ON REPRESENTING MIXED-INTEGER LINEAR PROGRAMS BY GRAPH NEURAL NETWORKS

ZIANG CHEN, JIALIN LIU, XINSHANG WANG, JIANFENG LU, AND WOTAO YIN

Abstract. While Mixed-integer linear programming (MILP) is NP-hard in general, practical MILP has received roughly 100–fold speedup in the past twenty years. Still, many classes of MILPs quickly become unsolvable as their sizes increase, motivating researchers to seek new acceleration techniques for MILPs. With deep learning, they have obtained strong empirical results, and many results were obtained by applying graph neural networks (GNNs) to making decisions in various stages of MILP solution processes. This work discovers a fundamental limitation: there exist feasible and infeasible MILPs that all GNNs will, however, treat equally, indicating GNN's lacking power to express general MILPs. Then, we show that, by restricting the MILPs to unfoldable ones or by adding random features, there exist GNNs that can reliably predict MILP feasibility, optimal objective values, and optimal solutions up to prescribed precision. We conducted small-scale numerical experiments to validate our theoretical findings.

1. Introduction

Mixed-integer linear programming (MILP) is a type of optimization problems that minimize a linear objective function subject to linear constraints, where some or all variables must take integer values. MILP has a wide type of applications, such as transportation [36], control [31], scheduling [15], etc. Branch and Bound (B&B) [25], an algorithm widely adopted in modern solvers that exactly solves general MILPs to global optimality, unfortunately, has an exponential time complexity in the worst-case sense. To make MILP more practical, researchers have to analyze the features of each instance of interest based on their domain knowledge, and use such features to adaptively warm-start B&B or design the heuristics in B&B.

To automate such laborious process, researchers turn attention to Machine learning (ML) techniques in recent years [6]. The literature has reported some encouraging findings that a proper chosen ML model is able to learn some useful knowledge of MILP from data and generalize well to some similar but unseen instances. For example, one can learn fast approximations of Strong Branching, an effective but time-consuming branching strategy usually used in B&B [2, 22, 26, 43]. One may also learn cutting strategies [8, 21, 39], node selection/pruning strategies [20, 42], or decomposition strategies [38] with ML models. The role of ML models in those approaches can be summarized as: *approximating useful mappings or parameterizing*

Date: October 20, 2022.

A major part of the work of Z. Chen was completed during his internship at Alibaba US DAMO Academy.
Corresponding author: Jialin Liu, jialin.liu@alibaba-inc.com.

key strategies in MILP solvers, and these mappings/strategies usually take an MILP instance as input and output its key peroperties.

The graph neural network (GNN), due to its nice properties, say *permutation invariance*, is considered a suitable model to represent such mappings/strategies for MILP. More specifically, permutations on variables or constraints of an MILP do not essentially change the problem itself, reliable ML models such as GNNs should satisfy such properties, otherwise the model may overfit to the variable/constraint orders in the training data. [16] proposed that an MILP can be encoded into a bipartite graph on which one can use a GNN to approximate Strong Branching. [12] proposed to represent MILP with a tripartite graph. Since then, GNNs have been adopted to represent mappings/strategies for MILP, for example, approximating Strong Branching [18, 19, 28, 37], approximating optimal solution [23, 28], parameterizing cutting strategies [29], and parameterizing branching strategies [30, 35].

However, theoretical foundations in this direction still remain unclear. A key problem is the ability of GNN to approximate important mappings related with MILP. In this paper, we ask the following questions:

(Q1) Is GNN able to predict whether an MILP is feasible?

(Q2) Is GNN able to approximate the optimal objective value of an MILP?

(Q3) Is GNN able to approximate an optimal solution of an MILP?

To answer questions (Q1 - Q3), one needs theories of separation power and representation power of GNN. The ***separation power*** of GNN is measured by whether it can distinguish two non-isomorphic graphs. Given two graphs G_1, G_2, we say a mapping F (e.g. a GNN) has strong separation power if $F(G_1) \neq F(G_2)$ as long as G_1, G_2 that are not the isomorphic. In our settings, since MILPs are represented by graphs, the separation power actually indicates the ability of GNN to distinguish two different MILP instances. The ***representation power*** of GNN refers to how well it can approximate mappings with permutation invariant properties. In our settings, we study whether GNN can map an MILP to its feasiblity, optimal objective value and an optimal solution.

The separation power and representation power of GNNs are closely related to the Weisfeiler-Lehman (WL) test [40], a classical algorithm to identify whether two graphs are isomorphic or not. In the literature, it has been shown that GNN has the same separation power with the WL test [41], and, based on this result, GNNs can universally approximate continuous graph-input mappings with separation power no stronger than WL test [5, 17].

Our contributions. With the above tools in hand, one still cannot directly answer questions (Q1 - Q3) since the relationship between characteristics of general MILPs and properties of graphs are not clear yet. Although there are some works studying the representation power of GNN on some graph-related optimization problems [27, 33] and linear programming [11], representing general MILPs with GNNs are still not theoretically studied, to the best of our knowledge. Our contributions are listed below:

- (Limitation of GNNs for MILP) We show with an example that GNNs do not have strong enough separation power to distinguish any two different MILP instances. There exist two

MILPs such that one of them is feasible while the other one is not, but, unfortunately, all GNNs treat them equally without detecting the essential difference between them. In fact, there are infinitely many pairs of MILP instances that can puzzle GNN.

- (Foldable and unfoldable MILP) We provide a precise mathematical description on what type of MILPs makes GNNs fail. These hard MILP instances are named as *foldable MILPs*. We prove that, after restricting dataset to *unfoldable MILPs*, GNN has strong enough separation power and representation power to approximate the feasibility, optimal objective value and an optimal solution of an MILP.
- (MILP with random features) To handle those foldable MILPs, we propose to append random features to the MILP-induced graphs. We prove that, with the random feature technique, the answers to questions (Q1 - Q3) are affirmative.

The rest of this paper is organized as follows. We state some preliminaries in Section 2. The limitation of GNN for MILP is presented in Section 3 and we provide the descriptions of foldable and unfoldable MILPs in Section 4. The random feature technique is introduced in Section 5. Section 6 contains some numerical results and the whole paper is concluded in Section 7.

2. Preliminaries

In this section, we introduce some preliminaries that will be used throughout this paper. Our notations and definitions follow [11]. Consider a general MILP problem defined with:

$$\min_{x\in\mathbb{R}^n} \ c^\top x, \quad \text{s.t. } Ax\circ b,\ l\le x\le u,\ x_j\in\mathbb{Z},\ \forall\ j\in N_I, \tag{2.1}$$

where $A\in\mathbb{R}^{m\times n}$, $c\in\mathbb{R}^n$, $b\in\mathbb{R}^m$, $l\in(\mathbb{R}\cup\{-\infty\})^n$, $u\in(\mathbb{R}\cup\{+\infty\})^n$, and $\circ\in\{\le,=,\ge\}^m$. The index set $N_I\subset\{1,2,\dots,n\}$ includes those indices j where x_j are constrained to be an integer. The feasible set is defined with $X_{\text{feasible}}:=\{x\in\mathbb{R}^n|Ax\circ b,\ l\le x\le u,\ x_j\in\mathbb{Z},\forall\ j\in N_I\}$, and we say an MILP is *infeasible* if $X_{\text{feasible}}=\emptyset$ and *feasible* otherwise. For feasible MILPs, $\inf\{c^\top x : x\in X_{\text{feasible}}\}$ is named as the *optimal objective value*. If there exists $x^*\in X_{\text{feasible}}$ with $c^\top x^*\le c^\top x,\ \forall\ x\in X_{\text{feasible}}$, then we say that x^* is an *optimal solution*. It is possible that the objective value is arbitrarily good, i.e., for any $R>0$, $c^\top\hat{x}<-R$ holds for some $\hat{x}\in X_{\text{feasible}}$. In this case, we say the MILP is *unbounded* or its optimal objective value is $-\infty$.

2.1. **MILP as a weighted bipartite graph with vertex features.** All information in an MILP (2.1) can be represented into a *weighted bipartite graph*. The vertex set of such a graph is $V\cup W$, where $V=\{v_1,v_2,\dots,v_m\}$ with v_i corresponding to the i-th constraint, and $W=\{w_1,w_2,\dots,w_m\}$ with w_j corresponding to the j-th variable. The edge $E_{i,j}\in\mathbb{R}$ connects $v_i\in V$ and $w_j\in W$, and $E_{i,j}=0$ if there is no connection between v_i and w_j. Note that there is no edge connecting vertices in the same vertex group (V or W). Thus, the whole graph is denoted as $G=(V\cup W,E)$, and we denote $\mathcal{G}_{m,n}$ as the collection of all such weighted bipartite graphs whose two vertex groups have size m and n, respectively.

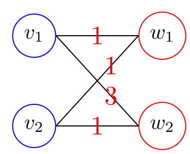
To fully represent all information in (2.1), we associate each vertex in $V\cup W$ with vertex features:

- The vertex $v_i \in V$ represents the i-th constraint in (2.1), and it is equipped with a feature vector $h_i^V = (b_i, \circ_i)$ that is in the space $\mathcal{H}^V = \mathbb{R} \times \{\leq, =, \geq\}$.
- The vertex $w_j \in W$ represents the j-th variable in (2.1), and it is equipped with a feature vector $h_j^W = (c_j, l_j, u_j, \tau_j)$, where $\tau_j = 1$ if $j \in N_I$ and $\tau_j = 0$ otherwise. The feature h_j^W is in the space $\mathcal{H}^W = \mathbb{R} \times (\mathbb{R} \cup \{-\infty\}) \times (\mathbb{R} \cup \{+\infty\}) \times \{0, 1\}$.
- The edge $E_{i,j}$ connects the i-th constraint and the j-th variable in (2.1) with the weight being $E_{i,j} = A_{i,j}$.

Finally, we define $\mathcal{H}_m^V := (\mathcal{H}^V)^m$ and $\mathcal{H}_n^W := (\mathcal{H}^W)^n$, and stack all the vertex features together as $H = (h_1^V, h_2^V, \ldots, h_m^V, h_1^W, h_2^W, \ldots, h_n^W) \in \mathcal{H}_m^V \times \mathcal{H}_n^W$. Then a weighted bipartite graph with vertex features $(G, H) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ contains all information in the MILP problem (2.1). This approach to represent a general MILP with a graph is first introduced in [16], and in this paper, we call the resulting graph as an *MILP-induced graph* or *MILP-graph*. If $N_I = \emptyset$, the feature τ_j can be dropped and the graphs reduce to LP-graphs in [11]. We provide an example of MILP-graph in Figure 1.

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \ & x_1 + x_2, \\ \text{s.t. } & x_1 + 3x_2 \geq 1, \ x_1 + x_2 \geq 1, \\ & x_1 \leq 3, \ x_2 \leq 5, \ x_2 \in \mathbb{Z}. \end{aligned}$$

$h_1^V = (1, \geq)$ $\quad$ $h_1^W = (1, -\infty, 3, 0)$



$h_2^V = (1, \geq)$ $\quad$ $h_2^W = (1, -\infty, 5, 1)$

FIGURE 1. An example of MILP-graph

2.2. **Graph neural networks with message passing for MILP-graphs.** To represent properties of the whole graph, one needs to build a GNN that maps (G, H) to a real number: $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R}$; to represent properties of each vertex in W (or represent properties for each variable), one needs to build a GNN that maps (G, H) to a vector: $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R}^n$. In this section, we explicitly describe the GNNs used in this paper.

