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 [39], control [34], scheduling [16], etc. Branch and Bound (B&B) [27], 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 [7]. 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, 24, 28, 46]. One may also learn cutting strategies [9, 23, 42], node selection/pruning strategies [22, 45], or decomposition strategies [41] with ML models. The role of ML models in those approaches can be summarized as: approximating useful mappings or parameterizing
key strategies in MILP solvers, and these mappings/strategies usually take an MILP instance as input and output its key properties.

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. [17] proposed that an MILP can be encoded into a bipartite graph on which one can use a GNN to approximate Strong Branching. [13] 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 [20] [21] [31] [40], approximating optimal solution [25] [31], parameterizing cutting strategies [32], and parameterizing branching strategies [33] [38].

However, theoretical foundations in this direction still remain unclear. A key problem is the ability of GNN to approximate important mappings related with MILP. 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 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 feasibility, optimal objective value and an optimal solution. The separation power and representation power of GNNs are closely related to the Weisfeiler-Lehman (WL) test [43], a classical algorithm to identify whether two graphs are isomorphic or not. It has been shown that GNN has the same separation power with the WL test [44], and, based on this result, GNNs can universally approximate continuous graph-input mappings with separation power no stronger than the WL test [5] [18].

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 [30] [36] and linear programming [12], 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, for unfoldable MILPs, GNN has strong enough separation power and representation power to approximate the feasibility, optimal objective value and an optimal solution.

• (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 answers to (Q1 - Q3) serve as foundations of a more practical question: whether GNNs are able to predict branching strategies or primal heuristics for MILP? Although the answers to (Q1) and (Q2) do not directly answer that practical question, they illustrate the possibility that GNNs can capture some key information of a MILP instance and have the capacity to suggest an adaptive branching strategy for each instance. To obtain a GNN-based strategy, practitioners should consider more factors: feature spaces, action spaces, training methods, generalization performance, etc., and some recent empirical studies show encouraging results on learning such a strategy [17,20,21,31,33,38,40]. The answer to (Q3) directly shows the possibility of learning primal heuristics. With proper model design and training methods, one could obtain competitive GNN-based primal heuristics [13,31].

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, definitions, and examples follow [12]. Consider a general MILP problem defined with:

\[
\min_{x \in \mathbb{R}^n} \ c^\top x, \quad \text{s.t.} \ Ax \circ b, \ l \leq x \leq u, \ x_j \in \mathbb{Z}, \ \forall \ j \in I,
\]

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 \( \circ \in \{\leq, =, \geq\}^m. \) The index set \( I \subset \{1, 2, \ldots, n\} \) includes those indices \( j \) where \( x_j \) are constrained to be an integer. The feasible set is defined with \( X_{\text{feasible}} := \{x \in \mathbb{R}^n | Ax \circ b, \ l \leq x \leq u, \ x_j \in \mathbb{Z}, \forall \ j \in I\}, \) 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 \( c^\top x^* \leq c^\top x, \ \forall \ x \in X_{\text{feasible}}, \) then we say that \( x^* \) is an optimal solution. It is possible that the objective value is arbitrarily good, i.e., for any \( R > 0, \ c^\top x < -R \) holds for some \( x \in X_{\text{feasible}}. \) 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. Inspired by \cite{17} and following \cite{12}, we represent MILP by weighted bipartite graphs. The vertex set of such a graph is \(V \cup W\), where \(V = \{v_1, v_2, \ldots, v_m\}\) with \(v_i\) representing the \(i\)-th constraint, and \(W = \{w_1, w_2, \ldots, w_m\}\) with \(w_j\) representing the \(j\)-th variable. To fully represent all information in \cite{21}, we associate each vertex with features: The vertex \(v_i \in V\) is equipped with a feature vector \(h_i^V = (v_i, c_i)\) that is chosen from \(\mathcal{H}^V = \mathbb{R} \times \{\leq, =, \geq\}\). The vertex \(w_j \in W\) is equipped with a feature vector \(h_j^W = (c_j, l_j, u_j, \tau_j)\), where \(\tau_j = 1\) if \(j \in I\) 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 \(E_{i,j} \in \mathbb{R}\) connects \(v_i \in V\) and \(w_j \in W\), and its value is defined with \(E_{i,j} = A_{i,j}\). Note that there is no edge connecting vertices in the same vertex group (\(V\) or \(W\)) and \(E_{i,j} = 0\) if there is no connection between \(v_i\) and \(w_j\). Thus, the whole graph is denoted as \(G = (V \cup W, E)\), and we denote \(G_{m,n}\) as the collection of all such weighted bipartite graphs whose two vertex groups have size \(m\) and \(n\), respectively. Finally, we define \(\mathcal{H}^V := (\mathcal{H}^V)^m\) and \(\mathcal{H}^W := (\mathcal{H}^W)^n\), 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 \((G, H) \in G_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W\) contains all information in the MILP problem \cite{21}, and we name such a graph as a MILP-induced graph or MILP-graph \cite{21}. If \(I = \emptyset\), the feature \(\tau_j\) can be dropped and the graphs reduce to LP-graphs in \cite{12}. We provide an example of MILP-graph in Figure 1.

