



Network Interdiction Goes Neural

Lei Zhang*
zhanglei@niu.edu
Northern Illinois University
DeKalb, IL, USA

Chang-Tien Lu
ctl@vt.edu
Virginia Tech
Alexandria, VA, USA

Zhiqian Chen
zchen@cse.msstate.edu
Mississippi State University
Starkville, MS, USA

Liang Zhao
liang.zhao@emory.edu
Emory University
Atlanta, GA, USA

Abstract

Network interdiction problems, arising in critical applications from military strategy to disease control, involve a complex attacker-defender dynamic: one player optimizes a network-based objective, while the other strategically modifies the network to impede that objective. The inherent bi-level optimization and combinatorial nature of these problems pose a significant computational challenge, often rendering traditional exact solvers impractical and hindering the development of effective heuristics. While Graph Neural Networks (GNNs) have demonstrated promise in solving single-level combinatorial optimization problems on graphs, their direct application to bi-level interdiction problems remains limited. In this paper, we bridge this gap by introducing a novel approach that leverages the power of GNNs to learn Mixed-Integer Linear Programming (MILP) formulations of network interdiction problems. By representing the problem in this structured mathematical form, we empower a multipartite GNN with the representational capacity to effectively capture the complex interplay between the two players. This approach aligns the neural network with the underlying mathematical structure of interdiction problems, leading to improved performance. Through extensive experiments on two network interdiction tasks, we demonstrate the superiority of our proposed method over both baseline GNN models and traditional exact solvers, showcasing its potential for real-world applications.

CCS Concepts

• **Computing methodologies** → **Neural networks**; *Modeling methodologies*; **Neural networks**; • **Theory of computation** → *Shortest paths*.

Keywords

Graph Neural Network, Network Interdiction, Combinatorial Optimization

ACM Reference Format:

Lei Zhang, Zhiqian Chen, Chang-Tien Lu, and Liang Zhao. 2025. Network Interdiction Goes Neural. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25)*, August 3–7, 2025.



This work is licensed under a Creative Commons Attribution 4.0 International License. *KDD '25, Toronto, ON, Canada*

© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1454-2/2025/08
<https://doi.org/10.1145/3711896.3737063>

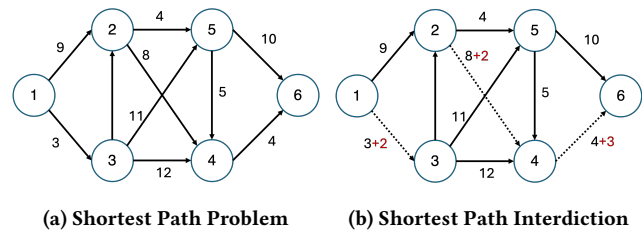


Figure 1: Illustration of the Shortest Path (left) and Shortest Path Interdiction (right) problems. The interdiction problem involves strategically increasing edge costs within a limited budget to maximize the shortest path length.

Toronto, ON, Canada. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3711896.3737063>

KDD Availability Link:

The source code of this paper has been made publicly available at <https://doi.org/10.5281/zenodo.15587006>.

1 Introduction

Graph Neural Networks (GNNs) have emerged as a powerful tool for tackling Combinatorial Optimization (CO) problems, leveraging the natural representation of many CO problems as graphs and the prevalence of network data in real-world applications. GNNs have achieved notable success in diverse CO tasks, including the Traveling Salesman Problem (TSP) [34], graph matching [17, 29], and graph edit distance [3]. Their inductive bias, particularly permutation invariance and sensitivity to input sparsity, makes them well-suited for encoding graph-structured data.

However, beyond the realm of CO problems where the optimization is unilateral, real-world scenarios often involve adversarial settings requiring *two* levels of optimization. Consider, for example, a power grid: optimizing network throughput (single level) is important, but equally crucial is identifying vulnerabilities where targeted network modifications could inflict maximum damage (bi-level). This exemplifies a more general class of CO problems known as *network interdiction*, where a *follower* aims to solve a CO problem on a fixed network, while a *leader* strategically modifies the network to hinder the follower's objective. Figure 1 contrasts a standard combinatorial optimization (CO) problem, a shortest path

problem, with a corresponding network interdiction problem, illustrating the latter’s bi-level structure where an inner CO problem is embedded within the interdiction layer.

Solving network interdiction problems presents a significant computational challenge. Traditional approaches fall into two categories: exact solvers and heuristic solvers [38]. Exact solvers aim to identify optimal solutions, necessitating the resolution of complex mathematical problems, typically NP-hard, which involves exploring a combinatorial space of potential network modifications and assessing their impact on the follower’s solution [12, 44]. As the network size increases, the exponential growth in the number of potential modifications results in computational complexity that exceeds polynomial time bounds. Conversely, heuristic solvers are algorithms designed to efficiently find good, if not optimal, solutions more quickly than exact methods. However, due to the complexity and variability of these problems—such as diverse assumptions, a vast solution space, and inherent problem intricacies—it’s uncommon for a pure heuristic solver to be effective. Instead, combining heuristic modules with traditional algorithms offers greater promise [38]. GNN-based machine learning methods, which have shown effectiveness in conventional single-level CO problems [8], are clearly strong candidates for serving as heuristic modules in solving network interdiction problems more efficiently.

Despite this potential, the application of GNNs to network interdiction remains underexplored due to critical challenges. *Firstly, there is no effective representation method for network interdiction instances.* For a GNN to learn to solve network interdiction problems, it must be able to encode all relevant information, including that of both the leader and the follower, in an attributed graph, and distinguish between instances with different solutions. *Secondly, there is no theoretical guarantee that GNNs can be trained to perform algorithmically and generalize in solving network interdiction problems.* Recent studies have shown that GNNs excel in certain CO tasks because they “align” with dynamic programming (DP) [46]. Given that DP or analogous polynomial-time heuristics can solve numerous CO problems, GNNs offer potential for generalization and extrapolation in these areas [16]. Unfortunately, *network interdiction problems do not align with DP*, making previous GNN for CO unable to be trivially used for network interdiction.

This paper addresses these challenges by proposing a novel computational framework for network interdiction. Specifically, we investigate two key questions: 1) *Can GNNs provide effective representation for network interdiction problems, and if so, what theoretical assurances exist?* 2) *If the alignment between DP and GNN does not justify the utilization of GNNs for network interdiction problems, what alternative rationale does?* To answer these questions, we leverage research on GNN expressiveness and tailor our approach to the specific characteristics of network interdiction and its traditional solution methodologies. Our key contributions include:

- We establish, with supporting mathematical proofs, that GNNs can approximate key properties of network interdiction problems, providing a theoretical foundation for their application in this domain.
- We demonstrate that network interdiction problems can be formulated as special cases of MILPs and introduce MMILP-GNN, a novel GNN architecture designed for the multipartite

graph structure of these network interdiction instances, enabling effective learning on these representations.

- Through extensive experiments on two representative network interdiction problems, we demonstrate the superior performance of MMILP-GNN compared to SCIP, one of the fastest non-commercial solvers for MILPs.

The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 provides background on network interdiction and traditional solvers; Section 4 details the proposed framework and MMILP-GNN architecture; Section 5 presents theoretical analysis and proofs; Section 6 reports and analyzes experimental results; Section 7 discusses limitations; and Section 8 concludes and outlines future directions.

2 Related Work

This work resides at the intersection of two active research areas: network interdiction and learning-based combinatorial optimization solvers. We briefly review the most relevant literature in each field.

Network Interdiction. Network interdiction problems, which involve strategically disrupting a network to impede its functionality, have been studied extensively across various domains. Early work focused on classical optimization approaches, formulating interdiction problems as MILPs and employing techniques like branch-and-bound and cutting planes [24, 44]. These methods, while capable of finding optimal solutions, often suffer from scalability issues due to the inherent NP-hardness of many interdiction problems. Researchers have thus explored approximation algorithms and heuristics to address larger instances [4]. Examples include greedy algorithms, Lagrangian relaxation, and simulated annealing [13, 31]. Despite the development of various approximation algorithms and heuristics, formulating and solving network interdiction problems as MILPs remains the dominant approach due to the ability of MILP solvers to guarantee optimality (for reasonably sized instances) and the flexibility of the MILP framework. However, the inherent NP-hardness of these problems leads to a significant computational burden for large-scale instances, limiting the practical size of solvable problems. Powerful MILP solvers like Gurobi [19] and SCIP [6], incorporating techniques like branch-and-cut and cutting planes, are essential tools for tackling this challenge. Our work aims to improve the scalability of these solvers by introducing a learning-based guidance approach.

Learning-based Combinatorial Optimization. The success of large language models in various reasoning tasks [1, 14] has fueled growing interest in learning-based approaches for solving CO problems [5, 8]. Early work focused on end-to-end learning frameworks where neural networks directly predict solutions to CO problems [27, 41]. These methods, often leveraging GNNs [20, 28], have shown promise in learning problem-specific representations and generating reasonable solutions. For example, GNNs have been applied to the Traveling Salesperson Problem [25] and the Maximum Independent Set problem [2, 33]. However, these end-to-end approaches frequently lack theoretical guarantees regarding solution quality and optimality.