The first step in the construction of GNNs is the embedding step. Let $f_{\text{in}}^V : \mathcal{H}^V \to \mathbb{R}^{d_0}$ and $f_{\text{in}}^W : \mathcal{H}^W \to \mathbb{R}^{d_0}$ be learnable functions that embed the vertex features, h_i^V and h_j^W, into initial hidden states $h_i^{0,V}, h_j^{0,W} \in \mathbb{R}^{d_0}$:

$$h_i^{0,V} = f_{\text{in}}^V(h_i^V), \ h_j^{0,W} = f_{\text{in}}^W(h_j^W), \ i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, n, \tag{2.2}$$

Then $\{h_i^{0,V}\}_{i=0}^m$ and $\{h_j^{0,W}\}_{j=0}^n$ are updated layer by layer via local updates as well as message passing. Let L be the number of message-passing layers. For $1 \leq l \leq L$, the hidden states at the l-th layer, $h_i^{l,V}, h_j^{l,W} \in \mathbb{R}^{d_l}$, can be computed from the the hidden states at the $(l-1)$-th layer via learnable functions $f_l^V, f_l^W : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$ and $g_l^V, g_l^W : \mathbb{R}^{d_{l-1}} \times \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$:

$$h_i^{l,V} = g_l^V\left(h_i^{l-1,V}, \sum_{j=1}^n E_{i,j} f_l^W(h_j^{l-1,W})\right), \quad i = 1, 2, \ldots, m, \tag{2.3}$$

$$h_j^{l,W} = g_l^W\left(h_j^{l-1,W}, \sum_{i=1}^m E_{i,j} f_l^V(h_i^{l-1,V})\right), \quad j = 1, 2, \ldots, n. \tag{2.4}$$

The output layer differs for scalar-output GNNs and vertex-output GNNs. The scalar output can be defined by:

$$y_{\text{out}} = f_{\text{out}}\left(\sum_{i=1}^{m} h_i^{L,V}, \sum_{j=1}^{n} h_j^{L,W}\right), \tag{2.5}$$

where $f_{\text{out}} : \mathbb{R}^{d_L} \times \mathbb{R}^{d_L} \to \mathbb{R}$ is some learnable function. For vertex-output GNNs, the output for vertex $w_j \in W$ is given by

$$y_{\text{out}}(w_i) = f_{\text{out}}^W\left(\sum_{i=1}^{m} h_i^{L,V}, \sum_{j=1}^{n} h_j^{L,W}, h_i^{L,W}\right), \quad i = 1, 2, \cdots, n, \tag{2.6}$$

where $f_{\text{out}}^W : \mathbb{R}^{d_L} \times \mathbb{R}^{d_L} \times \mathbb{R}^{d_L} \to \mathbb{R}$ is a learnable function. Throughout this paper, we require all learnable functions to be continuous following the same settings as in [5, 11], and we denote $\mathcal{F}_{\text{GNN}}$ and $\mathcal{F}_{\text{GNN}}^W$ as the function classes for the two types of GNNs. Particularly, we define

$$\begin{aligned} \mathcal{F}_{\text{GNN}} :=& \{F : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R} \mid F \text{ yields } (2.2), (2.3), (2.4), (2.5)\}, \\ \mathcal{F}_{\text{GNN}}^W :=& \{F : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R}^n \mid F \text{ yields } (2.2), (2.3), (2.4), (2.6)\}. \end{aligned} \tag{2.7}$$

2.3. **Mappings to represent MILP characteristics.** Now we introduce the mappings that are what we aim to approximate by GNNs. With the definitions, we will revisit questions (Q1-Q3) and describe them in a mathematically precise way.

Feasibility mapping. We first define the following mapping that indicates the feasibility of MILP:

$$\Phi_{\text{feas}} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \{0, 1\},$$

where $\Phi_{\text{feas}}(G, H) = 1$ if (G, H) corresponds to a feasible MILP and $\Phi_{\text{feas}}(G, H) = 0$ otherwise.

Optimal objective value mapping. We then define the following mapping that maps an MILP to its optimal objective value:

$$\Phi_{\text{obj}} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R} \cup \{\infty, -\infty\},$$

where $\Phi_{\text{obj}}(G, H) = \infty$ implies infeasibility and $\Phi_{\text{obj}}(G, H) = -\infty$ implies unboundedness. Note that the optimal objective value for MILP may be an infimum that can never be achieved. An example would be $\min_{x \in \mathbb{Z}^2} x_1 + \pi x_2$, s.t. $x_1 + \pi x_2 \geq \sqrt{2}$. The optimal objective value is $\sqrt{2}$, since for any $\epsilon > 0$, there exists a feasible x with $x_1 + \pi x_2 < \sqrt{2} + \epsilon$. However, there is no $x \in \mathbb{Z}^2$ such that $x_1 + \pi x_2 = \sqrt{2}$. Thus, the preimage $\Phi_{\text{obj}}^{-1}(\mathbb{R})$ cannot precisely describe all MILP instances with an optimal solution, it describes MILP problems with a finite optimal objective.

Optimal solution mapping. To give a well-defined optimal solution mapping is much more complicated [1] since the optimal objective value, as we discussed before, may never be achieved in some cases. To handle this issue, we only consider the case that any component in l or u must be finite, which implies that an optimal solution exists as long as the MILP problem is feasible.

[1]If we remove the integer constraints $N_I = \emptyset$ and let MILP reduces to linear programming (LP), the solution mapping will be easier to define. In this case, as long as the optimal objective value is finite, there must exist an optimal solution, and the optimal solution with the smallest ℓ_2-norm is unique [11]. Therefore, a mapping Φ_{solu}, which maps an LP to its optimal solution with the smallest ℓ_2-norm, is well defined on $\Phi_{\text{obj}}^{-1}(\mathbb{R})$.

More specifically, the vertex feature space is limited to $\widetilde{\mathcal{H}}_n^W = (\mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\{0,1\})^n \subset \mathcal{H}_n^W$ and we consider MILP problems taken from the following domain:

$$\mathcal{D}_{\text{solu}} = \left(\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \widetilde{\mathcal{H}}_n^W\right) \cap \Phi_{\text{feas}}^{-1}(1).$$

Note that $\Phi_{\text{feas}}^{-1}(1)$ describes the set of all feasible MILPs. Consequently, any MILP instance in $\mathcal{D}_{\text{solu}}$ admits at least one optimal solution. We can further define the following mapping which maps an MILP to exactly one of its optimal solutions:

$$\Phi_{\text{solu}} : \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}} \to \mathbb{R}^n,$$

where $\mathcal{D}_{\text{foldable}}$ is a subset of $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ that will be introduced in Section 4. The full definition of Φ_{solu} is placed in Appendix C due to its tediousness.

Invariance and equivariance. Now we discuss some properties of the three defined mappings. Mappings Φ_{feas} and Φ_{obj} are *permutation invariant* because the feasibility and optimal objective of an MILP would not change if the variables or constraints are reordered. We say the mapping Φ_{solu} is *permutation equivariant* because the solution of an MILP should be reordered consistently with the permutation on the variables. Now we define S_m as the group contains all permutations on the constraints of MILP and S_n as the group contains all permutations on the variables. For any $\sigma_V \in S_m$ and $\sigma_W \in S_n$, $(\sigma_V, \sigma_W) * (G, H)$ denotes the reordered MILP-graph with permutations σ_V, σ_W. It is clear that both Φ_{feas} and Φ_{obj} are permutation invariant in the following sense:

$$\Phi_{\text{feas}}((\sigma_V, \sigma_W) * (G, H)) = \Phi_{\text{feas}}(G, H),\ \ \Phi_{\text{obj}}((\sigma_V, \sigma_W) * (G, H)) = \Phi_{\text{obj}}(G, H),$$

for all $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$. In addition, Φ_{solu} is permutation equivariant in the following sense:

$$\Phi_{\text{solu}}((\sigma_V, \sigma_W) * (G, H)) = \sigma_W(\Phi_{\text{solu}}(G, H)),$$

for all $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H) \in \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}$. This will be discussed in Section C. Furthermore, one may check that any $F \in \mathcal{F}_{\text{GNN}}$ is invariant and any $F_W \in \mathcal{F}_{\text{GNN}}^W$ is equivariant.

Revisiting questions (Q1), (Q2) and (Q3). With the definitions above, questions (Q1),(Q2) and (Q3) actually ask: Given any finite set $\mathcal{D} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$, is there $F \in \mathcal{F}_{\text{GNN}}$ such that F well approximates Φ_{feas} or Φ_{obj} on set $\mathcal{D}$? Given any finite set $\mathcal{D} \subset \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}$, is there $F_W \in \mathcal{F}_{\text{GNN}}^W$ such that F_W is close to Φ_{solu} on set $\mathcal{D}$?

3. Directly Applying GNNs May Fail on General Datasets

In this section, we show a limitation of GNN to represent MILP. To well approximate the mapping Φ_{feas}, $\mathcal{F}_{\text{GNN}}$ should have *stronger separation power* than Φ_{feas}: for any two MILP instances $(G, H), (\hat{G}, \hat{H}) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$,

$$\Phi_{\text{feas}}(G, H) \neq \Phi_{\text{feas}}(\hat{G}, \hat{H}) \text{ implies } F(G, H) \neq F(\hat{G}, \hat{H}) \text{ for some } F \in \mathcal{F}_{\text{GNN}}.$$

In another word, as long as two MILP instances have different feasibility, there should be some GNNs that can detect that and give different outputs. Otherwise, we way that the whole GNN family $\mathcal{F}_{\text{GNN}}$ *cannot distinguish* two MILPs with different feasibility, hence, GNN cannot well approximate Φ_{feas}. This motivate us to study the separation power of GNN for MILP.

In the literature, the separation power of GNN is usually measured by so-called Weisfeiler-Lehman (WL) test [40]. We present a variant of WL test specially modified for MILP in Algorithm 1 that follows the same lines as in [11, Algorithm 1], where each vertex is labeled with a color. For example, $v_i \in V$ is initially labeled with $C_i^{0,V}$ based on its feature h_i^V by hash function $\text{HASH}_{0,\text{V}}$ that is assumed to be powerful enough such that it labels the vertices that have distinct information with distinct colors. After that, each vertex iteratively updates its color, based on its own color and information from its neighbors. Roughly speaking, *as long as two vertices in a graph are essentially different, they will get distinct colors finally.* The output of Algorithm 1 contains all vertex colors $\{\{C_i^{L,V}\}\}_{i=0}^m, \{\{C_j^{L,W}\}\}_{j=0}^n$, where $\{\{\}\}$ refers to a multiset in which the multiplicity of an element can be greater than one. Such approach is also named as color refinement [3, 4, 7].

Algorithm 1 WL test for MILP-Graphs[2](denoted by WL_{MILP})

Require: A graph instance $(G, H) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ and iteration limit $L > 0$.
1: Initialize with $C_i^{0,V} = \text{HASH}_{0,V}(h_i^V)$, $C_j^{0,W} = \text{HASH}_{0,W}(h_j^W)$.
2: **for** $l = 1, 2, \cdots, L$ **do**
3: $\quad C_i^{l,V} = \text{HASH}_{l,V}\left(C_i^{l-1,V}, \sum_{j=1}^n E_{i,j}\text{HASH}'_{l,W}\left(C_j^{l-1,W}\right)\right)$.
4: $\quad C_j^{l,W} = \text{HASH}_{l,W}\left(C_j^{l-1,W}, \sum_{i=1}^n E_{i,j}\text{HASH}'_{l,V}\left(C_i^{l-1,V}\right)\right)$.
5: **end for**
6: **return** The multisets containing all colors $\{\{C_i^{L,V}\}\}_{i=0}^m, \{\{C_j^{L,W}\}\}_{j=0}^n$.

Unfortunately, there exist some non-isomorphic graph pairs that WL test fail to distinguish [13]. Throughout this paper, we use $(G, H) \sim (\hat{G}, \hat{H})$ to denote that (G, H) and $(\hat{G}, \hat{H})$ *cannot be distinguished by the WL test*, i.e., $\text{WL}_{\text{MILP}}((G, H), L) = \text{WL}_{\text{MILP}}((\hat{G}, \hat{H}), L)$ holds for any $L \in \mathbb{N}$ and any hash functions. The following theorem indicates that $\mathcal{F}_{\text{GNN}}$ actually has the same separation power with the WL test.

