their individual best objective values (lower better) 435, 430, 420, and 415 in 1, 2, 3, and 4 minutes, respectively, and then cannot find a better solution in the remaining solve time out of 30 minutes. Let us 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. Another possible case is that CPLEX cannot find a solution having the objective value of 400 in the given time limit (one hour), so



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

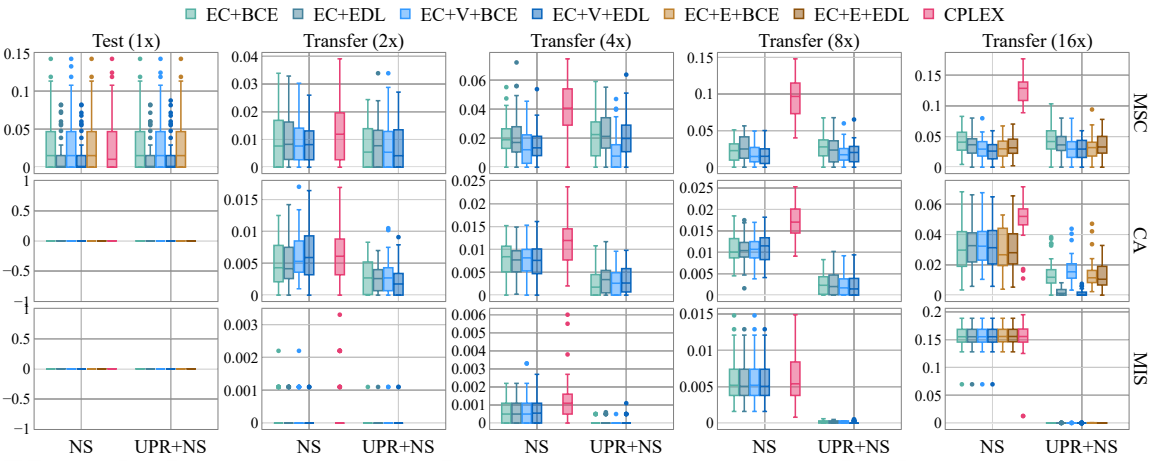


(b) Optimality Gap Values for FCMNF and GIS Datasets

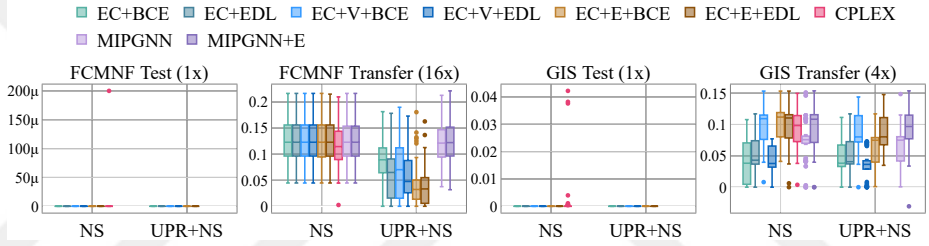
Figure 14: The Boxplot Distributions of Optimality Gap Values for MSC, CA, MIS, FCMNF, and GIS Datasets

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. The following tables include the average elapsed time values by the CO pipelines and CPLEX based on the prescribed calculation. Since there are only two datasets of FCMNF and GIS problems, the cells in the tables for the remaining problem scales are filled by '-' for these problems.