To address these limitations, hybrid approaches have emerged, combining traditional optimization methods with machine learning to leverage the strengths of both paradigms. Machine learning has been used to guide branch-and-bound solvers [18], enhance cutting-plane methods [39, 43], and improve holistic integration with optimization pipelines [23, 42]. Additionally, GNNs have been integrated with search strategies like beam search [26] and tree search [30], resulting in notable improvements in solving CO problems. Complementary to these advances, researchers have investigated the expressive power of GNNs for CO problems [7, 16, 46], offering deeper insights into their representational limits and mechanisms for capturing combinatorial structures.

Our work focuses on network interdiction, a special class of CO problems framed as bi-level optimization, which has not been studied in the learning-based CO literature. Following the pattern of recent hybrid and theory-grounded approaches, we provide the first formal guarantee that neural networks can represent and solve these problems. Furthermore, we introduce a novel neural network architecture specifically tailored to network interdiction, addressing its unique characteristics and offering a new pathway for scalable, learning-based solutions.

3 Preliminaries

A network interdiction problem typically includes two players engaging in a game of max-min or min-max on a defined graph. One player, commonly referred to as the follower or defender, aims to optimize a standard CO problem on the graph, such as identifying the shortest path or maximizing the maximum flow between two nodes. The other player, often known as the leader or attacker, manipulates the network on which the follower operates, strategically disrupting the follower’s objective by actions like removing edges that should be connected in the graph or adding costs to existing edges. Formally, we define an instance of a network interdiction problem as follows.

Definition 3.1 (Network Interdiction Problem). The general form of a max-min network interdiction problem is defined as:

$$\max_{\mathbf{x}} \Theta(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in X, \quad (1)$$

where \mathbf{x} represents the leader/attacker’s variable. The objective function of the interdiction $\Theta(\mathbf{x})$ is defined as the minimization of another function:

$$\Theta(\mathbf{x}) = \min_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}) \quad \text{s.t.} \quad \mathbf{y} \in Y(\mathbf{x}), \quad (2)$$

where \mathbf{y} represents the follower/defense variables, $f(\mathbf{x}, \mathbf{y})$ represents the follower/defender’s objective function (affected by the leader/attacker’s action \mathbf{x}), and $Y(\mathbf{x})$ is the set of feasible actions that the follower/defender can do for a given \mathbf{x} .

A min-max network interdiction problem is essentially the reverse of the max-min network interdiction problem, where the outer objective is now minimization, and the inner objective is maximization.

Example 3.2 (Shortest Path Interdiction).

$$\begin{aligned} & \max_{\mathbf{x} \in X} \{ \min_{\mathbf{y} \in Y(\mathbf{x})} \sum_{(i,j) \in A} (c_{i,j} + d_{i,j} x_{i,j}) y_{i,j} \} \\ \text{s.t.} \quad & x \in \{0, 1\}^{|A|} : \sum_{(i,j) \in A} x_{i,j} \leq \gamma, \quad T\mathbf{y} = \mathbf{b}, \quad \mathbf{y} \geq 0. \end{aligned} \quad (3)$$

Eq. (3) illustrates the mathematical formulation of the shortest-path network interdiction problem. In this context, $\mathbf{x} = \{x_{i,j}\}_{(i,j) \in E}$ is a binary decision vector indicating interdicted edges, with E being the network’s edge set. Here, $c_{i,j}$ denotes the length of the edge, $d_{i,j}$ represents the additional length if edge (i, j) is interdicted, $y_{i,j}$ indicates whether the edge exists in the network, and γ is the budget constraint on the number of interdictions.

Traditional exact solvers: Branch-and-Bound (BB) and Benders decomposition are two fundamental traditional exact methods for solving network interdiction problems. Compared to Benders decomposition, BB is more versatile and applicable to a wide range of network interdiction problems [15]. The problems that BB can handle are normally defined in MILPs, which can be formulated as

$$\min_{\mathbf{x} \in \mathbb{R}^n} c^T \mathbf{x}, \quad \text{s.t.}; Ax \circ \mathbf{b}, l \leq x \leq u, x_j \in \mathbb{Z}, \forall j \in I. \quad (4)$$

For general MILPs, the worst-case complexity can be similarly exponential, as the algorithm may need to consider an exponential number of subproblems in the search space. Efforts have been made to represent general LPs and MILPs using a bipartite graph structure and then utilize GNNs to estimate solutions [9, 10, 18], but these methods have not yet been applied to network interdiction problems.

It is important to emphasize that while network interdiction problems can be formulated as general MILPs, solving them is not equivalent to solving general MILPs — particularly when approached as a machine learning task. In this context, reducing the problem to a generic MILP structure overlooks the specific inductive biases inherent to network interdiction. This results in the loss of easily interpretable information for the model. For instance, in Equation (4), all variables are denoted as x , even though the original network interdiction problem may involve multiple variables with distinct meanings, such as decision variables and dual variables.

4 Proposed Framework

This section outlines the process of transforming a network interdiction problem into input suitable for a neural network, along with elucidating the neural network’s architecture. To ensure the model’s generalization ability, in Section 4.1, we process the problem instances into the Mixed Integer Linear Programming (MILP) form that BB can handle, following traditional methods. Given that GNNs have shown effectiveness in learning branching strategies [8], this approach simplifies the reasoning task for the GNN. These MILP formulations are then translated into multipartite graphs, capturing essential characteristics of the scenarios as outlined in Section 4.2. Subsequently, we propose a multipartite graph neural network, MMLP-GNN, tailored for estimating optimal interdiction decisions on the induced multipartite graphs rather than the original competitive network between the leader and follower, as elucidated in Section 4.3. More detailed theoretical rationale is provided in Section 5.

4.1 Preprocess Network Interdiction Instances

As the initial step in solving network interdiction problems, our aim is to process the problem instances using traditional methods up to the point where these methods become inadequate, leaving the challenging part to GNNs. As illustrated in Example 3.2, we have

already transformed the problem instances into a constrained optimization problem. However, this format is not suitable for neural networks due to the nested two levels of optimization. To simplify this, we employ the “dualize-and-combine” approach. First, we derive the dual formulation of the inner problem with a fixed leader decision variable (in Example 3.2’s case, the variable x), ensuring that both players’ problems align in the same optimization direction. Subsequently, we release the leader decision variable as a decision vector, converting the whole problem into a single-level MILP.

It is noteworthy that the network interdiction problems addressed in this paper, represented by Eq. (3), involve a followers’ problem that lends itself to being modeled as a convex optimization problem. This forms the basis for applying the “dualize-and-combine” technique for single-level reduction. Although this imposes a constraint on the types of problems we can tackle, the majority of network interdiction problems commonly encountered in real-world scenarios fall within this category [38].

Take the shortest path interdiction problem in Eq. (3) as an example, following the above-mentioned processes, the constrained bi-level optimization problem in Eq. (3) can be transformed into a single-level optimization:

Example 4.1 (Single-Level Reduction for the Shortest Path Interdiction Problem).

$$\begin{aligned} & \max_{\mathbf{x} \in X} \{ \min_{\mathbf{y} \in Y(\mathbf{x})} \sum_{(i,j) \in A} (c_{i,j} + d_{i,j}x_{i,j})y_{i,j} \} \\ \text{s.t. } & \mathbf{x} \in \{0,1\}^{|A|} : \sum_{(i,j) \in A} x_{i,j} \leq \gamma, \quad T\mathbf{y} = \mathbf{b}, \quad \mathbf{y} \geq 0, \end{aligned} \quad (5)$$

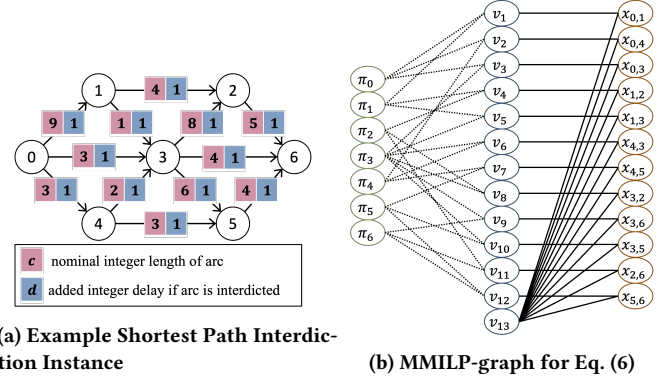
$$\begin{aligned} & \max_{\mathbf{x}, \boldsymbol{\pi}} \mathbf{b}^T \boldsymbol{\pi} \\ \text{s.t. } & T^T \boldsymbol{\pi} \leq \mathbf{c} + D\mathbf{x}, \quad \mathbf{x} \in \{0,1\}^{|E|} : \sum_{(i,j) \in A} x_{i,j} \leq \gamma \end{aligned}$$

4.2 Graph Representations for Network Interdiction Problems

We propose a *Multipartite MILP-induced graph* or *MMILP-graph* representation to encode the above single-level optimization into a form, i.e., MMILP-Graph, that is readable by GNNs while preserving all the needed information.

Definition 4.2 (MMILP-graph). A multipartite MILP-induced graph is defined as a tuple (G, H) , where $G \equiv (W_0 \cup W_1 \cup \dots \cup W_p \cup V, E)$ consists of vertex set $W_0 \cup W_1 \cup \dots \cup W_p \cup V$ and a weighted edge set E , and H represents vertex features.