Theorem 3.1 ([11, Theorem 4.2]). *For any $(G, H), (\hat{G}, \hat{H}) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$, it holds that $(G, H) \sim (\hat{G}, \hat{H})$ if and only if $F(G, H) = F(\hat{G}, \hat{H})$, $\forall\ F \in \mathcal{F}_{GNN}$.*

Theorem 3.1 is stated and proved in [11] for LP-graphs, but it actually also applies for MILP-graphs. We also remark that the equivalence between the separation powers of GNNs and WL test has been investigated in some earlier literature [5, 17, 41]. Unfortunately, the following lemma reveals that the separation power of WL test is weaker than Φ_{feas}, and, consequently, GNN has weaker separation power than Φ_{feas}, on some specific MILP datasets.

[2]In Algorithm 1, the hash functions $\{\text{HASH}_{l,V}, \text{HASH}_{l,W}\}_{l=0}^L$ can be any injective mappings defined on given domains, their output spaces consist of all possible vertex colors. The other hash functions $\{\text{HASH}'_{l,V}, \text{HASH}'_{l,W}\}_{l=0}^L$ are required to injectively map vertex colors to a linear space because we need to define sum and scalar multiplication on their outputs. Actually, Lemma 3.2 also applies on the WL test with a more general update scheme: $C_i^{l,V} = \text{HASH}_{l,V}(C_i^{l-1,V}, \{\{\text{HASH}'_{l,W}(C_j^{l-1,W}, E_{i,j})\}\}), C_j^{l,W} = \text{HASH}_{l,W}(C_j^{l-1,W}, \{\{\text{HASH}'_{l,V}(C_i^{l-1,V}, E_{i,j})\}\})$.

Lemma 3.2. *There exist two MILP problems (G, H) and $(\hat{G}, \hat{H})$ with one being feasible and the other one being infeasible, such that $(G, H) \sim (\hat{G}, \hat{H})$.*

Proof of Lemma 3.2. Consider two MILP provlems and their induced graphs:

$$\begin{aligned} \min_{x \in \mathbb{R}^6} \ & x_1 + x_2 + x_3 + x_4 + x_5 + x_6, \\ \text{s.t. } & x_1 + x_2 = 1,\ x_2 + x_3 = 1,\ x_3 + x_4 = 1, \\ & x_4 + x_5 = 1,\ x_5 + x_6 = 1,\ x_6 + x_1 = 1, \\ & 0 \le x_j \le 1,\ x_j \in \mathbb{Z},\ \forall\, j \in \{1, 2, \ldots, 6\}. \end{aligned}$$

$$\begin{aligned} \min_{x \in \mathbb{R}^6} \ & x_1 + x_2 + x_3 + x_4 + x_5 + x_6, \\ \text{s.t. } & x_1 + x_2 = 1,\ x_2 + x_3 = 1,\ x_3 + x_1 = 1, \\ & x_4 + x_5 = 1,\ x_5 + x_6 = 1,\ x_6 + x_4 = 1, \\ & 0 \le x_j \le 1,\ x_j \in \mathbb{Z},\ \forall\, j \in \{1, 2, \ldots, 6\}. \end{aligned}$$

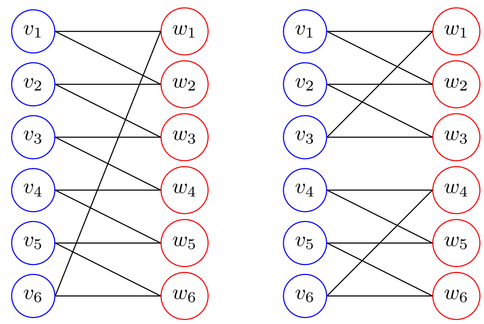


FIGURE 2. Two non-isomorphic MILP graphs that cannot be distinguished by WL test

The two MILP-graphs in Fugure 2 can not be distinguished by WL test, which can be proved by induction. First we consider the initial step in Algorithm 1. Based on the definitions in Section 2.2, we can explicitly write down the vertex features for each vertex here: $h_i^V = (1, =), 1 \le i \le 6$ and $h_j^W = (1, 0, 1, 1), 1 \le j \le 6$. Since all the vertices in V share the same information, they are labeled with an uniform color $C_1^{0,V} = C_2^{0,V} \cdots = C_6^{0,V}$ (We use blue in Figure 2), whatever the hash functions we choose. With the same argument, one would obtain $C_1^{0,W} = C_2^{0,W} \cdots = C_6^{0,W}$ and label all vertices in W with red. Both of the two graphs will be initialized in such an approach. Assuming $\{C_i^{l,V}\}_{i=1}^6$ are all blue and $\{C_j^{l,W}\}_{j=1}^6$ are all red, one will obtain *for both of the graphs in Figure 2* that $C_1^{l+1,V} = C_2^{l+1,V} \cdots = C_6^{l+1,V}$ and $C_1^{l+1,W} = C_2^{l+1,W} \cdots = C_6^{l+1,W}$ based on the update rule in Algorithm 1, because each blue vertex has two red neighbors and each red vertex has two blue neighbors, and each edge connecting a blue vertex with a red one has weight 1. This concludes that, one cannot distinguish the two graphs by checking the outputs of the WL test.

However, the first MILP problem is feasible, since $x = (0, 1, 0, 1, 0, 1)$ is a feasible solution, while the second MILP problem is infeasible, since the constraints imply $3 = 2(x_1 + x_2 + x_3) \in 2\mathbb{Z}$, which is a contradiction. □

4. UNFOLDABLE MILP PROBLEMS

We prove with example in Section 3 that one may not expect good performance of GNN to approximate Φ_{feas} on a general dataset of MILP problems. It's worth to ask: is it possible to describe the common characters of those examples that are not ideal? If so, one may restrict the dataset by removing such instances, and establish a strong separation/representation power of GNN on that restricted dataset. The following definition provides a rigorous description of such MILP instances.

Definition 4.1 (Foldable MILP)**.** *Given any MILP instance, one would obtain vertex colors* $\{C_i^{l,V}, C_j^{l,W}\}_{l,i,j}$ *by running Algorithm 1. We say that an MILP instance can be folded, or is foldable, if there exist* $1 \leq i, i' \leq m$ *or* $1 \leq j, j' \leq n$ *such that*

$$C_i^{l,V} = C_{i'}^{l,V}, \ i \neq i', \quad \text{or} \quad C_j^{l,W} = C_{j'}^{l,W}, \ j \neq j',$$

for any $l \in \mathbb{N}$ *and any hash functions. In another word, at least one color in the multisets generated by the WL test always has a multiplicity greater than* 1*. Furthermore, we denote* $\mathcal{D}_{foldable} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ *as the collection of all* $(G, H) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ *that can be folded.*

For example, the MILP in Figure 1 is not foldable, while the two MILP instances in Figure 2 are both foldable. The foldable examples in Figure 2 have been analyzed in the proof of Lemma 3.2, and now we provide some analysis of the example in Figure 1 here. Since the vertex features are distinct $h_1^W \neq h_2^W$, one would obtain $C_1^{0,W} \neq C_2^{0,W}$ as long as the hash function $\text{HASH}_{0,W}$ is injective. Although $h_1^V = h_2^V$ and hence $C_1^{0,V} = C_2^{0,V}$, the neighborhood information of v_1 and v_2 are different due to the difference of the edge weights. One could obtain $C_1^{1,V} \neq C_2^{1,V}$ by properly choosing $\text{HASH}_{0,W}, \text{HASH}'_{1,W}, \text{HASH}_{1,V}$, which concludes the unfoldability.

We prove that, as long as those foldable MILPs are removed, GNN is able to accurately predict the feasibility of all MILP instances in a dataset with finite samples. We use $(F(G,H) > 1/2)$ as the criteria, and the indicator $\mathbb{I}_{F(G,H)>1/2} = 1$ if $F(G,H) > 1/2$; $\mathbb{I}_{F(G,H)>1/2} = 0$ otherwise.

Theorem 4.2. *For any finite dataset* $\mathcal{D} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{foldable}$*, there exists* $F \in \mathcal{F}_{GNN}$ *such that*

$$\mathbb{I}_{F(G,H)>1/2} = \Phi_{feas}(G,H), \quad \forall \ (G,H) \in \mathcal{D}.$$

Similar results also hold for the optimal objective value mapping Φ_{obj} and the optimal solution mapping Φ_{solu} and we list the results below. All the proofs of theorems are deferred to the appendix.

Theorem 4.3. *Let* $\mathcal{D} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{foldable}$ *be a finite dataset. For any* $\delta > 0$*, there exists* $F_1 \in \mathcal{F}_{GNN}$ *such that*

$$\mathbb{I}_{F_1(G,H)>1/2} = \mathbb{I}_{\Phi_{obj}(G,H)\in\mathbb{R}}, \quad \forall \ (G,H) \in \mathcal{D},$$

and there exists some $F_2 \in \mathcal{F}_{GNN}$ *such that*

$$|F_2(G,H) - \Phi_{obj}(G,H)| < \delta, \quad \forall \ (G,H) \in \mathcal{D} \cap \Phi_{obj}^{-1}(\mathbb{R}).$$

Theorem 4.4. *Let* $\mathcal{D} \subset \mathcal{D}_{solu} \backslash \mathcal{D}_{foldable} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{foldable}$ *be a finite dataset. For any* $\delta > 0$*, there exists* $F_W \in \mathcal{F}_{GNN}^W$ *such that*

$$\|F(G,H) - \Phi_{solu}(G,H)\| < \delta, \quad \forall \ (G,H) \in \mathcal{D}.$$

Actually, conclusions in Theorems 4.2, 4.3 and 4.4 can be strengthened. The dataset $\mathcal{D}$ in the three theorems can be replaced with a measurable set with finite measure, which may contains infinitely many instances. Those strengthened theorems and their proofs can be found in Appendix A.

5. Symmetry Breaking Techniques via Random Features

Although we prove that GNN is able to approximate $\Phi_{\text{feas}}, \Phi_{\text{obj}}, \Phi_{\text{solu}}$ to any given precision for those unfoldable MILP instances, practitioners cannot benefit from that if there are foldable MILPs in their set of interest. To resolve this issue, we introduce a technique inspired by [1,34]. More specifically, we append the vertex features with an additional random feature ω and define a type of random GNNs as follows:

- Let $\Omega = [0,1]^m \times [0,1]^n$ and let $(\Omega, \mathcal{F}_\Omega, \mathbb{P})$ be the probability space corresponding to the uniform distribution $\mathcal{U}(\Omega)$.
- The class of random graph neural network $\mathcal{F}_{\text{GNN}}^R$ with scalar output is the collection of functions

$$\begin{aligned} F_R : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times \Omega &\to \qquad \mathbb{R}, \\ (G, H, \omega) \qquad &\mapsto F_R(G, H, \omega), \end{aligned}$$

  which is defined in the same way as $\mathcal{F}_{\text{GNN}}$ with input space being $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times \Omega \cong \mathcal{G}_{m,n} \times (\mathcal{H}^V \times [0,1])^m \times (\mathcal{H}^W \times [0,1])^n$ and ω being sampled from $\mathcal{U}(\Omega)$.
- The class of random graph neural network $\mathcal{F}_{\text{GNN}}^{W,R}$ with vector output is the collection of functions

$$\begin{aligned} F_{W,R} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times \Omega &\to \qquad \mathbb{R}^n, \\ (G, H, \omega) \qquad &\mapsto F_{W,R}(G, H, \omega), \end{aligned}$$

  which is defined in the same way as $\mathcal{F}_{\text{GNN}}^W$ with input space being $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times \Omega \cong \mathcal{G}_{m,n} \times (\mathcal{H}^V \times [0,1])^m \times (\mathcal{H}^W \times [0,1])^n$ and ω being sampled from $\mathcal{U}(\Omega)$.