Table 10 presents the average elapsed time (in minutes) of the CO pipelines to obtain individual best objective values for all datasets. The values are averaged across



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

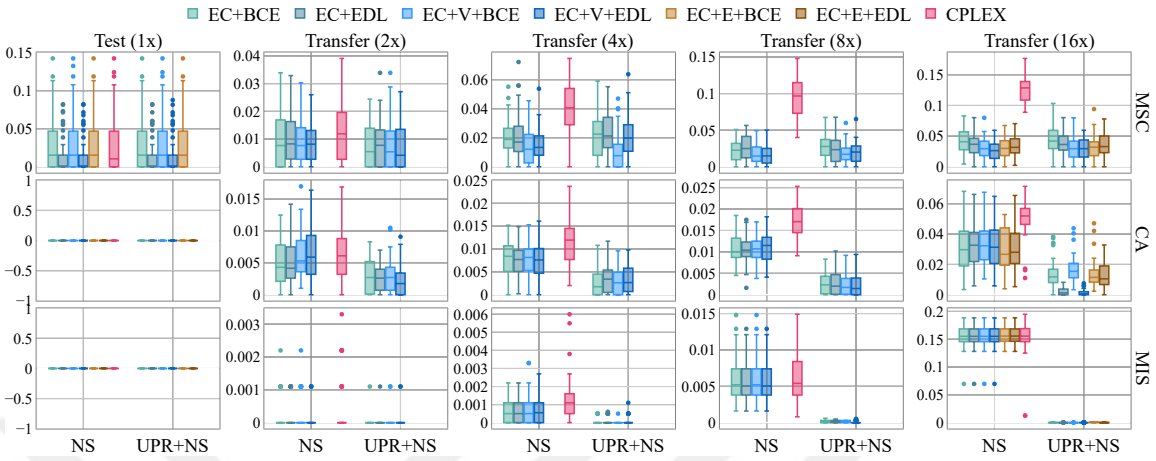


(b) Primal Gap Values for FCMNF and GIS Datasets

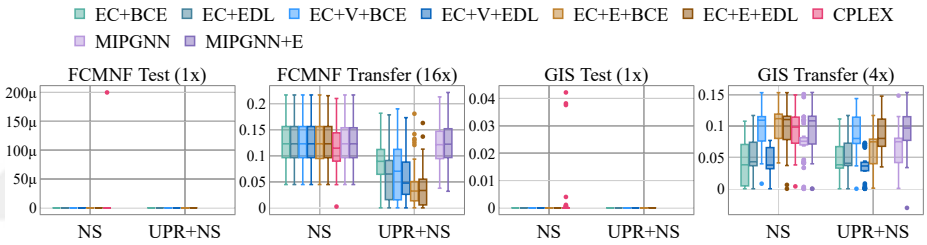
Figure 15: The Boxplot Distributions of Primal Gap Values for MSC, CA, MIS, FCMNF, and GIS Datasets

the models and 50 instances of each dataset. For example, 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 12.79 minutes on average.

Table 11 includes the average elapsed time values with CPLEX runs (for 0.5, 1, 2, 4, and 8 hours per instance at the scales of 1x, 2x, 4x, 8x, and 16x respectively). The values are averaged across the models and 50 instances of each dataset. For example, CPLEX spends 352.32 minutes on average to reach the individual best objective values by the NS pipelines for MSC-16x instances. 'NA' denotes that CPLEX cannot reach the best objective values in the given time limit for each scale of all problems.