- **Vertices:** The vertex set can be divided into three groups including one interdiction action variable vertex group W_0 , m dual-variable vertex group, and one constraint vertex group V . Each of the child vertex groups has no overlap with another child vertex group. For convenience, we use V' to represent one of the vertex groups: $V' = (W_0, W_1, \dots, W_p, V)$.
- **Edges:** One edge in E can connect one variable vertex in any of the variable vertex groups with one constraint vertex. Note that there is no edge connecting vertices in the same vertex group, or between two different vertex groups. An edge can carry edge features: $E = E^{W_0, V} \cup E^{W_1, V} \cup \dots \cup E^{W_p, V}$ where $E^{W_k, V}$



(a) Example Shortest Path Interdiction Instance

(b) MMILP-graph for Eq. (6)

Figure 2: Network Interdiction Instance and the Corresponding MMILP-Graph

represents the edges connecting one variable in W_p and one constraint in V .

- **Vertex features:** Each vertex has its corresponding feature vector that represents characteristics in the original interdiction problem.

Finally, an MMILP-graph is defined as $(G, H) \in \mathcal{G}$

Example 4.3 (MMILP-Graph for a Specific Shortest Path Interdiction Instance). Take the shortest path interdiction instance in Fig. 2a as an example, the induced MILP can be found in Eq. (6), and the corresponding MMILP-graph is shown in Fig. 2b.

$$\begin{aligned} & \max(\pi_0 - \pi_9) \\ \text{s.t. } & v_1 : \pi_0 - \pi_1 - x_{0,1} \leq 9, v_2 : \pi_0 - \pi_4 - x_{0,4} \leq 3, \\ & v_3 : \pi_0 - \pi_3 - x_{0,3} \leq 3, v_4 : \pi_1 - \pi_2 - x_{1,2} \leq 4, \\ & v_5 : \pi_1 - \pi_3 - x_{1,3} \leq 1, v_6 : \pi_4 - \pi_3 - x_{4,3} \leq 2, \\ & v_7 : \pi_4 - \pi_5 - x_{4,5} \leq 3, v_8 : \pi_3 - \pi_2 - x_{3,2} \leq 8, \\ & v_9 : \pi_3 - \pi_6 - x_{3,6} \leq 4, v_{10} : \pi_3 - \pi_5 - x_{3,5} \leq 6, \\ & v_{11} : \pi_2 - \pi_6 - x_{2,6} \leq 5, v_{12} : \pi_5 - \pi_6 - x_{5,6} \leq 4, \\ & v_{13} : x_{0,1} + x_{0,3} + x_{0,4} + x_{1,3} + x_{1,2} + x_{2,6} + x_{3,2} + x_{3,6} \\ & \quad + x_{3,5} + x_{4,3} + x_{4,5} + x_{5,6} \leq 1 \end{aligned} \quad (6)$$

Vertices: In in Fig. 2b, the vertices groups include one interdiction action variable group $\mathbf{x} = \{x_{0,1}, x_{0,3}, \dots, x_{5,6}\}$ (the vertices on the right side), only one dual-variable vertex group $\boldsymbol{\pi} = \{\pi_0, \pi_1, \dots, \pi_6\}$ (the vertices on the left side), and one constraint group $\mathbf{v} = \{v_1, v_2, \dots, v_{13}\}$ (the vertices in the middle). In this problem, p in Definition 4.2 equals 1 because there is only one dual variable $\boldsymbol{\pi}$ in the reduced form of MILP.

Edges: One edge in Fig. 2b connects one constraint vertex (in the middle column) and one variable vertex (in the left or right column), representing the relationship between a constraint ($v_i, i = 1, 2, \dots, 13$) and the involved variables in Eq. (6). The weight of the edge represents the coefficient of the variable in the constraint. For example, the weight of the edge between vertex π_0 and v_1 , i.e., $E_{0,1}^{W_0, V}$, is 1 because in the constraint v_1 , i.e., $\pi_0 - \pi_1 - x_{0,1} \leq 9$, the coefficient of π_0 is 1.

Vertex features: The vertex features for both variable vertices and constraint vertices align with the feature definitions in the

bipartite graphs in [9, 10, 18, 21, 32]. $h_i^{V'}$ represents the feature vector for the i -th node of vertex group V' .

Relationship with existing work. This MMILP-graph representation is inspired by the use of bipartite graphs for solving general MILPs, as initially introduced by Gasse et al. [18] and subsequently utilized in various studies ([32], [21], [9], [10]). Although we can demonstrate that the induced single-level MILPs of network interdiction instances in Section 4.1 can be represented in the standard MILP format, the use of MMILP-graph can not only capture the heterogeneity of different variables but also learn their different degrees of importance. Firstly, variables in the induced problems typically are different from each other. Comparing Eq. (5) with a general MILP in Eq. (4), the dual vector π and the interdiction decision vector x represent distinct variables. Incorporating them into the MILP-graph overlooks the difference between x and π , resulting in reduced information. Secondly, different variables have varying degrees of importance. Take shortest-path interdiction as an example, our primary interest lies in the value assignments of variable x because once x is assigned, the problem simplifies to a regular shortest-path problem that can be solved by Dijkstra’s algorithm within polynomial time. This characteristic is also called decomposability which is essential for using Benders Decomposition on network interdiction problems.

4.3 Graph Neural Network for MMILP-Graphs

Given the above-formulated MMILP-graph, here we propose a Multipartite Graph Neural Network that takes it as input to predict the solution network interdiction instance. The graph neural network aims to learn an $\mathcal{G} \rightarrow \mathbb{R}^n$ mapping where n is the number of nodes in the network interdiction problem as well as the number of variables in vertex group W_0 .

In detail, because of the heterogeneity and multipartite structure of the input graph, our graph neural network has several passes. In general, we design the encoding layer for each partite of the vertices; we then propose a new message-passing strategy to learn the mapping between different groups of vertices where they should communicate; and finally, a read-out layer will produce estimates for the interdiction variable vertices, each corresponding directly to one candidate edge in the original network.

Variable and constraint vertices encoding: Raw vertex features are encoded into the embedding space with trainable functions $f_{in}^{V'}$, ($V' = (W_0, W_1, \dots, W_p, V)$):

$$h_i^{0,V'} = f_{in}^{V'}(h_i^{V'}), \quad (7)$$

where $h_i^{0,V'}$ represents the embedding (layer 0) for node i in vertex group V' , and $f_{in}^{V'}$ represents the function for encoding the raw vertex features in group V' into embedding space. In practice, we append random features [36] to the vertex features to enhance the separation power of our method as explained in Section 5.

Variables-constraints message passing: We stack multiple graph neural network layers with variables-to-constraints message passing ($v \rightarrow c$) and constraint-to-variable message passing ($c \rightarrow v$). Since there are multiple sets/groups of variables in the MMILP-graph, the trainable functions are different as well. $v \rightarrow c$ and

$c \rightarrow v$ are defined in turn in Eq (8) and Eq (9):

$$h_i^{l,V} = g_l^V(h_i^{l-1,V}, \sum_{k=0}^p \sum_{e \in E^{W_k,V}} f_l^{W_k}(h_i^{l-1,V}, h_j^{l-1,W_k}, e), \quad (8)$$

$$h_i^{l,W_k} = g_l^{W_k}(h_i^{l-1,W_k}, \sum_{e \in E^{W_k,V}} f_l^{W_k}(h_i^{l-1,V}, h_j^{l-1,W_k}, e), \quad (9)$$

where h_i^{l,W_k} is vertex features for node i within variable set W_k at layer l , and $h_i^{l,V}$ is vertex features for node i in constraint set V at layer l .

Interdiction decision read-out: The read-out function is only applied on the interdiction decision variable vertices, i.e., W_0 as is demonstrated in Eq (10).

$$\hat{x}_i = g_{out}(h_i^{L,W_0}), \quad (10)$$

With the definition of the detailed operations, we denote the collection of GNNs with \mathcal{F} :

$$\mathcal{F} = \{F : \mathcal{G} \rightarrow \mathbb{R}^n | F \text{ yields (7), (8), (9), (10)}\} \quad (11)$$

In practice, we use multi-linear perceptrons (MLPs) for all the trainable functions in Eq (7) - (10).

Since we train the models using optimal solutions as labels, the predicted values can be interpreted as the likelihood of whether an edge should be interdicted or the budget to be allocated to that edge. This is actually more convenient than most unilateral CO problems that involve sequential decision-making. For instance, solutions to the shortest path problem must consist of a sequence of connected edges in the graph from the source to the target, whereas shortest path network interdiction does not have this requirement.

5 Theoretical Foundations and Results

In this section, we develop the theoretical foundation of our proposed approach, providing a rigorous analysis of its effectiveness for network interdiction through formal guarantees and analytical insights.

5.1 Representation Power Analysis

The representation power of a neural network is determined by its ability to produce different outcomes for different inputs, which is fundamental to its effectiveness. Our approach to measuring the representation power of MMILP-GNN is heavily inspired by existing research on the representation power of GNNs [46] and the separation power of GNNs in LP [9] and MILP [10]. Following the extension of the Weisfeiler-Lehman (WL) test for LP-Graphs [9] and MILP-Graphs [10], we further extend it for MMILP-Graphs as shown in Algorithm 1. We extend these results to MMILP-GNNs for network interdiction instances and validate them in a similar manner.