We also write $F_R(G,H) = F_R(G,H,\omega)$ and $F_{W,R}(G,H) = F_{W,R}(G,H,\omega)$ as random variables. The theorem below states that, by adding random features, GNNs have sufficient power to represent MILP feasibility, even including those foldable MILPs. The intuition is that by appending additional random features, with probability one, each vertex will have distinct features and the resulting MILP-graph is hence unfoldable, even if it is foldable originally.

Theorem 5.1. *Let $\mathcal{D} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ be a finite dataset. For any $\epsilon > 0$, there exists $F_R \in \mathcal{F}_{GNN}^R$, such that*

$$\mathbb{P}\left(\mathbb{I}_{F_R(G,H)>1/2} \neq \Phi_{feas}(G,H)\right) < \epsilon, \quad \forall\ (G,H) \in \mathcal{D}.$$

Similar results also hold for the optimal objective value and the optimal solution.

Theorem 5.2. *Let $\mathcal{D} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ be a finite dataset. For any $\epsilon, \delta > 0$, there exists $F_{R,1} \in \mathcal{F}_{GNN}^R$, such that*

$$\mathbb{P}\left(\mathbb{I}_{F_{R,1}(G,H)>1/2} \neq \mathbb{I}_{\Phi_{obj}(G,H)\in\mathbb{R}}\right) < \epsilon, \quad \forall\ (G,H) \in \mathcal{D},$$

and there exists $F_{R,2} \in \mathcal{F}_{GNN}^{R}$*, such that*

$$\mathbb{P}\left(|F_{R,2}(G,H) - \Phi_{obj}(G,H)| > \delta\right) < \epsilon, \quad \forall\ (G,H) \in \mathcal{D} \cap \Phi_{obj}^{-1}(\mathbb{R}).$$

Theorem 5.3. *Let* $\mathcal{D} \subset \Phi_{obj}^{-1}(\mathbb{R}) \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ *be a finite dataset. For any* $\epsilon, \delta > 0$*, there exists* $F_{W,R} \in \mathcal{F}_{GNN}^{W,R}$*, such that*

$$\mathbb{P}\left(\|F_{W,R}(G,H) - \Phi_{solu}(G,H)\| > \delta\right) < \epsilon, \quad \forall\ (G,H) \in \mathcal{D}.$$

6. Numerical Experiments

In this section, we experimentally validate our theories on some small-scale examples with $m = 6$ and $n = 20$. We first randomly generate two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. Set $\mathcal{D}_1$ consists of 1000 randomly generate MILPs that are all unfoldable, and there are 419 feasible MILPs whose optimal solutions are attachable while the others are all infeasible. Set $\mathcal{D}_2$ consists of 1000 randomly generate MILPs that are all foldable and similar to the example provided in Figure 2, and there are 500 feasible MILPs with attachable optimal solution while the others are infeasible. We call SCIP [9, 10], a state-of-the-art non-commercial MILP solver, to obtain the feasibility and optimal solution for each instance. In our GNNs, we set the number of message-passing layers as $L = 2$ and parameterize all the learnable functions $f_{\text{in}}^V, f_{\text{in}}^W, f_{\text{out}}, f_{\text{out}}^W, \{f_l^V, f_l^W, g_l^V, g_l^W\}_{l=0}^L$ as multilayer perceptrons (MLPs). Our codes are modified from [16]. All the results reported in this section are obtained on the training sets, not an separate testing set, because generalization is the focus of this paper. Details of the numerical experiments can be found in the appendix.

Feasibility. We first test whether GNN can represent the feasibility of an MILP and report our results in Figure 3. The orange curve with tag "Foldable MILPs" presents the training result of GNN on set $\mathcal{D}_2$. It's clear that GNN fails to distinguish the feasible and infeasible MILP pairs that are foldable, *whatever the GNN size we take*. However, if we train GNNs on those unfoldable MILPs in set $\mathcal{D}_1$, it's clear that the rate of errors goes to zero, as long as the size of GNN is large enough (the number of GNN parameters is large enough). This result validates Theorem 4.2 and the first conclusion in Theorem 4.3: the existence of GNNs that can accurately predict whether an MILP is feasible (or whether an MILP has a finite optimal objective value). Finally, we append additional random features to the vertex features in GNN. As the green curve with tag "Foldable + Rand Feat." shown, GNN can perfectly fit the foldable data $\mathcal{D}_2$, which validates Theorem 5.1 and the first conclusion in Theorem 5.2.

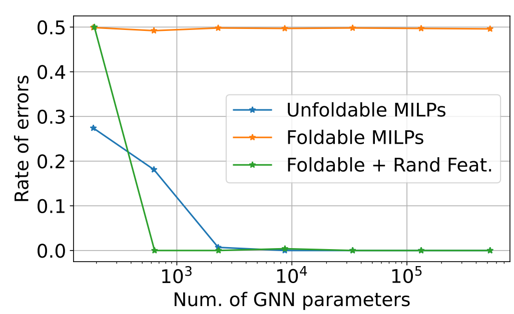
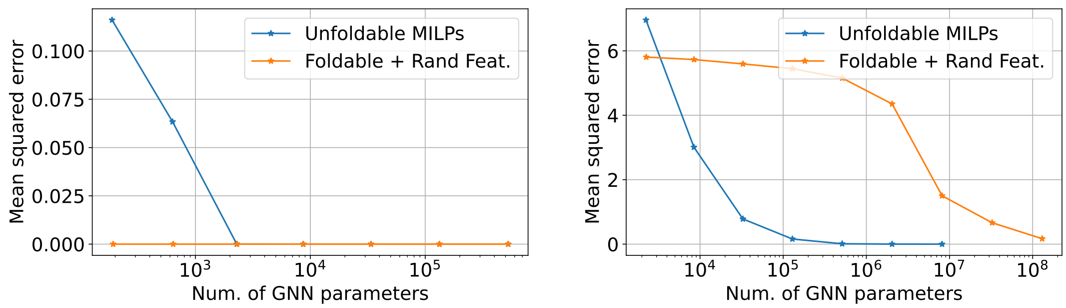


Figure 3. Feasibility

Optimal value and solution. Then we take the feasible instances from sets $\mathcal{D}_1$ and $\mathcal{D}_2$ and form new datasets $\mathcal{D}_1^{\text{feasible}}$ and $\mathcal{D}_2^{\text{feasible}}$, respectively. On $\mathcal{D}_1^{\text{feasible}}$ and $\mathcal{D}_2^{\text{feasible}}$, we validate that GNN is able to approximate the optimal objective value and one optimal solution. Figure 4a shows that, by restricting datasets to unfoldable instances, or by appending random features to the graph, one can train a GNN that has arbitrarily small approximation error for the optimal objective value. Such conclusions validates Theorems 4.3 and 5.2. Figure 4b shows that GNNs can even approximate an optimal solution of MILP, though it requires a much larger size than the case of approximating optimal objective. Theorems 4.4 and 5.3 are validated.



(A) Optimal objective value (B) Optimal solution

FIGURE 4. GNN can approximate Φ_{obj}, and Φ_{solu}

7. CONCLUSION

This work investigates the expressive power of graph neural networks for representing mixed-integer linear programming problems. It is found that the separation power of GNNs bounded by that of WL test is not sufficient for foldable MILP problems, and in contrast, we show that GNNs can approximate characteristics of unfoldable MILP problems with arbitrarily small error. To get rid of the requirement on unfoldability which may not be true in practice, a technique of appending random feature is discuss with theoretical guarantee. We conduct numerical experiments for all the theory. This paper will contribute to the recently active field of applying GNNs for MILP solvers.

REFERENCES

[1] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz, *The surprising power of graph neural networks with random node initialization*, arXiv preprint arXiv:2010.01179 (2020).

[2] Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel, *A supervised machine learning approach to variable branching in branch-and-bound*, In ecml, 2014.

[3] Vikraman Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky, *On the power of color refinement*, International symposium on fundamentals of computation theory, 2015, pp. 339–350.

[4] ______, *Graph isomorphism, color refinement, and compactness*, computational complexity **26** (2017), no. 3, 627–685.

[5] Waiss Azizian and Marc Lelarge, *Expressive power of invariant and equivariant graph neural networks*, International conference on learning representations, 2021.

[6] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost, *Machine learning for combinatorial optimization: a methodological tour d'horizon*, European Journal of Operational Research **290** (2021), no. 2, 405–421.

[7] Christoph Berkholz, Paul Bonsma, and Martin Grohe, *Tight lower and upper bounds for the complexity of canonical colour refinement*, Theory of Computing Systems **60** (2017), no. 4, 581–614.

[8] Timo Berthold, Matteo Francobaldi, and Gregor Hendel, *Learning to use local cuts*, arXiv preprint arXiv:2206.11618 (2022).

[9] 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 Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig, *The SCIP Optimization Suite 8.0*, Optimization Online, 2021.

[10] ———, *The SCIP Optimization Suite 8.0*, Technical Report 21-41, Zuse Institute Berlin, 2021.

[11] Ziang Chen, Jialin Liu, Xinshang Wang, Jianfeng Lu, and Wotao Yin, *On representing linear programs by graph neural networks*, arXiv preprint arXiv:2209.12288 (2022).

[12] Jian-Ya Ding, Chao Zhang, Lei Shen, Shengyin Li, Bing Wang, Yinghui Xu, and Le Song, *Accelerating primal solution findings for mixed integer programs based on solution prediction*, Proceedings of the aaai conference on artificial intelligence, 2020, pp. 1452–1459.

[13] Brendan L Douglas, *The weisfeiler-lehman method and graph isomorphism testing*, arXiv preprint arXiv:1101.5211 (2011).

[14] Lawrence C Evans and Ronald F Garzepy, *Measure theory and fine properties of functions*, Routledge, 2018.

[15] Christodoulos A Floudas and Xiaoxia Lin, *Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications*, Annals of Operations Research **139** (2005), no. 1, 131–162.

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

[17] Floris Geerts and Juan L Reutter, *Expressiveness and approximation properties of graph neural networks*, International conference on learning representations, 2022.

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

[19] Prateek Gupta, Elias B Khalil, Didier Chetélat, Maxime Gasse, Yoshua Bengio, Andrea Lodi, and M Pawan Kumar, *Lookback for learning to branch*, arXiv preprint arXiv:2206.14987 (2022).

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

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

[22] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina, *Learning to branch in mixed integer programming*, Proceedings of the aaai conference on artificial intelligence, 2016.

[23] Elias B Khalil, Christopher Morris, and Andrea Lodi, *Mip-gnn: A data-driven framework for guiding combinatorial solvers*, Update **2** (2022), x3.

[24] Diederik P Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2014).

[25] AH Land and AG Doig, *An automatic method of solving discrete programming problems*, Econometrica: Journal of the Econometric Society (1960), 497–520.

[26] Jiacheng Lin, Jialin Zhu, Huangang Wang, and Tao Zhang, *Learning to branch with tree-aware branching transformers*, Knowledge-Based Systems **252** (2022), 109455.

[27] Andreas Loukas, *What graph neural networks cannot learn: depth vs width*, International conference on learning representations, 2020.

[28] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki,

Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols, *Solving mixed integer programs using neural networks*, ArXiv **abs/2012.13349** (2020).

[29] Max B Paulus, Giulia Zarpellon, Andreas Krause, Laurent Charlin, and Chris Maddison, *Learning to cut by looking ahead: Cutting plane selection via imitation learning*, International conference on machine learning, 2022, pp. 17584–17600.

[30] Qingyu Qu, Xijun Li, Yunfan Zhou, Jia Zeng, Mingxuan Yuan, Jie Wang, Jinhu Lv, Kexin Liu, and Kun Mao, *An improved reinforcement learning algorithm for learning to branch*, arXiv preprint arXiv:2201.06213 (2022).

[31] Arthur Richards and Jonathan How, *Mixed-integer programming for control*, Proceedings of the 2005, american control conference, 2005., 2005, pp. 2676–2683.

[32] Walter Rudin, *Functional analysis*, 2nd ed., International series in pure and applied mathematics, McGraw-Hill, 1991.