![Figure 1. An example of MILP-graph](image)

\[
\begin{align*}
\min_{x \in \mathbb{R}^2} & \quad x_1 + x_2, \\
\text{s.t.} & \quad x_1 + 3x_2 \geq 1, \quad x_1 + x_2 \geq 1, \\
& \quad x_1 \leq 3, \quad x_2 \leq 5, \quad x_2 \in \mathbb{Z}.
\end{align*}
\]

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: \(G_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R}\); 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: \(G_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \to \mathbb{R}^n\). The GNNs used in this paper can be constructed with the following three steps:

1. **Initial mapping at level \(l = 0\).** Let \(p^0\) and \(q^0\) be learnable embedding mappings, then the embedded features are \(s_i^0 = p^0(h_i^V)\) and \(t_j^0 = q^0(h_j^W)\) for \(i = 1, 2, \ldots, m\) and \(j = 1, 2, \ldots, n\).

2. **Message-passing layers for \(l = 1, 2, \ldots, L\).** Given learnable mappings \(f^l_i, g^l_j, p^l, q^l\), one can update vertex features by the following formulas for all \(i = 1, 2, \ldots, m\) and \(j = 1, 2, \ldots, n\):

\[
s_i^l = p^l\left(s_i^{l-1} + \sum_{j=1}^m E_{i,j} f^l(t_j^{l-1})\right), \quad t_j^l = q^l\left(t_j^{l-1} + \sum_{i=1}^m E_{i,j} g^l(s_i^{l-1})\right).
\]

In \cite{17} and other related empirical studies, the feature spaces \(\mathcal{H}^V, \mathcal{H}^W\) are more complicated than those defined in this paper. We only keep the most basic features here for simplicity of analysis.
an MILP to exactly one of its optimal solutions: We can further define the following mapping which maps
Φ
and we consider MILP problems taken from

More specifically, the vertex feature space is limited to
be finite, which implies that an optimal solution exists as long as the MILP problem is feasible.
In some cases. To handle this issue, we only consider the case that any component in
2. To give a well-defined optimal solution mapping is much more

Optimal solution mapping

An example would be min

where Φ
 obj

Note that the optimal objective value for MILP may be an infimum that can never be achieved.

Feasibility mapping

Throughout this paper, we require all learnable mappings to be continuous following the
same settings as in
. In practice, one may parameterize those mappings with multi-layer
perceptions (MLPs). Define
F
GNN
is defined similarly by replacing (3a) with (3b).

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
(2) and (3a).

Optimal objective value mapping

where
Φ
obj

implies infeasibility and
Φ
feas
:solu
:folds

For a well-defined optimal solution mapping is much more

Optimal solution mapping. To give a well-defined optimal solution mapping is much more
complicated 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.

More specifically, the vertex feature space is limited to

and we consider MILP problems taken from

and the output is a
vector
y ∈ \mathbb{R}^n
with entries being

Therefore, a mapping
Φ
solu
, which maps an LP to its optimal solution with the smallest \ell_2-norm, is well defined on
Φ
−1
obj
(\mathbb{R}).
where $\mathcal{D}_{\text{foldable}}$ is a subset of $\mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n$ that will be introduced in Section 4. The full definition of $\Phi_{\text{solu}}$ is placed in Appendix C due to its tediousness.