THEOREM 5.1. *For two MMILP-graphs (G, H) and (\hat{G}, \hat{H}) , they cannot be distinguished by WL_{MMILP} if and only if $F(G, H) = F(\hat{G}, \hat{H}), \forall F \in \mathcal{F}$*

The proof of Theorem 5.1 follows from Lemma 5.2 and 5.3. The proof establishes the equivalence between the discriminative power of the WL_{MMILP} test and the functions $F \in \mathcal{F}$ for distinguishing MMILP-graphs (G, H) and (\hat{G}, \hat{H}) . The core argument hinges on an

Algorithm 1 WL_{MMILP}

Require: A graph instance (G, H) with p dual variables, iteration limit $L > 0$.

- 1: $C_i^{0,V} = \text{HASH}_V^0(h_i^V)$
- 2: $C_j^{0,W_k} = \text{HASH}_{W_k}^0(h_j^{W_k})$ for k in $\{0, 1, 2, \dots, p\}$.
- 3: **for** $l = 1, 2, \dots, L$ **do**
- 4: $C_i^{l,V} = \text{HASH}_V^l\left(C_i^{l-1,V}, \sum_{k=0}^p \sum_{j=1}^{n_k} E_{i,j}^{W_k,V} \text{HASH}_{W_k}^{l-1}\left(C_j^{l-1,W_k}\right)\right)$.
- 5: **for** $k = 0, 1, \dots, p$ **do**
- 6: $C_j^{l,W_k} = \text{HASH}_{W_k}^l\left(C_j^{l-1,W_k}, \sum_{i=1}^m E_{i,j}^{W_k,V} \text{HASH}_{V,W_k}^{l-1}\left(C_i^{l-1,V}\right)\right)$.
- 7: **end for**
- 8: **end for**
- 9: **return** The multisets containing all colors $\{\{C_i^{L,V}\}_{i=0}^m$ and $\{\{C_j^{L,W_k}\}_{j=0}^n\}_{k=0}^p$.

inductive analysis of the relationship between the WL_{MMILP} color refinement process and the hidden states of a GNN. Here is the structured overview:

LEMMA 5.2. *Let $(G, H), (\hat{G}, \hat{H}) \in \mathcal{G}$. if $(G, H) \sim (\hat{G}, \hat{H})$ then for any $F_W \in \mathcal{F}_{\text{GNN}}^W$, there exists a permutation $\sigma_W \in S_n$ such that $F_W(G, H) = \sigma_W(F_W(\hat{G}, \hat{H}))$.*

LEMMA 5.3. *Let $(G, H), (\hat{G}, \hat{H}) \in \mathcal{G}$. If $F(G, H) = F(\hat{G}, \hat{H})$ holds for any $F \in \mathcal{F}$, then $(G, H) \sim (\hat{G}, \hat{H})$.*

The proofs are detailed in Appendix A. Since our results align with those in [10] for MILPs, it is possible that the network interdiction instances induce *foldable* MILPs as introduced in their work. To address these cases, we enhance the representation power in our experiments by appending random features [36] to the MMILP-graphs.

5.2 Discussions on Generalization Ability

The concept of algorithmic alignment, introduced by Xu et al. [46], describes the reasoning capabilities of GNNs. The core idea is that a neural network is more effective at learning to perform a reasoning task, such as solving a CO problem, when its architecture aligns well with the algorithm designed to solve that task. Specifically, GNNs have been found to align with DP, a paradigm capable of solving many CO problems. This alignment between DP and GNNs has been further examined by Dudzik and Velickovic [16] using methods from category theory and abstract algebra.

However, network interdiction problems fall into a category that doesn't align with DP and thus cannot be naturally addressed by GNNs. DP relies on the principle of optimal substructure, where an optimal solution to a problem can be constructed from optimal solutions to its subproblems [11]. Network interdiction problems typically involve nested two-layer decision-making, which does not exhibit the overlapping subproblems characteristic. The way our proposed method aligns better with the exact solver can also be explained with Dudzik and Velickovic [16]'s theories in abstract algebra: by representing the network interdiction instances as multipartite graphs, the MMILP-GNN and the target algorithm (BB) at least share the same finite sets in the polynomial spans.

6 Experiments

In this section, we assess the performance of the proposed framework through empirical experiments, which serve to validate the theoretical findings presented in Section 5.

6.1 Datasets

We focus on two representative network interdiction problems: Shortest Path Interdiction (SPI) and Maximum Flow Interdiction (MFI). For both problems, we generated synthetic datasets. Specifically, we created four MFI datasets (MFI20, MFI30, MFI100, and MFI200) and four SPI datasets (SPI20, SPI30, SPI100, and SPI200), each containing 4,000 instances. Each datasets had 20, 30, 100, and 200 nodes, respectively.

To balance the computational cost of data generation with the challenge posed to learning-based models, we tuned the interdiction budget for each dataset. The budget was selected to be large enough to create non-trivial instances, allowing our model to demonstrate its advantages over traditional solvers, but not so large as to make data generation computationally prohibitive. Where applicable, we adopted hyperparameter settings from Schäfer [37].

For SPI, after preprocessing, we solve the problem defined in Eq (5). For each instance, we generate a fully connected directed graph and assign cost and capacity to each edge. The source node is set as the first node, and the sink node is set as the N_v -th node in the network, where N_v is the total number of nodes. The cost on each edge is sampled uniformly between 1.0 and 10.0, and the capacity is sampled uniformly between 20.0 and 70.0. The budget constraint for three synthetic datasets (SPI20, SPI30, and SPI 100) are all set to 15, 15, and 30, respectively.

Similarly, for MFI, we solve the problem in Eq (12). For each instance, we generate a fully connected directed graph and assign cost and capacity to each edge. The source node is set as the first node, and the sink node is set as the N_v -th node in the network, where N_v is the total number of nodes. The cost on each edge is sampled uniformly between 1.0 and 10.0, and the capacity is sampled uniformly between 10.0 and 60.0. The budget constraint for all three synthetic datasets (MFI20, MFI30, and MFI 100) is all set to be 15.

$$\begin{aligned}
 \min_{\alpha, \beta, \gamma} \quad & \sum_{(i,j) \in A} u_{i,j} \beta_{i,j}, \\
 \text{s.t.} \quad & \alpha_i - \alpha_j + \beta_{i,j} + \gamma_{i,j} \geq 0, \quad \forall (i,j) \in A, \quad \alpha_t - \alpha_s \geq 1, \\
 & \sum_{(i,j) \in A} r_{i,j} \gamma_{i,j} \leq R, \quad \alpha_i \in \{0, 1\}, \quad \forall i \in N, \\
 & \beta_{i,j}, \gamma_{i,j} \in \{0, 1\}, \quad \forall (i,j) \in A.
 \end{aligned} \tag{12}$$

6.2 Baselines

We compare MMILP-GNN against the following baselines:

- **GCN** [28]: A simplified ChebNet using a first-order Chebyshev polynomial approximation.
- **rGCN** [36]: GCN with the random feature trick.
- **GIN** [45]: A standard graph isomorphism network.
- **rGIN**: GIN with the random feature trick.
- **GAT** [40]: Graph Attention Networks.

- **SCIP** [6]: We formulate both MFI and SPI as MILPs and use SCIP, a leading non-commercial optimization software package incorporating branch-and-bound, as a strong baseline. SCIP’s configuration details, including parameter settings and termination criteria, are provided in Appendix C.

6.3 Experimental Setup

For each dataset, we used a 50%/25%/25% train/validation/test split. MMILP-GNN outputs marginal interdiction probabilities for each edge. We evaluate performance using two strategies:

- **End-to-End Prediction:** For an interdiction budget k , this strategy selects the top- k edges with the highest predicted marginal probabilities.
- **Predict-and-Search:** This strategy uses a trust-region-based algorithm, initialized with MMILP-GNN’s marginal probabilities, to search for high-quality feasible solutions [21, 22]. Details are in Appendix B.

6.4 Evaluation Rationale

We use two strategies for different purposes. End-to-end prediction is *not* expected to outperform SCIP, as SCIP aims for optimal solutions, and perfect generalization is unattainable for machine learning models (see Section 6.5). We use end-to-end prediction to compare MMILP-GNN with other learning-based baselines.

Predict-and-search leverages MMILP-GNN’s predictions to guide the search for better solutions *faster* than SCIP alone. While SCIP finds optimal solutions, it can be computationally expensive. Predict-and-search aims to demonstrate that MMILP-GNN prunes the search space, allowing a solver (like SCIP) to find good solutions more efficiently within a limited time budget.

6.5 End-to-End Prediction Results

For end-to-end prediction, we evaluated performance on the smaller SPI and MFI datasets (SPI20, SPI30, SPI100 and MFI20, MFI30, MFI100). Tables 1 and 2 demonstrate that MMILP-GNN consistently outperforms all learning-based baselines across these datasets. The approximation ratio, which measures the average ratio of the solution found to the optimal solution, is used as our primary evaluation metric. A higher approximation ratio indicates better performance for the max-min SPI problem, while a lower ratio is preferable for the min-max MFI problem. Figure 3 provides a visual example of MMILP-GNN’s predictions on an SPI instance.