[33] Ryoma Sato, Makoto Yamada, and Hisashi Kashima, *Approximation ratios of graph neural networks for combinatorial problems*, Advances in Neural Information Processing Systems **32** (2019).

[34] ______, *Random features strengthen graph neural networks*, Proceedings of the 2021 siam international conference on data mining (sdm), 2021, pp. 333–341.

[35] Lara Scavuzzo, Feng Yang Chen, Didier Chételat, Maxime Gasse, Andrea Lodi, Neil Yorke-Smith, and Karen Aardal, *Learning to branch with tree mdps*, arXiv preprint arXiv:2205.11107 (2022).

[36] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How, *Mixed integer programming for multi-vehicle path planning*, 2001 european control conference (ecc), 2001, pp. 2603–2608.

[37] Yunzhuang Shen, Yuan Sun, Andrew Eberhard, and Xiaodong Li, *Learning primal heuristics for mixed integer programs*, 2021 international joint conference on neural networks (ijcnn), 2021, pp. 1–8.

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

[39] Yunhao Tang, Shipra Agrawal, and Yuri Faenza, *Reinforcement learning for integer programming: Learning to cut*, International conference on machine learning, 2020, pp. 9367–9376.

[40] Boris Weisfeiler and Andrei Leman, *The reduction of a graph to canonical form and the algebra which appears therein*, NTI, Series **2** (1968), no. 9, 12–16.

[41] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, *How powerful are graph neural networks?*, International conference on learning representations, 2019.

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

[43] Giulia Zarpellon, Jason Jo, Andrea Lodi, and Yoshua Bengio, *Parameterizing branch-and-bound search trees to learn branching policies*, Proceedings of the aaai conference on artificial intelligence, 2021, pp. 3931–3939.

Appendix A. Proofs for Section 4

In this section, we prove the results in Section 4, i.e., graph neural networks can approximate Φ_{feas}, Φ_{obj}, and Φ_{solu}, on finite datasets with arbitrarily small error. We would consider more general results on finite-measure subset of $\mathcal{G}_{m,n}\times\mathcal{H}_m^V\times\mathcal{H}_n^W$ which may involve infinite elements, for which we need to define the topology and the measure on $\mathcal{G}_{m,n}\times\mathcal{H}_m^V\times\mathcal{H}_n^W$.

Topology and measure. The space $\mathcal{G}_{m,n}\times\mathcal{H}_m^V\times\mathcal{H}_n^W$ can be equipped with some natural topology and measure. In the construction of $\mathcal{G}_{m,n}\times\mathcal{H}_m^V\times\mathcal{H}_n^W$, let all Euclidean spaces be equipped with standard topology and the Lebesgue measure; let all discrete spaces be equipped with the discrete topology and the discrete measure with each elements being of measure 1. Then let all unions be disjoint unions and let all products induce the product topology and the product measure. Thus, the topology and the measure on $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ are defined. We use the notation $\text{Meas}(\cdot)$ for the measure on $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$.

Now we can state the following three theorems that can be viewed as generalized version of Theorem 4.2, 4.3, and 4.4.

Theorem A.1. *Let $X \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{foldable}$ be measurable with finite measure. For any $\epsilon > 0$, there exists some $F \in \mathcal{F}_{GNN}$, such that*

$$Meas\left(\left\{(G,H) \in X : \mathbb{I}_{F(G,H)>1/2} \neq \Phi_{feas}(G,H)\right\}\right) < \epsilon,$$

where $\mathbb{I}.$ is the indicator function, i.e., $\mathbb{I}_{F(G,H)>1/2} = 1$ if $F(G,H) > 1/2$ and $\mathbb{I}_{F(G,H)>1/2} = 0$ otherwise.

Theorem A.2. *Let $X \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{foldable}$ be measurable with finite measure. For any $\epsilon, \delta > 0$, the followings hold:*

(i) There exists some $F_1 \in \mathcal{F}_{GNN}$ such that

$$Meas\left(\left\{(G,H) \in X : \mathbb{I}_{F_1(G,H)>1/2} \neq \mathbb{I}_{\Phi_{obj}(G,H)\in\mathbb{R}}\right\}\right) < \epsilon.$$

(ii) There exists some $F_2 \in \mathcal{F}_{GNN}$ such that

$$Meas\left(\left\{(G,H) \in X \cap \Phi_{obj}^{-1}(\mathbb{R}) : |F_2(G,H) - \Phi_{obj}(G,H)| > \delta\right\}\right) < \epsilon.$$

Theorem A.3. *Let $X \subset \mathcal{D}_{solu} \backslash \mathcal{D}_{foldable} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{foldable}$ be measurable with finite measure. For any $\epsilon, \delta > 0$, there exists $F_W \in \mathcal{F}_{GNN}^W$ such that*

$$Meas\left(\{(G,H) \in X : \|F_W(G,H) - \Phi_{solu}(G,H)\| > \delta\}\right) < \epsilon.$$

The proof framework is similar to those in [11], and consists of two steps: i) show that measurability of the target mapping and apply Lusin's theorem to obtain a continuous mapping on a compact domain; ii) use Stone-Weierstrass-type theorem to show the uniform approximation result of graph neural networks.

We first prove that Φ_{feas} and Φ_{obj} are both measurable in the following two lemmas. The optimal solution mapping Φ_{solu} will be defined rigorously and proved as measurable in Section C.

Lemma A.4. *The feasibility mapping $\Phi_{feas} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \{0,1\}$ is measurable.*

Proof. It suffices to prove that $\Phi_{\text{feas}}^{-1}(1)$ is measurable, and the proof is almost the same as the that of [11]. The difference is that we should consider any fixed $\tau \in \{0,1\}^n$. Assuming $\tau = (0,\ldots,0,1,\ldots,1)$ where 0 and 1 appear for k and $n-k$ times respectively without loss of generality, we can replace $\mathbb{R}^n$ and $\mathbb{Q}^n$ in the proof of [11] by $\mathbb{R}^k \times \mathbb{Z}^{n-k}$ and $\mathbb{Q}^k \times \mathbb{Z}^{n-k}$ to get the proof in the MILP setting. □

Lemma A.5. *The optimal objective value mapping $\Phi_{obj} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R} \cup \{\infty, -\infty\}$ is measurable.*

Proof. Consider any $\phi \in \mathbb{R}$, $\Phi_{\text{obj}}(G,H) < \phi$ if and only if for the MILP problem associated to (G,H), there exists a feasible solution x and some $r \in \mathbb{N}_+$ such that $c^\top x \leq \phi - 1/r$. The rest of the proof can be done using the same techniques as in the proofs of [11], with the difference pointed in the proof of Lemma A.4. □

The Lusin's theorem, stated as follows, guarantees that any measurable function can be constricted on a compact such that only a small portion of domain is excluded and that the resulting function is continuous. The Lusin's theorem also applies for mappings defined on domains in $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$. This is because that $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ is isomorphic to the disjoint union of finitely many Euclidean spaces.

Theorem A.6 (Lusin's theorem [14, Theorem 1.14])**.** *Let μ be a Borel regular measure on $\mathbb{R}^n$ and let $f : \mathbb{R}^n \to \mathbb{R}^m$ be μ-measurable. Then for any μ-measurable $X \subset \mathbb{R}^n$ with $\mu(X) < \infty$ and any $\epsilon > 0$, there exists a compact set $E \subset X$ with $\mu(X \backslash E) < \epsilon$, such that $f|_E$ is continuous.*

The main tool in the proof of Theorem A.1 and Theorem A.2 is the universal approximation property of $\mathcal{F}_{\text{GNN}}$ stated below. Notice that the separation power of GNNs is the same as that of WL test. Theorem A.7 guarantees that GNNs can approximate any continuous function on a compact set whose separation power is less than or equal to that of the WL test, which can be proved using the Stone-Weierstrass theorem [32, Section 5.7].

Theorem A.7 ([11, Theorem 4.3])**.** *Let $X \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ be a compact set. For any $\Phi \in \mathcal{C}(X, \mathbb{R})$ satisfying*

$$\Phi(G,H) = \Phi(\hat{G},\hat{H}), \quad \forall\ (G,H) \sim (\hat{G},\hat{H}), \tag{A.1}$$

and any $\epsilon > 0$, there exists $F \in \mathcal{F}_{GNN}$ such that

$$\sup_{(G,H)\in X} |\Phi(G,H) - F(G,H)| < \epsilon.$$

Similar universal approximation results for GNNs and invariant target mappings can also be found in earlier literature; see e.g. [5,17]. With all the preparation above, we can then proceed to present the proof of Theorem A.1.

For applying Theorem A.7, one need to verify that the separation power of Φ_{feas} and Φ_{obj} are bounded by that of WL test for unfoldable MILP instances, i.e., the condition (A.1). For any $(G,H), (\hat{G},\hat{H}) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{\text{foldable}}$, since they are unfoldable, WL test should generate discrete coloring for them if there is no collision. If we further assume that they cannot be distinguished by WL test, then their stable discrete coloring must be identical (up to some permutation probably), which implies that they must be isomorphic. Therefore, we immediately obtain the following lemma:

Lemma A.8. *For any $(G,H), (\hat{G},\hat{H}) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{foldable}$, $(G,H) \sim (\hat{G},\hat{H})$ if and only if (G,H) and $(\hat{G},\hat{H})$ are isomorphic, i.e., $(\sigma_V, \sigma_W) * (G,H) = (\hat{G},\hat{H})$ for some $\sigma_V \in S_m$ and $\sigma_W \in S_n$.*

Now we can proceed to present the proof of Theorem A.1.

Proof of Theorem A.1. According to Lemma A.4, the mapping $\Phi_{\text{feas}} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \{0,1\}$ is measurable. By Lusin's theorem, there exists a compact $E \subset X \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{\text{foldable}}$ such that $\Phi_{\text{feas}}|_E$ is continuous and that $\text{Meas}(X \backslash E) < \epsilon$. For any $(G,H), (\hat{G},\hat{H}) \in E$, if

$(G, H) \sim (\hat{G}, \hat{H})$, Lemma A.8 guarantees that $\Phi_{\text{feas}}(G, H) = \Phi_{\text{feas}}(\hat{G}, \hat{H})$. Then using Theorem A.7, one can conclude that there exists $F \in \mathcal{F}_{\text{GNN}}$ with

$$\sup_{(G,H)\in E} |F(G, H) - \Phi_{\text{feas}}(G, H)| < \frac{1}{2}.$$

This implies that

$$\text{Meas}\left(\left\{(G, H) \in X : \mathbb{I}_{F(G,H)>1/2} \neq \Phi_{\text{feas}}(G, H)\right\}\right) \leq \text{Meas}(X \backslash E) < \epsilon.$$

□

Similar techniques can be used to prove a sequence of results:

Proof of Theorem 4.2. $\mathcal{D}$ is compact and $\Phi_{\text{feas}}|_{\mathcal{D}}$ is continuous. The rest of the proof follows the same argument as in the proof of Theorem A.1, using Lemma A.8 and Theorem A.7. □

Proof of Theorem A.2. Φ_{obj} is measurable by Lemma A.5, which implies that

$$\mathbb{I}_{\Phi_{\text{obj}}(\cdot,\cdot)\in\mathbb{R}} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \{0, 1\},$$

is also measurable. Then (i) and (ii) can be proved using the same argument as in the proof of Theorem A.1 for X and $\mathbb{I}_{\Phi_{\text{obj}}(\cdot,\cdot)\in\mathbb{R}}$, as well as $X \cap \Phi_{\text{obj}}^{-1}(\mathbb{R})$ and Φ_{obj}, respectively. □

Proof of Theorem 4.3. $\mathcal{D}$ is compact and $\mathbb{I}_{\Phi_{\text{obj}}(\cdot,\cdot)\in\mathbb{R}}$ restricted on $\mathcal{D}$ is continuous. In addition, as long as $\mathcal{D} \cap \Phi_{\text{obj}}^{-1}(\mathbb{R})$ is nonempty, it is compact and Φ_{obj} restricted on $\mathcal{D} \cap \Phi_{\text{obj}}^{-1}(\mathbb{R})$ is continuous. Then one can use Lemma A.8 and Theorem A.7 to obtain the desired results. □

Now we consider the optimal solution mapping Φ_{solu} and Theorem A.3, for proving which one needs a universal approximation result of $\mathcal{F}_{\text{GNN}}^W$ for equivariant functions. For stating the theorem rigorously, we define another equivalence relationship on $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$: $(G, H) \stackrel{W}{\sim} (\hat{G}, \hat{H})$ if $(G, H) \sim (\hat{G}, \hat{H})$ and in addition, $C_j^{L,W} = \hat{C}_j^{L,W}$, $\forall\, j \in \{1, 2, \ldots, n\}$, for any $L \in \mathbb{N}$ and any hash functions.