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

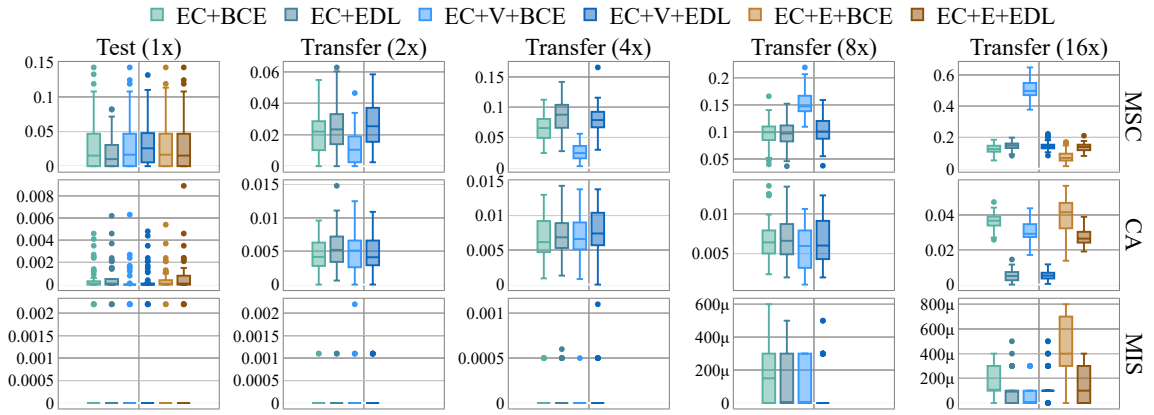


(b) Primal Integral Values for FCMNF and GIS Datasets

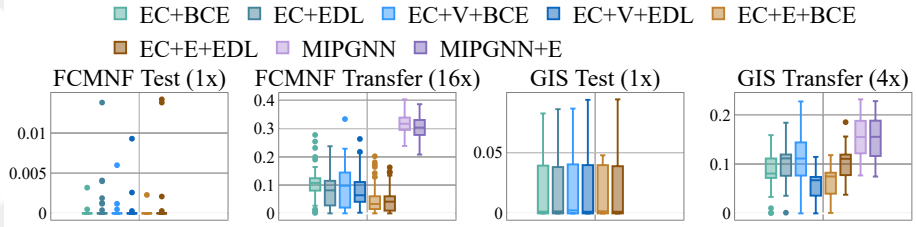
Figure 16: The Boxplot Distributions of Primal Integral Values for MSC, CA, MIS, FCMNF, and GIS Datasets

Table 10: The average elapsed time (in minutes) of the CO pipelines that use EC+BCE, EC+EDL, EDL+V+BCE, and EC+V+EDL models to obtain individual best objective values for each scale of all problems.

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	-	-

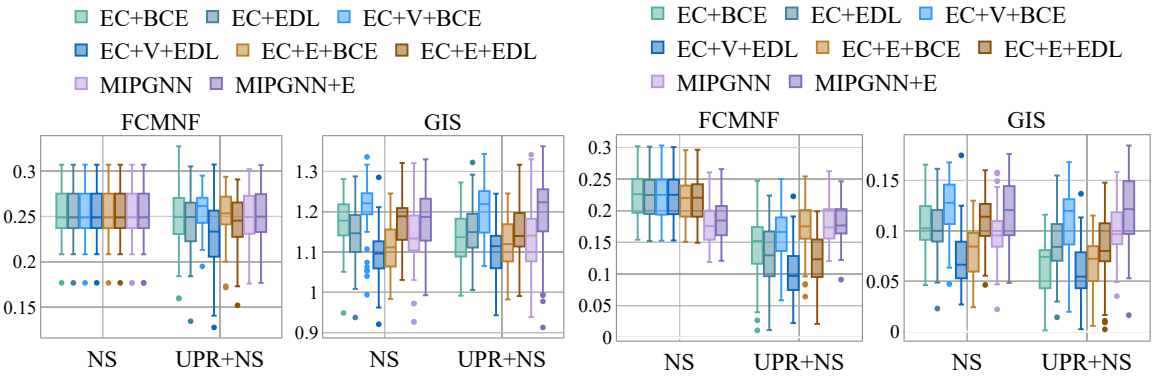


(a) Warm-start Primal Gap Values for MSC, CA, and MIS Datasets



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

Figure 17: The Boxplot Distributions of Primal Gap Values of Warm-start Solutions through UPR algorithm for MSC, CA, MIS, FCMNF, and GIS Datasets



(a) Optimality Gap Values

(b) Primal Integral Values

Figure 18: The Boxplot Distributions Optimality Gap and Primal Integral Values by the Models with 4-layered GNN and *mean* aggregation, and MIP-GNN models for FCMNF-16x and GIS-4x Transfer Datasets

Table 11: The average elapsed time (in minutes) of CPLEX to obtain the best objective values achieved by the CO pipelines that use EC+BCE, EC+EDL, EDL+V+BCE, and EC+V+EDL models for each scale of all problems.