It is important to clarify that these smaller instances can be solved by both learning-based methods and SCIP almost instantly (on GPU and CPU, respectively), with SCIP providing exact solutions. These end-to-end experiments serve two crucial purposes. First, they provide necessary empirical validation of our theoretical findings presented in Section 5. In Section 5, we prove the existence of an MMILP-GNN capable of representing network interdiction problems. However, this theoretical result does not guarantee that such a model can be effectively realized through stochastic optimization. The end-to-end experiments demonstrate the *real-world performance* of our proposed MMILP-GNN model, showcasing its practical applicability and highlighting areas for potential future improvement. Second, these experiments demonstrate the *relative*

performance gains achieved by our multipartite-based MMILP-GNN compared to traditional GNN architectures. This comparison underscores the effectiveness of our tailored representation and learning approach for the network interdiction problem.

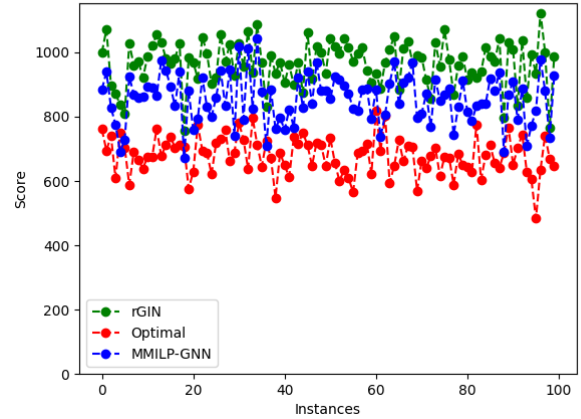


Figure 3: Example Shortest Path Interdiction Instance

Dataset	Method	Approx. Ratio	Optimality Gap
MFI20	rGIN	1.33 ± 0.07	14.2 ± 0.92
	GIN	1.35 ± 0.08	14.5 ± 1.02
	GCN	1.42 ± 0.06	15.2 ± 0.65
	rGCN	1.40 ± 0.05	13.9 ± 0.85
	GAT	1.39 ± 0.06	14.5 ± 0.72
	MMILP-GNN	1.22 ± 0.06	8.2 ± 0.32
MFI30	rGIN	1.43 ± 0.02	23.2 ± 0.53
	GIN	1.46 ± 0.03	23.8 ± 0.68
	GCN	1.59 ± 0.07	25.8 ± 0.97
	rGCN	1.50 ± 0.05	21.5 ± 0.75
	GAT	1.47 ± 0.06	22.0 ± 0.82
	MMILP-GNN	1.28 ± 0.02	9.6 ± 0.69
MFI100	rGIN	1.48 ± 0.19	46.2 ± 3.48
	GIN	1.51 ± 0.18	47.5 ± 3.60
	GCN	1.59 ± 0.23	59.2 ± 5.89
	rGCN	1.50 ± 0.15	49.0 ± 3.99
	GAT	1.55 ± 0.17	51.2 ± 4.21
	MMILP-GNN	1.33 ± 0.17	38.2 ± 3.83

Table 1: End-to-end prediction results for MFI datasets. Lower approximation ratio is better.

6.6 Predict-and-Search Results

For predict-and-search, we leverage MMILP-GNN to generate marginal interdiction probabilities, which then guide a trust-region-based algorithm in searching for near-optimal solutions. We compare this combined approach (MMILP-GNN with predict-and-search) against SCIP, a leading non-commercial MILP solver, within a limited time frame. The inference time for MMILP-GNN consists of

Dataset	Method	Approx. Ratio	Optimality Gap
SPI20	rGIN	0.75 ± 0.18	19.75 ± 2.56
	GIN	0.78 ± 0.19	20.25 ± 2.70
	GCN	0.69 ± 0.17	23.25 ± 3.37
	rGCN	0.78 ± 0.15	20.85 ± 2.80
	GAT	0.74 ± 0.16	21.95 ± 3.01
	MMILP-GNN	0.82 ± 0.14	18.25 ± 2.73
SPI30	rGIN	0.68 ± 0.25	20.5 ± 2.66
	GIN	0.71 ± 0.22	21.0 ± 2.78
	GCN	0.64 ± 0.29	26.25 ± 5.25
	rGCN	0.70 ± 0.22	22.75 ± 3.01
	GAT	0.66 ± 0.24	24.10 ± 3.68
	MMILP-GNN	0.81 ± 0.17	19.75 ± 2.37
SPI100	rGIN	0.72 ± 0.22	24.96 ± 5.24
	GIN	0.74 ± 0.21	25.75 ± 5.43
	GCN	0.56 ± 0.26	27.75 ± 6.38
	rGCN	0.69 ± 0.19	24.50 ± 5.02
	GAT	0.68 ± 0.20	25.75 ± 5.88
	MMILP-GNN	0.75 ± 0.19	23.25 ± 4.18

Table 2: End-to-end prediction results for SPI datasets. Higher approximation ratio is better.

two components: model inference (performed on GPUs) and the subsequent search time (using SCIP). The model inference time is negligible compared to the search time.

Due to computational resource limitations, we conducted predict-and-search experiments on 2,000 instances each of the 200-node MFI problem (MFI200) and the 200-node SPI problem (SPI200). While 200 nodes might appear modest, network interdiction problems are inherently NP-hard, making even instances of this size computationally challenging for exact solvers like SCIP.

Figures 4 and 5 present the averaged solving times from the 2,000 runs for each method on the MFI200 and SPI200 datasets, respectively. The x-axis in both figures represents solving time. SCIP’s execution consists of two stages: presolving and solving (branch-and-bound). Because the presolving time is similar for SCIP both with and without the guidance of MMILP-GNN, and because our primary focus is on the efficiency of the search phase, only the solving time (branch-and-bound) is plotted in the figures.

For both the MFI200 and SPI200 datasets, the predict-and-search approach using MMILP-GNN demonstrates a significant reduction in solving time compared to SCIP alone. This improvement is noteworthy, considering SCIP’s reputation as a highly efficient non-commercial MILP solver.

7 Limitations

While our proposed MMILP-GNN approach offers promising results for network interdiction, it is important to acknowledge certain limitations. *First*, our framework relies on MILP formulations of network interdiction problems. While we demonstrate effectiveness on SPI and MFI, not all interdiction problems readily lend themselves to MILP representations. Problems involving nonlinear constraints, multi-stage decisions, or stochastic elements may pose significant challenges or even preclude MILP formulations, thus limiting the broader applicability of our current approach. Future

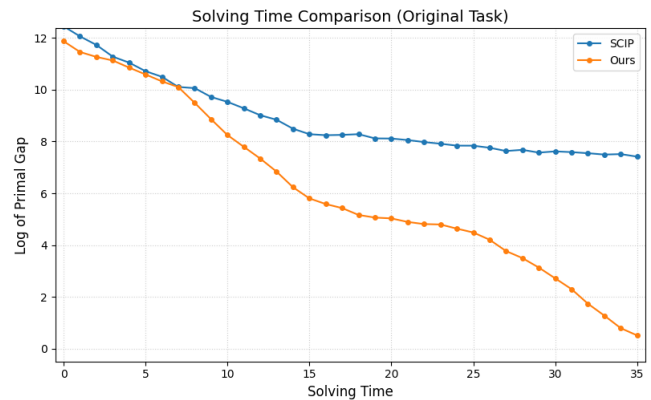


Figure 4: Predict and search against SCIP on SPI200 dataset: Averaged results visualization from 2,000 runs for each of the two methods.

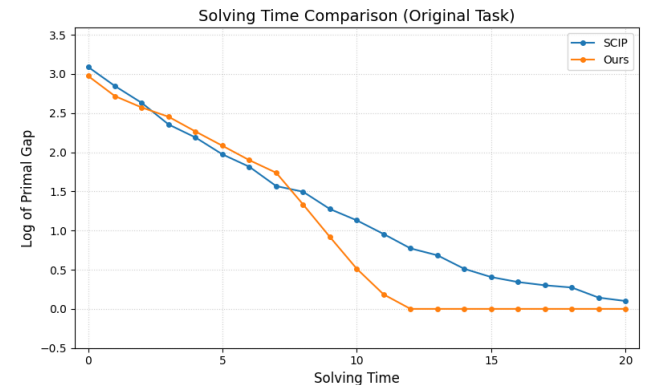


Figure 5: Predict and search against SCIP on MFI200 dataset: Averaged results visualization from 2,000 runs for each of the two methods.

research could explore extending MMILP-GNN to handle such complexities. *Second*, our theoretical analysis of representation power is a crucial first step, but it does not yet guarantee learnability or generalization. While we have shown that MMILP-GNN *can* represent relevant interdiction problems, further investigation is needed to understand how effectively these representations can be learned through stochastic optimization and how well the trained models generalize to unseen instances. This requires further theoretical and empirical analysis. *Finally*, our approach currently relies on supervised learning with pre-labeled data. While MMILP-GNN itself has polynomial complexity, generating labels for large-scale network interdiction problems using exact solvers (which are typically NP-hard) can be prohibitively expensive. This labeling bottleneck could hinder the practical application of our method to very large real-world instances. Future work might explore alternative learning paradigms, such as reinforcement learning or unsupervised learning, to mitigate this labeling challenge.