The universal approximation result of $\mathcal{F}_{\text{GNN}}^W$ is stated as follows, which guarantees that $\mathcal{F}_{\text{GNN}}^W$ can approximate any continuous equivariant function on a compact set whose separation power is less than or equal to that of the WL test, and is based on a generalized Stone-Weierstrass theorem established in [5].

Theorem A.9 ([11, Theorem E.1])**.** *Let $X \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$ be a compact subset that is closed under the action of $S_m \times S_n$. Suppose that $\Phi \in \mathcal{C}(X, \mathbb{R}^n)$ satisfies the followings:*

(i) For any $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H) \in X$, it holds that

$$\Phi\left((\sigma_V, \sigma_W) * (G, H)\right) = \sigma_W(\Phi(G, H)).$$

(ii) $\Phi(G, H) = \Phi(\hat{G}, \hat{H})$ holds for all $(G, H), (\hat{G}, \hat{H}) \in X$ with $(G, H) \stackrel{W}{\sim} (\hat{G}, \hat{H})$.

(iii) Given any $(G, H) \in X$ and any $j, j' \in \{1, 2, \ldots, n\}$, if $C_j^{l,W} = C_{j'}^{l,W}$ holds for any $l \in \mathbb{N}$ and any choices of hash functions, then $\Phi(G, H)_j = \Phi(G, H)_{j'}$.

Then for any $\epsilon > 0$, there exists $F_W \in \mathcal{F}_{GNN}^W$ such that

$$\sup_{(G,H)\in X} \|\Phi(G, H) - F_W(G, H)\| < \epsilon.$$

We refer to [5, 17] for other results on the closure of equivariant GNNs. Now we can prove Theorem A.3 and Corollary 4.4.

Proof of Theorem A.3. We can assume that $X \subset \mathcal{D}_{\text{solu}}\backslash\mathcal{D}_{\text{foldable}} \subset \mathcal{G}_{m,n}\times\mathcal{H}_m^V\times\mathcal{H}_n^W\backslash\mathcal{D}_{\text{foldable}}$ is close under the action of $S_m\times S_n$, since otherwise one can consider $\bigcup_{(\sigma_V,\sigma_W)\in S_m\times S_n}(\sigma_V,\sigma_W)*X$ instead of X. The solution mapping $\Phi_{\text{solu}}:\mathcal{D}_{\text{solu}}\backslash\mathcal{D}_{\text{foldable}}$ is measurable by Lemma C.5. According to Lusin's theorem, there exists a compact $E'\subset X\subset\mathcal{D}_{\text{solu}}\backslash\mathcal{D}_{\text{foldable}}$ with $\text{Meas}(X\backslash E')<\epsilon/|S_m\times S_n|$ such that $\Phi_{\text{solu}}|_{E'}$ is continuous. Let us set

$$E=\bigcap_{(\sigma_V,\sigma_W)\in S_m\times S_n}(\sigma_V,\sigma_W)*E'\subset X,$$

which is compact and is closed under the action of $S_m\times S_n$. Furthermore, it holds that

$$\begin{aligned}\text{Meas}(X\backslash E)&=\text{Meas}\left(X\backslash\bigcap_{(\sigma_V,\sigma_W)\in S_m\times S_n}(\sigma_V,\sigma_W)*E'\right)\\&\leq\sum_{(\sigma_V,\sigma_W)\in S_m\times S_n}\text{Meas}(X\backslash(\sigma_V,\sigma_W)*E')\\&=|S_m\times S_n|\cdot\text{Meas}(X\backslash E')\\&<\epsilon.\end{aligned}$$

For any $(G,H),(\hat{G},\hat{H})\in E$, if $(G,H)\overset{W}{\sim}(\hat{G},\hat{H})$, similar to Lemma A.8, we know that there exists $\sigma_V\in S_m$ such that $(\sigma_V,\text{Id})*(G,H)=(\hat{G},\hat{H})$. Then by the construction of Φ_{solu} in Section C, one has that $\Phi_{\text{solu}}(G,H)=\Phi_{\text{solu}}(\hat{G},\hat{H})\in\mathbb{R}^n$. Condition (i) in Theorem A.9 is satisfied by the definition of Φ_{solu} (see Section C) and Condition (iii) in Theorem A.9 follows from the fact that WL test yields discrete coloring for any graph in $\mathcal{G}_{m,n}\times\mathcal{H}_m^V\times\mathcal{H}_n^W\backslash\mathcal{D}_{\text{foldable}}$. Then applying Theorem A.9, one can conclude that for any $\delta>0$, there exists $F_W\in\mathcal{F}_{\text{GNN}}^W$ with

$$\sup_{(G,H)\in E}\|F_W(G,H)-\Phi_{\text{solu}}(G,H)\|<\delta.$$

Therefore, it holds that

$$\text{Meas}\left(\{(G,H)\in X:\|F_W(G,H)-\Phi_{\text{solu}}(G,H)\|>\delta\}\right)\leq\text{Meas}(X\backslash E)<\epsilon,$$

which completes the proof. □

Proof of Theorem 4.4. Without loss of generality, we can assume that the finite dataset $\mathcal{D}\subset\mathcal{D}_{\text{solu}}\backslash\mathcal{D}_{\text{foldable}}$ is closed under the action of $S_m\times S_n$. Otherwise, we can replace $\mathcal{D}$ by a larger dataset $\bigcup_{(\sigma_V,\sigma_W)\in S_m\times S_n}(\sigma_V,\sigma_W)*\mathcal{D}$.

Note that $\mathcal{D}$ is compact and $\Phi_{\text{solu}}|_{\mathcal{D}}$ is continuous. The rest of the proof can be done by using similar techniques as in the proof of Theorem A.3, with Theorem A.9. □

Appendix B. Proofs for Section 5

We collect the proof of Theorem 5.1, Theorem 5.2, and Theorem 5.3 in this section. Before we proceed, we remark that one can also implement the WL test, i.e., Algorithm 1, for $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times \Omega \cong \mathcal{G}_{m,n} \times (\mathcal{H}^V \times [0,1])^m \times (\mathcal{H}^W \times [0,1])^n$, and the equivalence relationships $\sim$ and $\overset{W}{\sim}$ are defined in the same way as in $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W$. In addition, Theorem A.7 and Theorem A.9 also hold for $\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times \Omega \cong \mathcal{G}_{m,n} \times (\mathcal{H}^V \times [0,1])^m \times (\mathcal{H}^W \times [0,1])^n$.

Proof of Theorem 5.1. Define

$$\begin{aligned}\Omega_F = \big\{\omega = (\omega_1^V, \ldots, \omega_m^V, \omega_1^W, \ldots, \omega_n^W) \in \Omega :\exists\ i \neq i' \in \{1,2,\ldots,m\},\ \text{s.t. } \omega_i^V = \omega_{i'}^V,\\ \text{or } \exists\ j \neq j' \in \{1,2,\ldots,n\},\ \text{s.t. } \omega_j^W = \omega_{j'}^W\big\}.\end{aligned} \tag{B.1}$$

One can see that $\mathbb{P}(\Omega_F) = 0$ and that for any $(G,H,\omega) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times (\Omega \backslash \Omega_F)$, different vertices in V or W are equipped with different vertex features. Therefore, similar to Lemma A.8, one can see for any $(G,H,\omega), (\hat{G},\hat{H},\hat{\omega}) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times (\Omega\backslash\Omega_F)$ that $(G,H,\omega) \sim (\hat{G},\hat{H},\hat{\omega})$ if and only if $(\sigma_V,\sigma_W) * (G,H,\omega) = (\hat{G},\hat{H},\hat{\omega})$ for some $\sigma_V \in S_m$ and $\sigma_W \in S_n$. This implies that

$$\Phi_{\text{feas}}(G,H) = \Phi_{\text{feas}}(\hat{G},\hat{H}), \quad \forall\ (G,H,\omega) \sim (\hat{G},\hat{H},\hat{\omega}) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \times (\Omega\backslash\Omega_F).$$

For any $\epsilon > 0$, there exists a compact subset $\Omega_\epsilon \subset \Omega\backslash\Omega_F$ such that

$$\mathbb{P}(\Omega\backslash\Omega_\epsilon) = \mathbb{P}\left((\Omega\backslash\Omega_F)\backslash\Omega_\epsilon\right) < \epsilon.$$

Note that $\mathcal{D}\times\Omega_\epsilon$ is also compact and that Φ_{feas} is continuous on $\mathcal{D}\times\Omega_\epsilon$. Applying Theorem A.7 for Φ_{feas} and $\mathcal{D} \times \Omega_\epsilon \subset \mathcal{G}_{m,n} \times (\mathcal{H}^V \times [0,1])^m \times (\mathcal{H}^W \times [0,1])^n$, one can conclude the existence of $F^R \in \mathcal{F}_{\text{GNN}}^R$ with

$$\sup_{(G,H,\omega)\in\mathcal{D}\times\Omega_\epsilon} |F_R(G,H,\omega) - \Phi_{\text{feas}}(G,H)| < \frac{1}{2}.$$

It thus holds for any $(G,H) \in \mathcal{D}$ that

$$\mathbb{P}\left(\mathbb{I}_{F_R(G,H)>1/2} \neq \Phi_{\text{feas}}(G,H)\right) \leq \mathbb{P}(\Omega\backslash\Omega_\epsilon) < \epsilon.$$

□

Proof of Theorem 5.2. The results can be obtained by applying similar techniques as in the proof of Theorem 5.1 for $\mathbb{I}_{\Phi_{\text{obj}}(\cdot,\cdot)\in\mathbb{R}}$ and Φ_{obj}. □

Proof of Theorem 5.3. Without loss of generality, we can assume that $\mathcal{D}$ is closed under the action of $S_m \times S_n$; otherwise, we can use $\{(\sigma_V,\sigma_W) * (G,H) : (G,H) \in \mathcal{D}, \sigma_V \in S_m, \sigma_W \in S_n\}$ instead of $\mathcal{D}$. Let $\Omega_F \subset \Omega$ be the set defined in (B.1) which is clearly closed under the action of $S_m \times S_n$. There exists a compact $\Omega'_\epsilon \subset \Omega\backslash\Omega_F$ with $\mathbb{P}(\Omega\backslash\Omega'_\epsilon) = \mathbb{P}((\Omega\backslash\Omega_F)\backslash\Omega'_\epsilon) < \epsilon/|S_m \times S_n|$. Define

$$\Omega_\epsilon = \bigcap_{(\sigma_V,\sigma_W)\in S_m\times S_n} (\sigma_V,\sigma_W) * \Omega'_\epsilon \subset \Omega\backslash\Omega_F,$$

which is compact and closed under the action of $S_m \times S_n$. One has that

$$\begin{aligned}
\mathbb{P}(\Omega \backslash \Omega_\epsilon) &\leq \mathbb{P}\left(\Omega \backslash \bigcap_{(\sigma_V,\sigma_W)\in S_m\times S_n} (\sigma_V,\sigma_W) * \Omega'_\epsilon\right) \\
&\leq \sum_{(\sigma_V,\sigma_W)\in S_m\times S_n} \mathbb{P}(\Omega \backslash (\sigma_V,\sigma_W) * \Omega'_\epsilon) \\
&= |S_m \times S_n| \cdot \mathbb{P}(\Omega \backslash \Omega'_\epsilon) \\
&< \epsilon.
\end{aligned}$$

In addition, $\mathcal{D} \times \Omega_\epsilon$ is compact in $\mathcal{G}_{m,n} \times (\mathcal{H}^V \times [0,1])^m \times (\mathcal{H}^W \times [0,1])^n$ and is closed under the action of $S_m \times S_n$.