**Invariance and equivariance.** Now we discuss some properties of the three defined mappings. Mappings $\Phi_{\text{feas}}$ and $\Phi_{\text{obj}}$ 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 $\Phi_{\text{solu}}$ 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 $\sigma_W \in S_n$, $(\sigma_V, \sigma_W) \ast (G, H)$ denotes the reordered MILP-graph with permutations $\sigma_V, \sigma_W$. It is clear that both $\Phi_{\text{feas}}$ and $\Phi_{\text{obj}}$ are permutation invariant in the following sense:

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

for all $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H) \in \mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n$. In addition, $\Phi_{\text{solu}}$ is permutation equivariant: $\Phi_{\text{solu}}((\sigma_V, \sigma_W) \ast (G, H)) = \sigma_W(\Phi_{\text{solu}}(G, H))$, for all $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H) \in \mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}}$. This will be discussed in Section C. Furthermore, one may check that any $F \in \mathcal{F}_{\text{GNN}}$ is invariant and any $F_W \in \mathcal{F}^W_{\text{GNN}}$ is equivariant.

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

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 $\Phi_{\text{feas}}$, $\mathcal{F}_{\text{GNN}}$ should have stronger separation power than $\Phi_{\text{feas}}$: for any two MILP instances $(G, H), (\hat{G}, \hat{H}) \in \mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n$,

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

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 $\Phi_{\text{feas}}$. 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 \[43\]. We present a variant of WL test specially modified for MILP in Algorithm 1 that follows the same lines as in \[12\] Algorithm 1, where each vertex is labeled with a color. For example, $v_i \in V$ is initially labeled with $C^0_{i,V}$ based on its feature $h^V_i$ by hash function $\text{HASH}_{0,V}$ 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 $\{\{C^i_{i,V}\}_{i=0}^m, \{\{C^j_{j,W}\}_{j=0}^n\}$, where $\{\{\}$ refers to a
Lemma 3.2. There exist two MILP problems \( G, H \) such that one is feasible and the other one being infeasible, such that \( G \sim H \). As noted in Theorem 3.1 (Theorem 4.2 in [12]), the separation power of WL test is weaker than \( \Phi_{\text{feas}} \) of MILP-Graphs. This theorem reveals that the separation power of WL test is weaker than \( \Phi_{\text{feas}} \) of MILP-Graphs.

Theorem 3.1 (Theorem 4.2 in [12]). 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 \( F(G, H) = F(\hat{G}, \hat{H}), \forall F \in \mathcal{F}_{\text{GNN}} \).

Theorem 3.1 is stated and proved in [12] for LP-graphs, but it actually also applies for MILP-graphs. The intuition for Theorem 3.1 is straightforward since the color refinement in WL test and the message-passing layer in GNNs have similar structure. The proof of [12] Theorem 4.2 basically uses the fact that given finitely many inputs, there always exist hash functions or continuous functions that are injective and can generate linearly independent outputs, i.e., without collision. We also remark that the equivalence between the separation powers of GNNs and WL test has been investigated in some earlier literature [5,18,44]. Unfortunately, the following lemma reveals that the separation power of WL test is weaker than \( \Phi_{\text{feas}} \) of MILP-Graphs.

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 problems and their induced graphs:

The two MILP-graphs in Figure 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 Algorithm 1, the hash functions \{HASH_{i,V}, HASH_{i,W}\} 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: \( C_i^{L,V} = HASH_{i,V}(C_i^{l-1,V}, \{\{HASH_{i,W}(C_j^{l-1,W}, E_{i,j})\})\), \( C_j^{L,W} = HASH_{i,W}(C_j^{l-1,W}, \{\{HASH_{i,V}(C_i^{l-1,V}, E_{i,j})\})\).
Section 2.2, we can explicitly write down the vertex features for each vertex here: \( h_i^V = (1, \ldots, 1), 1 \leq i \leq 6 \) and \( h_j^W = (1, 0, 1, 1), 1 \leq j \leq 6 \). 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 \( \{ C_i^{1, V} \}_{i=1}^6 \) are all blue and \( \{ C_j^{1, W} \}_{j=1}^6 \) are all red, one will obtain for both of the graphs in Figure 2 that \( C_1^{2+1, V} = C_2^{2+1, V} = \cdots = C_6^{2+1, V} \) and \( C_1^{2+1, W} = C_2^{2+1, W} = \cdots = C_6^{2+1, W} \) 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.

For those instances in Figure 2 GNNs may struggle to predict a meaningful neural diving or branching strategy. In neural diving [31], one usually trains a GNN to predict an estimated solution; in neural branching [17], one usually trains a GNN to predict a ranking score for each variable and uses that score to decide which variable is selected to branch first. However, the analysis above illustrates that the WL test or GNNs cannot distinguish the 6 variables, and one cannot get a meaningful ranking score or estimated solution.