8 Conclusion

In this paper, we introduced a novel framework for addressing network interdiction problems using GNNs integrated with MILP formulations. Traditional solvers, such as SCIP, are highly optimized and serve as formidable benchmarks with strong performance guarantees. Achieving results that are competitive with—or even surpass—these methods is a significant breakthrough, underscoring the potential of neural networks in solving complex bi-level optimization problems. Beyond empirical performance, a key contribution of our work lies in the theoretical analysis that establishes the representational capacity of GNNs for network interdiction. This theoretical guarantee is essential, offering deeper insights into how GNNs approximate critical properties of these problems and enhancing the reliability of our approach.

In summary, this work marks an important step toward applying GNNs to bi-level combinatorial optimization. By combining theoretical foundations with empirical validation, we provide a pathway for future research on neural network-based solvers, with broad applications in network security, resource allocation, and infrastructure defense. Additionally, our findings open new research directions, such as developing alternative learning paradigms and improving the generalization and extrapolation capabilities of learning-based approaches for network interdiction and other combinatorial optimization problems.

References

- [1] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large Language Models for Mathematical Reasoning: Progresses and Challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, Neele Falk, Sara Papi, and Mike Zhang (Eds.). Association for Computational Linguistics, St. Julian's, Malta, 225–237. <https://aclanthology.org/2024.eacl-srw.17/>
- [2] Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. 2020. Learning what to defer for maximum independent sets. In *International conference on machine learning*. PMLR, 134–144.
- [3] Yunsheng Bai, Hao Ding, Ken Gu, Yizhou Sun, and Wei Wang. 2020. Learning-Based Efficient Graph Similarity Computation via Multi-Scale Convolutional Set Matching. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*. AAAI Press, 3219–3226. doi:10.1609/AAAI.V34I04.5720
- [4] Michael O Ball. 1986. Computational complexity of network reliability analysis: An overview. *Ieee transactions on reliability* 35, 3 (1986), 230–239.
- [5] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research* 290, 2 (2021), 405–421.
- [6] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schläpfer, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. 2021. *The SCIP Optimization Suite 8.0*. Technical Report. Optimization Online. http://www.optimization-online.org/DB_HTML/2021/12/8728.html
- [7] Maximilian Böther, Otto Kifbig, Martin Taraz, Sarel Cohen, Karen Seidel, and Tobias Friedrich. 2022. What's Wrong with Deep Learning in Tree Search for Combinatorial Optimization. *CoRR* abs/2201.10494 (2022). [arXiv:2201.10494](https://arxiv.org/abs/2201.10494)
- [8] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. 2023. Combinatorial Optimization and Reasoning with Graph Neural Networks. *J. Mach. Learn. Res.* 24 (2023), 130:1–130:61. <https://jmlr.org/papers/v24/21-0449.html>
- [9] Ziang Chen, Jialin Liu, Xinshang Wang, and Wotao Yin. 2023. On Representing Linear Programs by Graph Neural Networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1–5, 2023*. OpenReview.net. <https://openreview.net/forum?id=cP2QVK-uygd>
- [10] Ziang Chen, Jialin Liu, Xinshang Wang, and Wotao Yin. 2023. On Representing Mixed-Integer Linear Programs by Graph Neural Networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1–5, 2023*. OpenReview.net. <https://openreview.net/forum?id=4gc3MGZra1d>
- [11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2022. *Introduction to Algorithms* (4th ed.). MIT Press, Cambridge, MA.
- [12] Kelly J Cormican. 1995. *Computational Methods for Deterministic and Stochastic Network Interdiction Problems*. Technical Report. NAVAL POSTGRADUATE SCHOOL MONTEREY CA.
- [13] Kelly J Cormican, David P Morton, and R Kevin Wood. 1998. Stochastic network interdiction. *Operations Research* 46, 2 (1998), 184–197.
- [14] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucang Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyuan Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shutong Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjuan Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zhou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] <https://arxiv.org/abs/2501.12948>
- [15] Ding-Zhu Du and Panos M Pardalos. 2013. *Handbook of Combinatorial Optimization: Supplement Volume A*. Vol. 1. Springer Science & Business Media.
- [16] Andrew Joseph Dudzik and Petar Velickovic. 2022. Graph Neural Networks are Dynamic Programmers. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sammi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.). http://papers.nips.cc/paper_files/paper/2022/hash/8248b1ded388fcd9bd121bcdfea3068c-Abstract-Conference.html
- [17] Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. 2020. Deep Graph Matching Consensus. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020*. OpenReview.net. <https://openreview.net/forum?id=HyeJf1HKvS>
- [18] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. 2019. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems* 32 (2019).
- [19] Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual. <https://www.gurobi.com>
- [20] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (Long Beach, California, USA) (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1025–1035.
- [21] Qingyu Han, Linxin Yang, Qian Chen, Xiang Zhou, Dong Zhang, Akang Wang, Ruoyu Sun, and Xiaodong Luo. 2022. A GNN-Guided Predict-and-Search Framework for Mixed-Integer Linear Programming. In *The Eleventh International Conference on Learning Representations*.
- [22] Qingyu Han, Linxin Yang, Qian Chen, Xiang Zhou, Dong Zhang, Akang Wang, Ruoyu Sun, and Xiaodong Luo. 2023. A GNN-Guided Predict-and-Search Framework for Mixed-Integer Linear Programming. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=pHmPgT5xWaE>

- [23] Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. 2022. Graph Neural Network Guided Local Search for the Traveling Salesperson Problem. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=ar92oEosBlg>
- [24] Eitan Israeli and R Kevin Wood. 2002. Shortest-path network interdiction. *Networks: An International Journal* 40, 2 (2002), 97–111.
- [25] Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. 2022. Learning the travelling salesperson problem requires rethinking generalization. *Constraints* 27, 1 (2022), 70–98.
- [26] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. 2019. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227* (2019).
- [27] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning Combinatorial Optimization Algorithms over Graphs. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/d9896106ca98d3d05b8cbdf4fd8b13a1-Paper.pdf
- [28] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=SJU4ayYgl>
- [29] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. 2019. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*. PMLR, 3835–3845.
- [30] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems* 31 (2018).
- [31] Churlzu Lim and J Cole Smith. 2007. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IEEE Transactions* 39, 1 (2007), 15–26.
- [32] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid Von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. 2020. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349* (2020).
- [33] Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. 2019. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems* 32 (2019).
- [34] Marcelo Prates, Pedro HC Avelar, Henrique Lemos, Luis C Lamb, and Moshe Y Vardi. 2019. Learning to solve np-complete problems: A graph neural network for decision tsp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4731–4738.
- [35] Antoine Prouvost, Justin Dumouchelle, Lara Scavuzzo, Maxime Gasse, Didier Chételat, and Andrea Lodi. 2020. Ecole: A Gym-like Library for Machine Learning in Combinatorial Optimization Solvers. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*. <https://openreview.net/forum?id=IVc9hgibyB>
- [36] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. 2021. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*. SIAM, 333–341.
- [37] Luca E Schäfer. 2021. GitHub - Luca-Elias-Schaefer/Gurobi-Models: In this project, one can find an extensive collection of classical optimization problems along with a proposed solution procedure using Gurobi. — [github.com](https://github.com/Luca-Elias-Schaefer/Gurobi-Models). <https://github.com/Luca-Elias-Schaefer/Gurobi-Models>. [Accessed 21-05-2024].
- [38] J Cole Smith and Yongjia Song. 2020. A survey of network interdiction models and algorithms. *European Journal of Operational Research* 283, 3 (2020), 797–811.
- [39] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. 2020. Reinforcement learning for integer programming: Learning to cut. In *International conference on machine learning*. PMLR, 9367–9376.
- [40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [41] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in neural information processing systems* 28 (2015).
- [42] Runzhong Wang, Zhigang Hua, Gan Liu, Jiayi Zhang, Junchi Yan, Feng Qi, Shuang Yang, Jun Zhou, and Xiaokang Yang. 2021. A bi-level framework for learning to solve combinatorial optimization on graphs. *Advances in neural information processing systems* 34 (2021), 21453–21466.
- [43] Zhihai Wang, Xijun Li, Jie Wang, Yufei Kuang, Mingxuan Yuan, Jia Zeng, Yongdong Zhang, and Feng Wu. 2023. Learning Cut Selection for Mixed-Integer Linear Programming via Hierarchical Sequence Model. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=Zob4P9bRNcK>
- [44] R Kevin Wood. 1993. Deterministic network interdiction. *Mathematical and Computer Modelling* 17, 2 (1993), 1–18.
- [45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=ryGs6iA5Kmn>
- [46] Keyulu Xu, Jingling Li, Mzhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2020. What Can Neural Networks Reason About?. In *8th*

International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net. <https://openreview.net/forum?id=rjxbjeHFPS>

A Proofs for Section 5

Our approach to measuring the representation power of MMILP-GNN is heavily inspired by existing research on the representation power of GNNs [46] and the separation power of GNNs in LP [9] and MILP [10]. We also prove Lemma 5.2 and Lemma 5.3 in the same way corresponding lemmas and theorems are proved in [9].