We then verify the three conditions in Theorem A.9 for Φ_{solu} and $\mathcal{D} \times \Omega_\epsilon$. Condition (i) holds automatically by the definition of the optimal solution mapping. For Condition (ii), given any (G,H,ω) and $(\hat{G},\hat{H},\hat{\omega})$ in $\mathcal{D} \times \Omega_\epsilon$ with $(G,H,\omega) \stackrel{W}{\sim} (\hat{G},\hat{H},\hat{\omega})$, since graphs with vertex features in $\mathcal{D} \times \Omega_\epsilon \subset \mathcal{G}_{m,n} \times \mathcal{H}_n^V \times \mathcal{H}_n^W \times (\Omega \backslash \Omega_F)$ cannot be folded, similar to Lemma A.8, one can conclude that $(\hat{G},\hat{H},\hat{\omega}) = (\sigma_V, \text{Id}) * (G,H,\omega)$ for some $\sigma_V \in S_n$, which leads to $\Phi_{\text{solu}}(G,H) = \Phi_{\text{solu}}(\hat{G},\hat{H})$. Condition (iii) also follows from the fact that any graph in $\mathcal{D} \times \Omega_\epsilon$ cannot be folded and WL test yields discrete coloring. Therefore, Theorem A.9 applies and there exists $F_{W,R} \in \mathcal{F}_{\text{GNN}}^{W,R}$ such that

$$\sup_{(G,H,\omega)\in\mathcal{D}\times\Omega_\epsilon} \|F_{W,R}(G,H,\omega) - \Phi_{\text{solu}}(G,H)\| < \delta,$$

which implies that

$$\mathbb{P}\left(\|F_{W,R}(G,H) - \Phi_{\text{solu}}(G,H)\| > \delta\right) \leq \mathbb{R}(\Omega \backslash \Omega_\epsilon) < \epsilon, \quad \forall\ (G,H) \in \mathcal{D}.$$

□

Remark B.1. *A more deeper observation is that, Theorem 5.1, Theorem 5.2, and Theorem A.2 are actually true even if we only allow one message-passing layer in the GNN structure. This is because that the separation power of GNNs with one message-passing layer is the same as the WL test with one iteration (see* [11, Appendix C]*), and that one iteration in WL test suffices to yield a discrete coloring for MILP-graphs in $\mathcal{G}_{m,n} \times \mathcal{H}_n^V \times \mathcal{H}_n^W \times (\Omega \backslash \Omega_F)$.*

Appendix C. The Optimal Solution Mapping Φ_{solu}

In this section, we define the equivariant optimal solution mapping Φ_{solu}, and prove the measurability. The definition consists of several steps.

C.1. **The sorting mapping.** We first define a sorting mapping:

$$\Phi_{\text{sort}} : \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{\text{foldable}} \to S_n,$$

that returns a permutation on $\{w_1, w_2, \ldots, w_n\}$, $\mathcal{D}_{\text{foldable}}$ is the collection of all foldable MILP problem as in Definition 4.1. This can be done via an order refinement procedure, similar to the WL test. The initial order as well as the order refinement are defined in the following several definitions.

Definition C.1. *We define total order on $\mathcal{H}^V$ and $\mathcal{H}^W$ using lexicographic order:*

(i) For any $(b_i, \circ_i), (b_{i'}, \circ_{i'}) \in \mathcal{H}^V = \mathbb{R} \times \{\leq, =, \geq\}$, we say $(b_i, \circ_i) < (b_{i'}, \circ_{i'})$ if one of the followings holds:
  - $b_i < b_{i'}$.
  - $b_i = b_{i'}$ *and* $\iota(\circ_i) < \iota(\circ_{i'})$, *where* $\iota(\leq) = -1$, $\iota(=) = 0$, *and* $\iota(\geq) = 1$.

(ii) For any $(c_j, l_j, u_j, \tau_j), (c_{j'}, l_{j'}, u_{j'}, \tau_{j'}) \in \mathcal{H}^W = \mathbb{R} \times (\mathbb{R} \cup \{-\infty\}) \times (\mathbb{R} \cup \{+\infty\}) \times \{0, 1\}$, we say $(c_j, l_j, u_j, \tau_j) < (c_{j'}, l_{j'}, u_{j'}, \tau_{j'})$ if one of the followings holds:
  - $c_j < c_{j'}$.
  - $c_j = c_{j'}$ *and* $l_j < l_{j'}$.
  - $c_j = c_{j'}$, $l_j = l_{j'}$, *and* $u_j = u_{j'}$.
  - $c_j = c_{j'}$, $l_j = l_{j'}$, $u_j = u_{j'}$, *and* $\tau_j = \tau_{j'}$.

Definition C.2. *Let $X = \{\{x_1, x_2, \ldots, x_k\}\}$ and $X' = \{\{x'_1, x'_2, \ldots, x'_{k'}\}\}$ be two multisets whose elements are taken from the same totally-ordered set. Then we say $X \leq X'$ if $x_{\sigma(1)} x_{\sigma(2)} \cdots x_{\sigma(k)} \leq x'_{\sigma'(1)} x'_{\sigma'(2)} \cdots x'_{\sigma'(k')}$ in the sense of lexicographical order, where $\sigma \in S_k$ and $\sigma' \in S_{k'}$ are permutations such that $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \cdots \leq x_{\sigma(k)}$ and $x'_{\sigma'(1)} \leq x'_{\sigma'(2)} \leq \cdots \leq x'_{\sigma'(k')}$.*

Definition C.3. *Suppose that $V = \{v_1, v_2, \ldots, v_n\}$ and $W = \{w_1, w_2, \ldots, w_n\}$ are already ordered, and that $E \in \mathbb{R}^{m \times n}$. The order refinement is defined lexicographicaly:*

(i) For any $i, i' \in \{1, 2, \ldots, m\}$, we say that

$$(v_i, \{\{(E_{i,j}, w_j) : E_{i,j} \neq 0\}\}) < (v_{i'}, \{\{(E_{i',j}, w_j) : E_{i',j} \neq 0\}\}),$$

if one of the followings holds:
  - $v_i < v_{i'}$.
  - $v_i = v_{i'}$ *and* $\{(E_{i,j}, w_j) : E_{i,j} \neq 0\} < \{(E_{i',j}, w_j) : E_{i',j} \neq 0\}$ *in the sense of Definition C.2 where* $(E_{i,j}, w_j) < (E_{i',j'}, w_{j'})$ *if and only if* $E_{i,j} < E_{i',j'}$ *or* $E_{i,j} = E_{i',j'}$, $w_j < w_{j'}$.

(ii) For any $j, j' \in \{1, 2, \ldots, n\}$, we say that

$$(w_j, \{\{(E_{i,j}, v_i) : E_{i,j} \neq 0\}\}) < (w_{j'}, \{\{(E_{i,j'}, v_i) : E_{i,j'} \neq 0\}\}),$$

if one of the followings holds:
  - $w_j < w_{j'}$.
  - $w_j = w_{j'}$ *and* $\{(E_{i,j}, v_i) : E_{i,j} \neq 0\} < \{(E_{i,j'}, v_i) : E_{i,j'} \neq 0\}$ *in the sense of Definition C.2 where* $(E_{i,j}, v_i) < (E_{i',j'}, v_{i'})$ *if and only if* $E_{i,j} < E_{i',j'}$ *or* $E_{i,j} = E_{i',j'}$, $v_i < v_{i'}$.

With the preparations above, we can now define Φ_{sort} in Algorithm 2.

Remark C.4. *The output of Algorithm 2 is well-defined and is unique. This is because that we use unfoldable (G, H) as input. Note that the order refinement in Definition C.3 is more strict than the color refinement in WL test. Therefore, after $m + n$ iterations, there is no $j \neq j' \in \{1, 2, \ldots, n\}$ with $w_j = w_{j'}$, since the WL test returns discrete coloring if no collisions for $(G, H) \notin \mathcal{D}_{foldable}$.*

Algorithm 2 The sorting mapping Φ_{sort}

Require: A weighted bipartite graph $G = (V \cup W, E) \in \mathcal{G}_{m,n}$, with vertex features $H = (h_1^V, h_2^V, \ldots, h_m^V, h_1^W, h_2^W, \ldots, h_n^W) \in \mathcal{H}_m^V \times \mathcal{H}_n^W$ such that $(G, H) \notin \mathcal{D}_{\text{foldable}}$.

1: Order V and W according to Definition C.1.
2: **for** $l = 1 : m + n$ **do**
3: Refine the ordering on V and W according to Definition C.3.
4: **end for**
5: Return $\sigma_W \in S_n$ such that $w_{\sigma_W(1)} < w_{\sigma_W(2)} < \cdots < w_{\sigma_W(n)}$.

The sorting mapping Φ_{sort} has some straightforward properties:

- Φ_{sort} is equivariant:
$$\Phi_{\text{sort}}((\sigma_V, \sigma_W) * (G, H)) = (\sigma_V, \sigma_W) * \Phi_{\text{sort}}(G, H) = \sigma_W \circ \Phi_{\text{sort}}(G, H),$$
for any $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{\text{foldable}}$. This is due to the fact that Definition C.3 defines the total order and its refinement solely depending on the vertex features that are independent of the input order.
- Φ_{sort} is measurable, where the range S_n is equipped with the discrete measure. This is because that Φ_{sort} is defined via finitely many comparisons.

C.2. **The optimal solution mapping.** We then define the optimal solution mapping
$$\Phi_{\text{solu}} : \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}} \to \mathbb{R}^n,$$
based on Φ_{sort}, where
$$\mathcal{D}_{\text{solu}} = \left(\mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \widetilde{\mathcal{H}}_n^W\right) \cap \Phi_{\text{feas}}^{-1}(1),$$
with $\widetilde{\mathcal{H}}_n^W = (\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \{0,1\})^n \subset \mathcal{H}_n^W$, is the collection of all feasible MILP problems in which every component in l and u is finite.

We have mentioned before that any MILP problem in $\mathcal{D}_{\text{solu}}$ admits at least one optimal solution. For any $(G, H) \in \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}$, we denote $X_{\text{solu}}(G, H) \in \mathbb{R}^n$ as the set of optimal solutions to the MILP problem associated to (G, H). One can see that $X_{\text{solu}}(G, H)$ is compact since for every $(G, H) \in \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}$, both l and u are finite, which leads to the boundedness (and hence the compactness) of $X_{\text{solu}}(G, H)$.

Given any permutation $\sigma \in S_n$, let us define a total order on $\mathbb{R}^d$: $x \overset{\sigma}{\prec} x'$ if and only if $x_{\sigma(1)} x_{\sigma(2)} \cdots x_{\sigma(n)} < x'_{\sigma(1)} x'_{\sigma(2)} \cdots x'_{\sigma(n)}$ in the sense of lexicographic order. For any $(G, H) \in \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \backslash \mathcal{D}_{\text{foldable}}$, as in Section C.1, the permutation $\Phi_{\text{sort}}(G, H) \in S_n$ is well-defined. Then we define $\Phi_{\text{solu}}(G, H)$ is the smallest element in $X_{\text{solu}}(G, H)$ with respect to the order $\overset{\sigma}{\prec}$, where $\sigma = \Phi_{\text{sort}}(G, H)$. The existence and the uniqueness are true since $X_{\text{solu}}(G, H)$ is compact. More explicitly, the components of $\Phi_{\text{solu}}(G, H)$ can be determined recursively:
$$\Phi_{\text{solu}}(G, H)_{\sigma(1)} = \inf \left\{ x_{\sigma(1)} : x \in X_{\text{solu}}(G, H) \right\},$$
and
$$\Phi_{\text{solu}}(G, H)_{\sigma(j)} = \inf \left\{ x_{\sigma(j)} : x \in X_{\text{solu}}(G, H),\ x_{\sigma(j')} = \Phi_{\text{solu}}(G, H)_{\sigma(j)},\ j' = 1, 2, \ldots, j \right\},$$

for $j = 2, 3, \dots, n$. It follows from the equivariance of $\Phi_{\text{sort}}(G, H)$ and $X_{\text{solu}}(G, H)$ that $\Phi_{\text{solu}}(G, H)$ is also equivariant, i.e.,

$$\Phi_{\text{solu}}((\sigma_V, \sigma_W) * (G, H)) = \sigma_W(\Phi_{\text{solu}}(G, H)), \quad \forall\, \sigma_V \in S_m, \sigma_W \in S_n, (G, H) \in \mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}.$$

We then show the measurability of Φ_{solu}.