### 4. Unfoldable MILP Problems

We prove with example in Section 3 that one may not expect good performance of GNN to approximate \( \Phi_{\text{feas}} \) on a general dataset of MILP problems. It’s worth to ask: is it possible to describe the common characters of those hard examples? 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 \( \{C_i^{l,V}, C_j^{l,W}\}_{i,j} \) 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

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

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 \( \mathcal{D}^{\text{foldable}} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \) 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 \( \text{HASH}_{0,W}, \text{HASH}_{1,W,1}, \text{HASH}_{1,V} \), 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 \( \mathbb{I}(F(G, H) > 1/2) \) 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 \( \mathcal{D} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \setminus \mathcal{D}^{\text{foldable}} \), there exists \( F \in \mathcal{F}_{GNN} \) such that

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

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

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

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

and there exists some \( F_2 \in \mathcal{F}_{GNN} \) such that

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

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

\[
\|F_W(G, H) - \Phi_{\text{solu}}(G, H)\| < \delta, \quad \forall (G, H) \in \mathcal{D}.
\]
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.

Let us also mention that the foldability of an MILP instance may depend on the feature spaces $\mathcal{H}^V$ and $\mathcal{H}^W$. If more features are appended (see [17]), i.e., $h_i^V$ and $h_j^W$ have more entries, and then a foldable MILP problem may become unfoldable. In addition, all the analysis here works for any topological linear spaces $\mathcal{H}^V$ and $\mathcal{H}^W$, as long as $\Phi_{\text{feas}}$, $\Phi_{\text{obj}}$, and $\Phi_{\text{solu}}$ can be verified as measurable and permutation-invariant/equivariant mappings.

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. For example, for around $1/4$ of the problems in MIPLIB 2017 [19], the number of colors generated by WL test is smaller than one half of the number of vertices, respect to the $\mathcal{H}^V$ and $\mathcal{H}^W$ used in this paper. To resolve this issue, we introduce a technique inspired by [1,37]. More specifically, we append the vertex features with an additional random feature $\omega$ and define a type of random GNNs as follows.

Let $\Omega = [0,1]^m \times [0,1]^n$ and let $(\Omega, \mathcal{F}_\Omega, \mathcal{P})$ 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

$$F_R : \mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \times \Omega \rightarrow \mathbb{R},$$

$$(G, H, \omega) \mapsto F_R(G, H, \omega),$$

which is defined in the same way as $\mathcal{F}_{\text{GNN}}$ with input space being $\mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \times \Omega \cong \mathcal{G}_{m,n} \times (\mathcal{H}^V \times [0,1])^m \times (\mathcal{H}^W \times [0,1])^n$ and $\omega$ 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

$$F_{W,R} : \mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \times \Omega \rightarrow \mathbb{R}^n,$$

$$(G, H, \omega) \mapsto F_{W,R}(G, H, \omega),$$

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

By the definition of graph neural networks, $F_R$ is permutation invariant and $F_{W,R}$ is permutation equivariant, where the permutations $\sigma_V$ and $\sigma_W$ act on $(\mathcal{H}^V \times [0,1])^m$ and $(\mathcal{H}^W \times [0,1])^n$. 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 \( D \subset G_{m,n} \times H_{V}^{m} \times H_{W}^{n} \) be a finite dataset. For any \( \epsilon > 0 \), there exists \( F_{R} \in \mathcal{F}_{GNN}^{R} \), such that
\[
P \left( \mathbb{I}_{I_{F_{R}}(G,H) > 1/2} \neq \Phi_{\text{feas}}(G,H) \right) < \epsilon, \quad \forall (G,H) \in D.
\]
Similar results also hold for the optimal objective value and the optimal solution.

Theorem 5.2. Let \( D \subset G_{m,n} \times H_{V}^{m} \times H_{W}^{n} \) be a finite dataset. For any \( \epsilon, \delta > 0 \), there exists \( F_{R,1} \in \mathcal{F}_{GNN}^{R} \), such that
\[
P \left( \mathbb{I}_{I_{F_{R,1}}(G,H) > 1/2} \neq \Phi_{\text{obj}}(G,H) \right) < \epsilon, \quad \forall (G,H) \in D,
\]
and there exists \( F_{R,2} \in \mathcal{F}_{GNN}^{R} \), such that
\[
P \left( |F_{R,2}(G,H) - \Phi_{\text{soln}}(G,H)| > \delta \right) < \epsilon, \quad \forall (G,H) \in D \cap \Phi_{\text{obj}}^{-1}(R).
\]