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	-	-

REFERENCES

[1] V. Nair *et al.*, "Solving mixed integer programs using neural networks," *CoRR*, vol. abs/2012.13349, 2020.

[2] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th ed., 2012.

[3] G. Desaulniers, J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis, "Daily aircraft routing and scheduling," *Management Science*, vol. 43, no. 6, pp. 841–855, 1997.

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

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

[6] K. Derinkuyu, F. Tanrisever, N. Kurt, and G. Ceyhan, "Optimizing day-ahead electricity market prices: Increasing the total surplus for energy exchange istanbul," *Manufacturing & Service Operations Management*, vol. 22, no. 4, pp. 700–716, 2020.

[7] B. Wilder, E. Ewing, B. Dilkina, and M. Tambe, "End to end learning and optimization on graphs," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 4674–4685, 2019.

[8] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

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

[10] T. Achterberg and R. Wunderling, *Mixed Integer Programming: Analyzing 12 Years of Progress*, pp. 449–481. Springer Berlin Heidelberg, 2013.

[11] K. Bestuzheva *et al.*, "Enabling research through the scip optimization suite 8.0," *ACM Transactions on Mathematical Software*, 2023.

[12] Z. Chen, J. Liu, X. Wang, and W. Yin, "On representing linear programs by graph neural networks," in *The Eleventh International Conference on Learning Representations*, 2023.

[13] P. Gupta, M. Gasse, E. B. Khalil, M. P. Kumar, A. Lodi, and Y. Bengio, "Hybrid models for learning to branch," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, (Red Hook, NY, USA), Curran Associates Inc., 2020.

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

[15] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds.), pp. 15554–15566, 2019.

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

[17] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent, "Learning TSP Requires Rethinking Generalization," in *27th International Conference on Principles and Practice of Constraint Programming* (L. D. Michel, ed.), vol. 210 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 33:1–33:21, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

[18] Y. Peng, B. Choi, and J. Xu, "Graph learning for combinatorial optimization: A survey of state-of-the-art," *Data Science and Engineering*, vol. 6, pp. 119–141, 2021.

[19] E. Khalil, C. Morris, and A. Lodi, "Mip-gnn: A data-driven framework for guiding combinatorial solvers," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 10219–10227, 2022.

[20] Q. Han, L. Yang, Q. Chen, X. Zhou, D. Zhang, A. Wang, R. Sun, and X. Luo, "A gnn-guided predict-and-search framework for mixed-integer linear programming," in *International Conference on Machine Learning*, 2023.

[21] J. Ding, C. Zhang, L. Shen, S. Li, B. Wang, Y. Xu, and L. Song, "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, 2020.

[22] L. Huang, X. Chen, W. Huo, J. Wang, F. Zhang, B. Bai, and L. Shi, "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, 2022.

[23] Y. Shen, Y. Sun, A. Eberhard, and X. Li, "Learning primal heuristics for mixed integer programs," in *International Joint Conference on Neural Networks*, pp. 1–8, IEEE, 2021.

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

[25] S. Liu, Y. Zhang, K. Tang, and X. Yao, "How good is neural combinatorial optimization?," *CoRR*, vol. abs/2209.10913, 2022.

[26] T. Achterberg, T. Berthold, T. Koch, and K. Wolter, "Constraint integer programming: A new approach to integrate cp and mip," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (L. Perron and M. A. Trick, eds.), (Berlin, Heidelberg), pp. 6–20, Springer Berlin Heidelberg, 2008.

[27] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, pp. 61–80, 2009.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, Toulon, France, Conference Track Proceedings*, OpenReview.net, 2017.

[29] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "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, 2017.