Lemma C.5. *The optimal solution mapping $\Phi_{solu} : \mathcal{D}_{solu} \backslash \mathcal{D}_{foldable} \to \mathbb{R}^n$ is measurable.*

Proof. It suffices to show that for any fixed $\circ \in \{\le, =, \ge\}^m$, $\tau \in \{0, 1\}^n$, and $\sigma \in S_n$, the mapping

$$\begin{aligned} \Phi_j : \iota^{-1}(\mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}) \cap (\Phi_{\text{sort}} \circ \iota)^{-1}(\sigma) \to &\quad \mathbb{R}, \\ (A, b, c, l, u) &\mapsto \Phi_{\text{solu}}(\iota(A, b, c, l, u))_{\sigma(j)}, \end{aligned}$$

is measurable for all $j \in \{1, 2, \dots, n\}$, where

$$\iota : \mathbb{R}^{m \times n} \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W,$$

is the embedding map when $\circ$ and τ are fixed. Without loss of generality, we assume that $\circ = \{\le, \dots, \le, =, \dots, =, \ge, \dots, \ge\}$ where $\le$, $=$, and $\ge$ appear for k_1, $k_2 - k_1$, and $m - k_2$ times, respectively, and that $\tau = (0, \dots, 0, 1, \dots, 1)$ where 0 and 1 appear for k and $n - k$ times, respectively. Note that the domain of Φ_j, i.e., $\iota^{-1}(\mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}) \cap (\Phi_{\text{sort}} \circ \iota)^{-1}(\sigma)$ is measurable due to the measurability of Φ_{sort}.

Define

$$\begin{aligned} V_{\text{feas}}(A, b, c, l, u, x) = \max \Bigg\{ &\max_{1 \le i \le k_1} \left(\sum_{j'=1}^n A_{i,j'} x_{j'} - b_i \right)_+, \max_{k_1 < i \le k_2} \left| \sum_{j'=1}^n A_{i,j'} x_{j'} - b_i \right|, \\ &\max_{k_2 < i \le m} \left(b_i - \sum_{j'=1}^n A_{i,j'} x_{j'} \right)_+, \max_{1 \le j' \le n} (l_{j'} - x_{j'})_+, \max_{1 \le j' \le n} (x_{j'} - u_{j'})_+ \Bigg\}, \end{aligned}$$

and

$$V_{\text{solu}}(A, b, c, l, u, x) = \max \left\{ \left(c^\top x - \Phi_{\text{obj}}(\iota(A, b, c, l, u)) \right)_+, V_{\text{feas}}(A, b, c, l, u, x) \right\},$$

for $(A, b, c, l, u) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$ and $x \in \mathbb{R}^k \times \mathbb{Z}^{n-k}$. It is clear that V_{feas} is continuous and that $x \in \mathbb{R}^k \times \mathbb{Z}^{n-k}$ is an optimal solution to the problem $\iota(A, b, c, l, u)$ if and only if $V_{\text{solu}}(A, b, c, l, u, x) = 0$. In addition, V_{solu} is measurable by the measurability (see Lemma A.5), and is continuous with respect to x.

Then we proceed to prove that Φ_j is measurable by induction. We first consider the case that $j = 1$. For any $(A, b, c, l, u) \in \iota^{-1}(\mathcal{D}_{\text{solu}} \backslash \mathcal{D}_{\text{foldable}}) \cap (\Phi_{\text{sort}} \circ \iota)^{-1}(\sigma)$ and any $\phi \in \mathbb{R}$, the followings are equivalent:

- $\Phi_{\text{solu}}(\iota(A, b, c, l, u))_{\sigma(1)} < \phi$.
- $\inf \left\{ x_{\sigma(1)} : x \in X_{\text{solu}}(\iota(A, b, c, l, u)) \right\} < \phi$.
- There exist $r \in \mathbb{N}_+$ and $x \in X_{\text{solu}}(\iota(A, b, c, l, u))$, such that $x_{\sigma(1)} \le \phi - 1/r$.
- There exists $r \in \mathbb{N}_+$, for any $r' \in \mathbb{N}_+$, $x_{\sigma(1)} \le \phi - 1/r$ holds for some $x \in \mathbb{Q}^k \times \mathbb{Z}^{n-k}$ satisfying $V_{\text{solu}}(A, b, c, l, u, x) < 1/r'$.

Therefore, it holds that

$$\Phi_1^{-1}(-\infty,\phi) = \iota^{-1}(\mathcal{D}_{\text{solu}}\backslash\mathcal{D}_{\text{foldable}}) \cap (\Phi_{\text{sort}}\circ\iota)^{-1}(\sigma) \cap \bigcup_{r\in\mathbb{N}_+}\bigcap_{r'\in\mathbb{N}_+}\bigcup_{x\in\mathbb{Q}^k\times\mathbb{Z}^{n-k},\ x_{\sigma(1)}\le\phi-1/r} \left\{(A,b,c,l,u)\in\mathbb{R}^{m\times n}\times\mathbb{R}^m\times\mathbb{R}^n\times\mathbb{R}^n\times\mathbb{R}^n : V_{\text{solu}}(A,b,c,l,u,x) < \frac{1}{r'}\right\},$$

is measurable, which implies the measurability of Φ_1

Then we assume that $\Phi_1,\ldots,\Phi_{j-1}$ ($j\ge 2$) are all measurable and show that Φ_j is also measurable. Define

$$V_{\text{solu}}^j(A,b,c,l,u,x) = \max\left\{V_{\text{solu}}(A,b,c,l,u,x),\ \max_{1\le j'<j}\left|x_{\sigma(j')}-\Phi_{j'}(A,b,c,l,u)\right|\right\},$$

for $(A,b,c,l,u)\in\iota^{-1}(\mathcal{D}_{\text{solu}}\backslash\mathcal{D}_{\text{foldable}})\cap(\Phi_{\text{sort}}\circ\iota)^{-1}(\sigma)$ and $x\in\mathbb{R}^k\times\mathbb{Z}^{n-k}$. Similar to V_{solu}, V_{solu}^j is also measurable and is continuous with respect to x. For any $(A,b,c,l,u)\in\iota^{-1}(\mathcal{D}_{\text{solu}}\backslash\mathcal{D}_{\text{foldable}})\cap(\Phi_{\text{sort}}\circ\iota)^{-1}(\sigma)$, the followings are equivalent:

- $\Phi_{\text{solu}}(\iota(A,b,c,l,u))_{\sigma(j)}<\phi$.
- $\inf\left\{x_{\sigma(j)} : x\in X_{\text{solu}}(\iota(A,b,c,l,u)),\ x_{\sigma(j')}=\Phi_{j'}(A,b,c,l,u),\ j'=1,2,\ldots,j-1\right\}<\phi$.
- There exist $r\in\mathbb{N}_+$ and $x\in\mathbb{R}^k\times\mathbb{Z}^{n-k}$, such that $V_{\text{solu}}^j(A,b,c,l,u,x)=0$ and that $x_{\sigma(j)}\le\phi-1/r$.
- There exists $r\in\mathbb{N}_+$, for any $r'\in\mathbb{N}_+$, $x_{\sigma(j)}\le\phi-1/r$ holds for some $x\in\mathbb{Q}^k\times\mathbb{Z}^{n-k}$ satisfying $V_{\text{solu}}^j(A,b,c,l,u,x)<1/r'$.

Therefore, $\Phi_j^{-1}(-\infty,\phi)$ can be expressed in similar format as $\Phi_1^{-1}(-\infty,\phi)$, and is hence measurable. □

Appendix D. Details of the Numerical Experiments

MILP instance generation. Each instance in $\mathcal{D}_1$ has 20 variables, 6 constraints and is generated with:

- For each variable, $c_j\sim\mathcal{N}(0,1)$, $l_j,u_j\sim\mathcal{N}(0,10)$. If $l_j>u_j$, then switch l_j and u_j. The probability that x_j is an integer variable is 0.5.
- For each constraint, $\circ_i\sim\mathcal{U}(\{\le,=,\ge\})$ and $b_i\sim\mathcal{N}(0,1)$.
- A has 60 nonzero elements with each nonzero element distributing as $\mathcal{N}(0,1)$.

Each instance in $\mathcal{D}_2$ has 20 variables, 6 equality constraints, and we construct the $(2k-1)$-th and $2k$-th problems via following approach ($1\le k\le 500$)

- Sample $J=\{j_1,j_2,\ldots,j_6\}$ as a random subset of $\{1,2,\ldots,20\}$ with 6 elements. For $j\in J$, $x_j\in\{0,1\}$. For $j\notin J$, x_j is a continuous variable with bounds $l_j,u_j\sim\mathcal{N}(0,10)$. If $l_j>u_j$, then switch l_j and u_j.
- $c_1=\cdots=c_{20}=0$.
- The constraints for the $(2k-1)$-th problem (feasible) is $x_{j_1}+x_{j_2}=1$, $x_{j_2}+x_{j_3}=1$, $x_{j_3}+x_{j_4}=1$, $x_{j_4}+x_{j_5}=1$, $x_{j_5}+x_{j_6}=1$, $x_{j_6}+x_{j_1}=1$.
- The constraints for the $2k$-th problem (infeasible) is $x_{j_1}+x_{j_2}=1$, $x_{j_2}+x_{j_3}=1$, $x_{j_3}+x_{j_1}=1$, $x_{j_4}+x_{j_5}=1$, $x_{j_5}+x_{j_6}=1$, $x_{j_6}+x_{j_4}=1$.

MLP architectures. As we mentioned in the main text, all the learnable functions in GNN are taken as MLPs. All the learnable functions $f_{\text{in}}^V, f_{\text{in}}^W, f_{\text{out}}, f_{\text{out}}^W, \{f_l^V, f_l^W, g_l^V, g_l^W\}_{l=0}^L$ are parameterized with multilayer perceptrons (MLPs) and have two hidden layers. The embedding size $d_0, \cdots, d_L$ are uniformly taken as d chosen from $\{2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. All the activation functions are ReLU.

Training settings. We use Adam [24] as our training optimizer with learning rate of 0.0001. The loss function is taken as mean squared error. All the experiments are conducted on a Linux server with an Intel Xeon Platinum 8163 GPU and eight NVIDIA Tesla V100 GPUs.

(Z. Chen) Department of Mathematics, Duke University, Durham, NC 27708.
Email address: ziang@math.duke.edu

(J. Liu) Decision Intelligence Lab, Damo Academy, Alibaba US, Bellevue, WA 98004.
Email address: jialin.liu@alibaba-inc.com

(X. Wang) Decision Intelligence Lab, Damo Academy, Alibaba US, Bellevue, WA 98004.
Email address: xinshang.w@alibaba-inc.com

(J. Lu) Departments of Mathematics, Physics, and Chemistry, Duke University, Durham, NC 27708.
Email address: jianfeng@math.duke.edu

(W. Yin) Decision Intelligence Lab, Damo Academy, Alibaba US, Bellevue, WA 98004.
Email address: wotao.yin@alibaba-inc.com