PROOF OF LEMMA 5.2. Define $p+1$ sets (not multisets) that collect different colors:

$$C_{l-1}^V = \{C_1^{l-1,V}, \dots, C_m^{l-1,V}, \hat{C}_1^{l-1,V}, \dots, \hat{C}_m^{l-1,V}\},$$

and

$$C_{l-1}^{W_k} = \{C_1^{l-1,W_k}, \dots, C_n^{l-1,W_k}, \hat{C}_1^{l-1,W_k}, \dots, \hat{C}_n^{l-1,W_k}\}.$$

We aim to prove by induction that for any $l \in \{0, 1, \dots, L\}$, the followings hold:

- (i) $C_i^{l,V} = C_{i'}^{l,V}$ implies $h_i^{l,V} = h_{i'}^{l,V}$, for $1 \leq i, i' \leq m$;
- (ii) $\hat{C}_i^{l,V} = \hat{C}_{i'}^{l,V}$ implies $\hat{h}_i^{l,V} = \hat{h}_{i'}^{l,V}$, for $1 \leq i, i' \leq m$;
- (iii) $C_j^{l,V} = \hat{C}_{j'}^{l,V}$ implies $h_j^{l,V} = \hat{h}_{j'}^{l,V}$, for $1 \leq i, i' \leq m$;
- (iv) $C_j^{l,W_k} = C_{j'}^{l,W_k}$ implies $h_j^{l,W_k} = h_{j'}^{l,W_k}$, for $1 \leq j, j' \leq n_k$;
- (v) $\hat{C}_j^{l,W_k} = \hat{C}_{j'}^{l,W_k}$ implies $\hat{h}_j^{l,W_k} = \hat{h}_{j'}^{l,W_k}$, for $1 \leq j, j' \leq n_k$;
- (vi) $C_j^{l,W_k} = \hat{C}_{j'}^{l,W_k}$ implies $h_j^{l,W_k} = \hat{h}_{j'}^{l,W_k}$, for $1 \leq j, j' \leq n_k$.

The above claims (i)-(vi) are clearly true for $l = 0$ due to the injectivity of HASH_V^0 and $\text{HASH}_{W_k}^0$. Now we assume that (i)-(vi) are true for some $l-1 \in \{0, 1, \dots, L-1\}$. Suppose that $C_i^{l,V} = C_{i'}^{l,V}$, i.e.,

$$\begin{aligned} & \text{HASH}_V^l \left(C_i^{l-1,V}, \sum_{k=0}^p \sum_{j=1}^{n_k} E_{i,j}^{W_k,V} \text{HASH}_{W_k}^{l'} \left(C_j^{l-1,W_k} \right) \right) \\ &= \text{HASH}_{V'}^l \left(C_{i'}^{l-1,V}, \sum_{k=0}^p \sum_{j=1}^{n_k} E_{i',j}^{W_k,V} \text{HASH}_{W_k}^{l'} \left(C_j^{l-1,W_k} \right) \right), \end{aligned}$$

for some $1 \leq i, i' \leq m$. It follows from the injectivity of HASH_V^l that

$$C_i^{l-1,V} = C_{i'}^{l-1,V}, \quad (13)$$

and

$$\begin{aligned} & \sum_{k=0}^p \sum_{j=1}^{n_k} E_{i,j}^{W_k,V} \text{HASH}_{W_k}^{l'} \left(C_j^{l-1,W_k} \right) \\ &= \sum_{k=0}^p \sum_{j=1}^{n_k} E_{i',j}^{W_k,V} \text{HASH}_{W_k}^{l'} \left(C_j^{l-1,W_k} \right). \end{aligned}$$

According to the linearly independent property of $\text{HASH}_{W_k}^{l'}$, the above equation implies that

$$\sum_{C_j^{l-1,W_k}=C} E_{i,j} = \sum_{C_j^{l-1,W_k}=C} E_{i',j}, \quad \forall C \in C_{l-1}^{W_k}. \quad (14)$$

Note that the induction assumption guarantees that $h_j^{l-1,W_k} = h_{j'}^{l-1,W_k}$ as long as $C_j^{l-1,W_k} = C_{j'}^{l-1,W_k}$. So one can assign for each $C \in C_{l-1}^{W_k}$ some $h(C) \in \mathbb{R}^{d_{l-1}}$ such that $h_j^{l-1,W_k} = h(C)$ as long as

$C_j^{l-1, W_k} = C$ for any $1 \leq j \leq n_k$. Therefore, it follows from (14) that

$$\begin{aligned} \sum_{j=1}^n E_{i,j} f_l^{W_k}(h_j^{l-1, W_k}) &= \sum_{C \in \mathcal{C}_{l-1}^{W_k}} \sum_{C_j^{l-1, W_k} = C} E_{i,j} f_l^{W_k}(h(C)) \\ \sum_{j=1}^n E_{i',j} f_l^{W_k}(h_j^{l-1, W_k}) &= \sum_{C \in \mathcal{C}_{l-1}^{W_k}} \sum_{C_j^{l-1, W_k} = C} E_{i',j} f_l^{W_k}(h(C)) \\ \sum_{j=1}^n E_{i,j} f_l^{W_k}(h_j^{l-1, W_k}) &= \sum_{j=1}^n E_{i',j} f_l^{W_k}(h_j^{l-1, W_k}) \end{aligned}$$

Note also that (13) and the induction assumption lead to $h_i^{l-1, V} = h_{i'}^{l-1, V}$. Then one can conclude that

$$\begin{aligned} h_i^{l, V} &= g_l^V \left(h_i^{l-1, V}, \sum_{k=0}^p \sum_{j=1}^{n_k} E_{i,j}^{W_k, V} f_l^{W_k}(h_j^{l-1, W_k}) \right) \\ &= g_l^V \left(h_{i'}^{l-1, V}, \sum_{k=0}^p \sum_{j=1}^{n_k} E_{i',j}^{W_k, V} f_l^{W_k}(h_j^{l-1, W_k}) \right) = h_{i'}^{l, V}. \end{aligned}$$

This proves the claim (i) for l . The other five claims can be proved using similar arguments.

Therefore, we obtain from $(G, H) \sim (\hat{G}, \hat{H})$ that

$$\left\{ \left\{ h_1^{L, V}, h_2^{L, V}, \dots, h_m^{L, V} \right\} \right\} = \left\{ \left\{ \hat{h}_1^{L, V}, \hat{h}_2^{L, V}, \dots, \hat{h}_m^{L, V} \right\} \right\},$$

and that

$$\left\{ \left\{ h_1^{L, W_k}, h_2^{L, W_k}, \dots, h_n^{L, W_k} \right\} \right\} = \left\{ \left\{ \hat{h}_1^{L, W_k}, \hat{h}_2^{L, W_k}, \dots, \hat{h}_n^{L, W_k} \right\} \right\}.$$

By the definition of the output layer, the above conclusion guarantees that $F(G, H) = \sigma(F(\hat{G}, \hat{H}))$ for some $\sigma \in S_n$. \square

PROOF OF LEMMA 5.3. It suffices to prove that, if (G, H) can be distinguished from (\hat{G}, \hat{H}) by the WL test, then there exists $F \in \mathcal{F}$, such that $F(G, H) \neq F(\hat{G}, \hat{H})$. The distinguish-ability of the WL test implies that there exists $L \in \mathbb{N}$ and hash functions, $\text{HASH}_{0, V}$, $\text{HASH}_{0, W}$, $\text{HASH}_{l, V}$, $\text{HASH}_{l, W}$, $\text{HASH}'_{l, V}$, and $\text{HASH}'_{l, W}$, for $l = 1, 2, \dots, L$, such that

$$\left\{ \left\{ C_1^{L, V}, C_2^{L, V}, \dots, C_m^{L, V} \right\} \right\} \neq \left\{ \left\{ \hat{C}_1^{L, V}, \hat{C}_2^{L, V}, \dots, \hat{C}_m^{L, V} \right\} \right\}, \quad (15)$$

or

$$\left\{ \left\{ C_1^{L, W}, C_2^{L, W}, \dots, C_n^{L, W} \right\} \right\} \neq \left\{ \left\{ \hat{C}_1^{L, W}, \hat{C}_2^{L, W}, \dots, \hat{C}_n^{L, W} \right\} \right\}, \quad (16)$$

We aim to construct some GNNs such that the followings hold for any $l = 0, 1, \dots, L$:

- (i) $h_i^{l, V} = h_{i'}^{l, V}$ implies $C_i^{l, V} = C_{i'}^{l, V}$, for $1 \leq i, i' \leq m$;
- (ii) $\hat{h}_i^{l, V} = \hat{h}_{i'}^{l, V}$ implies $\hat{C}_i^{l, V} = \hat{C}_{i'}^{l, V}$, for $1 \leq i, i' \leq m$;
- (iii) $h_j^{l, V} = \hat{h}_{j'}^{l, V}$ implies $C_j^{l, V} = \hat{C}_{j'}^{l, V}$, for $1 \leq i, i' \leq m$;
- (iv) $h_j^{l, W_k} = h_{j'}^{l, W_k}$ implies $C_j^{l, W_k} = C_{j'}^{l, W_k}$, for $1 \leq j, j' \leq n$;
- (v) $\hat{h}_j^{l, W_k} = \hat{h}_{j'}^{l, W_k}$ implies $\hat{C}_j^{l, W_k} = \hat{C}_{j'}^{l, W_k}$, for $1 \leq j, j' \leq n$;
- (vi) $h_j^{l, W_k} = \hat{h}_{j'}^{l, W_k}$ implies $C_j^{l, W_k} = \hat{C}_{j'}^{l, W_k}$, for $1 \leq j, j' \leq n$.