[30] J. Zhang, C. Liu, X. Li, H.-L. Zhen, M. Yuan, Y. Li, and J. Yan, "A survey for solving mixed integer programming via machine learning," *Neurocomputing*, vol. 519, pp. 205–217, 2023.

[31] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[32] P. Gupta, E. B. Khalil, D. Chetélat, M. Gasse, Y. Bengio, A. Lodi, and M. P. Kumar, "Lookback for learning to branch," *Transactions of Machine Learning Research (TMLR)*, 2022.

[33] H. He, H. Daumé, and J. Eisner, "Learning to search in branch-and-bound algorithms," in *NIPS'14*, (Cambridge, MA, USA), p. 3293–3301, MIT Press, 2014.

[34] K. Yilmaz and N. Yorke-Smith, "A study of learning search approximation in mixed integer branch and bound: Node selection in scip," *AI*, vol. 2, no. 2, pp. 150–178, 2021.

[35] A. G. Labassi, D. Chételat, and A. Lodi, "Learning to compare nodes in branch and bound with graph neural networks," in *NeurIPS*, 2022.

[36] M. Balcan, S. Prasad, T. Sandholm, and E. Vitercik, "Structural analysis of branch-and-cut and the learnability of gomory mixed integer cuts," in *NeurIPS*, 2022.

[37] Z. Huang, K. Wang, F. Liu, H.-L. Zhen, W. Zhang, M. Yuan, J. Hao, Y. Yu, and J. Wang, "Learning to select cuts for efficient mixed-integer programming," *Pattern Recognition*, vol. 123, p. 108353, 2022.

[38] J. Song, R. Lanka, Y. Yue, and B. Dilkina, "A general large neighborhood search framework for solving integer linear programs," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, (Red Hook, NY, USA), Curran Associates Inc., 2020.

[39] Y. Wu, W. Song, Z. Cao, and J. Zhang, "Learning large neighborhood search policy for integer programming," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, virtual* (M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds.), pp. 30075–30087, 2021.

[40] D. Liu, M. Fischetti, and A. Lodi, "Learning to search in local branching," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, Virtual Event*, pp. 3796–3803, AAAI Press, 2022.

[41] J. Kotary, F. Fioretto, P. V. Hentenryck, and B. Wilder, "End-to-end constrained optimization learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Virtual Event / Montreal, Canada* (Z. Zhou, ed.), pp. 4475–4482, ijcai.org, 2021.

[42] M. Böther, O. Kißig, M. Taraz, S. Cohen, K. Seidel, and T. Friedrich, "What's wrong with deep learning in tree search for combinatorial optimization," in *The Tenth International Conference on Learning Representations, Virtual Event*, OpenReview.net, 2022.

[43] B. Li, G. Wu, Y. He, M. Fan, and W. Pedrycz, "An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, pp. 1115–1138, 2021.

[44] W. Liao, B. Bak-Jensen, J. Pillai, Y. Wang, and Y. Wang, "A review of graph neural networks and their applications in power systems," *Journal of Modern Power Systems and Clean Energy*, vol. 10, pp. 345–360, 2022.

[45] B. Donon, B. Donnot, I. Guyon, and A. Marot, "Graph neural solver for power systems," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2019.

[46] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning," *CoRR*, vol. abs/1911.04936, 2019.

[47] Z. Fu, K. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large TSP instances," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, Virtual Event*, pp. 7474–7482, AAAI Press, 2021.

[48] E. B. Khalil, B. Dilkina, G. L. Nemhauser, S. Ahmed, and Y. Shao, "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, 2017.

[49] T. Berthold, "A computational study of primal heuristics inside an mi(nl)p solver," *J. of Global Optimization*, vol. 70, no. 1, p. 189–206, 2018.

[50] IBM, "CPLEX User's Manual, Version 22.1.0," 2022.

[51] M. Karimi-Mamaghan, M. Mohammadi, P. Meyer, A. M. Karimi-Mamaghan, and E.-G. Talbi, "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art," *European Journal of Operational Research*, vol. 296, no. 2, pp. 393–422, 2022.