It is clear that the above conditions (i)-(vi) hold for $l = 0$ as long as we choose $f_{\text{in}}^{V'}$ that is injective on the following $p + 1$ sets (not multisets) respectively: $\{h_1^V, \dots, h_m^V, \hat{h}_1^V, \dots, \hat{h}_m^V\}$ and $\{h_1^{W_k}, \dots, h_n^{W_k}, \hat{h}_1^{W_k}, \dots, \hat{h}_n^{W_k}\}$. We then assume that (i)-(vi) hold for some $0 \leq l - 1 < L$, and show that these conditions are also satisfied for l if we choose $f_l^V, f_l^W, g_l^V, g_l^W$ properly. Let us consider the set (not multiset):

$$\{\alpha_1, \alpha_2, \dots, \alpha_s\} \subset \mathbb{R}^{d_{l-1}}$$

that collects all different values in $h_1^{l-1, W}, h_2^{l-1, W}, \dots, h_n^{l-1, W}, \hat{h}_1^{l-1, W}, \hat{h}_2^{l-1, W}, \dots, \hat{h}_n^{l-1, W}$. Let $d_l \geq s$ and let $e_p^{d_l} = (0, \dots, 0, 1, 0, \dots, 0)$ be the vector in \mathbb{R}^{d_l} with the p -th entry being 1 and all other entries being 0, for $1 \leq p \leq s$. Choose $f_l^W : \mathbb{R}^{d_{l-1}} \rightarrow \mathbb{R}^{d_l}$ as a continuous function satisfying $f_l^W(\alpha_p) = e_p^{d_l}$, $p = 1, 2, \dots, s$, and choose $g_l^V : \mathbb{R}^{d_{l-1}} \times \mathbb{R}^{d_l} \rightarrow \mathbb{R}^{d_l}$ that is continuous and is injective when restricted on the set (not multiset)

$$\begin{aligned} &\left\{ \left(h_i^{l-1, V}, \sum_{j=1}^n E_{i,j} f_l^W(h_j^{l-1, W}) \right) : 1 \leq i \leq m \right\} \\ &\cup \left\{ \left(\hat{h}_i^{l-1, V}, \sum_{j=1}^n \hat{E}_{i,j} f_l^W(\hat{h}_j^{l-1, W}) \right) : 1 \leq i \leq m \right\}. \end{aligned}$$

Noticing that

$$\sum_{j=1}^n E_{i,j} f_l^W(h_j^{l-1, W}) = \sum_{p=1}^s \left(\sum_{h_j^{l-1, W} = \alpha_p} E_{i,j} \right) e_p^{d_l},$$

and that $\{e_1^{d_l}, e_2^{d_l}, \dots, e_s^{d_l}\}$ is linearly independent, one can conclude that $h_i^{l, V} = h_{i'}^{l, V}$ if and only if $h_i^{l-1, V} = h_{i'}^{l-1, V}$ and $\sum_{j=1}^n E_{i,j} f_l^W(h_j^{l-1, W}) = \sum_{j=1}^n E_{i',j} f_l^W(h_j^{l-1, W})$, where the second condition is equivalent to

$$\sum_{h_j^{l-1, W} = \alpha_p} E_{i,j} = \sum_{h_j^{l-1, W} = \alpha_p} E_{i',j}, \quad \forall p \in \{1, 2, \dots, s\}.$$

This, as well as the condition (iv) for $l - 1$, implies that

$$\sum_{j=1}^n E_{i,j} \text{HASH}'_{l, W_k}(C_j^{l-1, W}) = \sum_{j=1}^n E_{i',j} \text{HASH}'_{l, W_k}(C_j^{l-1, W}),$$

and hence that $C_i^{l, V} = C_{i'}^{l, V}$ by using $h_i^{l-1, V} = h_{i'}^{l-1, V}$ and condition (i) for $l - 1$. Therefore, we know that (i) is satisfied for l , and one can show (ii) and (iii) for l using similar arguments by taking d_l large enough. In addition, f_l^V and g_l^W can also be chosen in a similar way such that (iv)-(vi) are satisfied for l .

Combining (15), (16), and condition (i)-(iv) for L , we obtain that

$$\left\{ \left\{ h_1^{L, V}, h_2^{L, V}, \dots, h_m^{L, V} \right\} \right\} \neq \left\{ \left\{ \hat{h}_1^{L, V}, \hat{h}_2^{L, V}, \dots, \hat{h}_m^{L, V} \right\} \right\}, \quad (17)$$

or

$$\left\{ \left\{ h_1^{L, W_k}, h_2^{L, W_k}, \dots, h_n^{L, W_k} \right\} \right\} \neq \left\{ \left\{ \hat{h}_1^{L, W_k}, \hat{h}_2^{L, W_k}, \dots, \hat{h}_n^{L, W_k} \right\} \right\}.$$

Without loss of generality, we can assume that (17) holds.

Consider the set (not multiset)

$$\{\beta_1, \beta_2, \dots, \beta_t\} \subset \mathbb{R}^{d_L},$$

Parameter	Value
randomization/randomseedshift	0
randomization/lpseed	0
randomization/permutationseed	0
separating/maxrounds	0
presolving/maxrestarts	0
Heuristics	AGGRESSIVE

Table 3: SCIP solver parameter settings .

that collects all different values in $h_1^{L,V}, h_2^{L,V}, \dots, h_m^{L,V}, \hat{h}_1^{L,V}, \hat{h}_2^{L,V}, \dots, \hat{h}_m^{L,V}$. Let $k > 1$ be a positive integer that is greater than the maximal multiplicity of an element in the multisets $\{\{h_1^{L,V}, h_2^{L,V}, \dots, h_m^{L,V}\}\}$ and $\{\{\hat{h}_1^{L,V}, \hat{h}_2^{L,V}, \dots, \hat{h}_m^{L,V}\}\}$. There exists a continuous function $\varphi : \mathbb{R}^{d_L} \rightarrow \mathbb{R}$ such that $\varphi(\beta_q) = k^q$ for $q = 1, 2, \dots, t$, and due to (17) and the fact that the way of writing an integer as k -ary expression is unique, it hence holds that

$$\sum_{i=1}^m \varphi(h_i^{L,V}) \neq \sum_{i=1}^m \varphi(\hat{h}_i^{L,V}).$$

Set the dimension of $(L + 1)$ -th layer as 1: $d_{L+1} = 1$,

$$f_{\text{out}} \left(\sum_{i=1}^m h_i^{L+1,V}, \sum_{j=1}^n h_j^{L+1,W} \right) = \sum_{i=1}^m \varphi(h_i^{L,V})$$

$$\neq \sum_{i=1}^m \varphi(\hat{h}_i^{L,V}) = f_{\text{out}} \left(\sum_{i=1}^m \hat{h}_i^{L+1,V}, \sum_{j=1}^n \hat{h}_j^{L+1,W} \right),$$

which guarantees the existence of $F \in \mathcal{F}_{\text{GNN}}$ that has $L + 1$ layers and satisfies $F(G, H) \neq F(\hat{G}, \hat{H})$. \square

B The Predict-and-Search Strategy

The predict-and-search strategy, as employed in the work of Han et al. [21], involved predicting solution distributions and subsequently conducting a search for near-optimal solutions within a trust region derived from the predictions. The algorithm detailing this approach is outlined in Algorithm 2.

Algorithm 2 Predict-and-search Algorithm (Adapted from [21])

Parameter: Size $\{k_0, k_1\}$, radius of the neighborhood: Δ

Input: Instance M , Probability prediction $F_\theta(M)$

Output: Solution $x \in \mathbb{R}^n$

- 1: Sort the components in $F_\theta(M)$ from smallest to largest to obtain sets I_0 and I_1 .
 - 2: **for** $d = 1 : n$ **do**
 - 3: **if** $d \in I_0 \cup I_1$ **then**
 - 4: **create binary variable** δ_d
 - 5: **if** $d \in I_0$ **then**
 - 6: **create constraint**
 - 7: $x_d \leq \delta_d$
 - 8: **else**
 - 9: **create constraint**
 - 10: $1 - x_d \leq \delta_d$
 - 11: **end if**
 - 12: **end if**
 - 13: **end for**
 - 14: **create constraint** $\sum_{d \in I_0 \cup I_1} \delta_d \leq \Delta$
 - 15: Let M' denote the instance M with new constraints and variables
 - 16: Let $x = \text{SOLVE}(M')$
 - 17: **return** x
-

C SCIP Parameter Settings

The parameter settings for the SCIP solver are shown in Table 3.

D Other Experiment Settings

The implementation of our MMLP-GNN is modified from the branching-imitation module in the ecole package with Pytorch [35]. We use ADAM as the optimizer with a 0.0001 learning rate. All experiments were conducted on a desktop with one Intel 12900K CPU and one Nvidia Tesla V100 GPU, with 64GB RAM.