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

[53] A. Gleixner *et al.*, "MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library," *Mathematical Programming Computation*, vol. 13, no. 1, 2021.

[54] J. Kotary, F. Fioretto, and P. V. Hentenryck, "Learning hard optimization problems: A data generation perspective," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, virtual* (M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds.), pp. 24981–24992, 2021.

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

[56] T. Cai, S. Luo, K. Xu, D. He, T. Liu, and L. Wang, "Graphnorm: A principled approach to accelerating graph neural network training," in *Proceedings of the 38th International Conference on Machine Learning, Virtual Event* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 1204–1215, PMLR, 2021.

[57] G. Corso, L. Cavalleri, D. Beaini, P. Liò, and P. Velickovic, "Principal neighbourhood aggregation for graph nets," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[58] P. L. Donti, D. Rolnick, and J. Z. Kolter, "DC3: A learning method for optimization with hard constraints," in *9th International Conference on Learning Representations, Virtual Event, Austria*, OpenReview.net, 2021.

[59] T. Zhao, X. Pan, M. Chen, and S. Low, "Ensuring DNN solution feasibility for optimization problems with linear constraints," in *The Eleventh International Conference on Learning Representations*, 2023.

[60] R. M. Neal, *Bayesian learning for neural networks*, vol. 118. Springer Science & Business Media, 2012.

[61] Y. Gal and Z. Ghahramani, "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, 2016.

[62] T. Achterberg, "Conflict analysis in mixed integer programming," *Discrete Optimization*, vol. 4, no. 1, pp. 4–20, 2007. Mixed Integer Programming.

[63] M. Hewitt, G. L. Nemhauser, and M. W. P. Savelsbergh, "Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem," *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 314–325, 2010.

[64] M. Colombi, R. Mansini, and M. Savelsbergh, "The generalized independent set problem: Polyhedral analysis and solution approaches," *European Journal of Operational Research*, vol. 260, no. 1, pp. 41–55, 2017.

[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

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

[67] T. Berthold, "Measuring the impact of primal heuristics," *Operations Research Letters*, vol. 41, no. 6, pp. 611–614, 2013.

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

[69] A. P. Soleimany, A. Amini, S. Goldman, D. Rus, S. N. Bhatia, and C. W. Coley, "Evidential deep learning for guided molecular property prediction and discovery," *ACS Central Science*, vol. 7, no. 8, pp. 1356–1367, 2021.

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

[71] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, pp. 5782–5799, 2020.

[72] E. Balas and A. Ho, *Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study*. Springer, 1980.

[73] K. Leyton-Brown, M. Pearson, and Y. Shoham, "Towards a universal test suite for combinatorial auction algorithms," in *Proceedings of the 2nd ACM Conference on Electronic Commerce, Minneapolis, MN, USA* (A. Jhingran, J. MacKie-Mason, and D. J. Tygar, eds.), pp. 66–76, ACM, 2000.

[74] R. M. Karp and A. Wigderson, "A fast parallel algorithm for the maximal independent set problem," *Journal of the ACM (JACM)*, vol. 32, no. 4, pp. 762–773, 1985.

VITA

Furkan Cantürk received his bachelor's degree from Industrial Engineering Department at Özyeğin University in 2020. He has been working as a Research Assistant at Özyeğin University AI Research Center Interactive Intelligence Systems Lab and Teaching Assistant at the Department of Computer Science under the supervision of Asst. Prof. Dr. Reyhan Aydoğan since September 2020. He carried out a team project on disaster relief management with unmanned vehicles for TEKNOFEST 2021 Heterogeneous Swarm Simulation Competition. He worked on an international research project on explainable AI funded by TÜBİTAK from August 2021 to August 2022. His research studies are on machine learning for optimization, explainable AI, swarm intelligence, and Multi-Agent Path Finding.