Theorem 5.3. Let \( D \subset \Phi_{\text{obj}}^{-1}(R) \subset G_{m,n} \times H_{V}^{m} \times H_{W}^{n} \) be a finite dataset. For any \( \epsilon, \delta > 0 \), there exists \( F_{W,R} \in \mathcal{F}_{GNN}^{W,R} \), such that
\[
P \left( \|F_{W,R}(G,H) - \Phi_{\text{soln}}(G,H)\| > \delta \right) < \epsilon, \quad \forall (G,H) \in D.
\]

The idea behind those theorems is that GNNs can distinguish \((G,H,\omega)\) and \((\hat{G},\hat{H},\hat{\omega})\) with high probability, even if they cannot distinguish \((G,H)\) and \((\hat{G},\hat{H})\). How to choose the random feature in training is significant. In practice, one may generate one random feature vector \( \omega \in [0,1]^{m} \times [0,1]^{n} \) for all MILP instances. This setting leads to efficiency in training GNN models, but the trained GNNs cannot be applied to datasets with MILP problems of different sizes. Another practice is sampling several independent random features for each MILP instance, but one may suffer from difficulty in training. Such a trade-off also occurs in other GNN tasks \([6,29]\). In this paper, we generate one random vector for all instances (both training and testing instances) to directly validate our theorems. How to balance the trade-off in practice will be an interesting future topic.

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 \( D_{1} \) and \( D_{2} \). Set \( D_{1} \) consists of 1000 randomly generate MILPs that are all unfoldable, and there are 460 feasible MILPs whose optimal solutions are attachable while the others are all infeasible. Set \( D_{2} \) 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 \([10,11]\), 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 \( f_{V}, f_{W}, f_{out}, f_{out}^{L}, \{f_{V,l}, f_{W,l}, g_{V,l}, g_{W,l}\}_{l=0}^{L} \) as multilayer perceptrons (MLPs). Our codes are modified from \([17]\) and released to https://github.com/liujl11/GNN-MILP.git. All the results reported in this section are obtained on the training sets, not an separate testing set. Generalization tests and 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 $D_2$. 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 $D_1$, 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 $D_2$, which validates Theorem 5.1 and the first conclusion in Theorem 5.2.

Optimal value and solution. Then we take the feasible instances from sets $D_1$ and $D_2$ and form new datasets $D_1^{\text{feasible}}$ and $D_2^{\text{feasible}}$, respectively. On $D_1^{\text{feasible}}$ and $D_2^{\text{feasible}}$, 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.

Figure 3. Feasibility

Figure 4. GNN can approximate $\Phi_{\text{obj}}$ and $\Phi_{\text{solu}}$

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] Waiss Azizian and Marc Lelarge, Expressive power of invariant and equivariant graph neural networks, International conference on learning representations, 2021.
[5] Muhammet Balciar, Pierre Héroux, Benoit Gauzere, Pascal Vasseur, Sébastien Adam, and Paul Honeine, Breaking the limits of message passing graph neural networks, International conference on machine learning, 2021, pp. 599–608.
[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 Graceyk, Katrijn Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübecke, Stephen J. Maher, Frederic Matter, Erik Mühlmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sopronac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig, The SCIP Optimization Suite 8.0, Optimization Online, 2021.
[10] 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).
[11] Jian-Ya Ding, Chao Zhang, Lei Shen, Shengyu 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.
[12] Brendan L Douglas, The weisfeiler-lehman method and graph isomorphism testing, arXiv preprint arXiv:1101.5211 (2011).
[13] Lawrence C Evans and Ronald F Garzepy, Measure theory and fine properties of functions, Routledge, 2018.
[14] 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.
[17] 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).

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

[19] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarrack, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozsyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano, *MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library*, Mathematical Programming Computation (2021).

[20] 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.

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

[22] 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).

[23] 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.

[24] 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.

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

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

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

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

[29] Andreas Loukas, *What graph neural networks cannot learn: depth vs width*, International conference on learning representations, 2019.

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

[31] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O’Donoghue, Nicolas Sonnerat, Christian Tjandraatmadjja, Pengming Wang, Ravichandra Addanki, Tharindu 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).

[32] 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.

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

[34] Arthur Richards and Jonathan How, *Mixed-integer programming for control*, Proceedings of the 2005, american control conference, 2005., 2005, pp. 2076–2083.

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

[36] Ryoma Sato, Makoto Yamada, and Hisashi Kashima, *Approximation ratios of graph neural networks for combinatorial problems*, Advances in Neural Information Processing Systems 32 (2019).
Appendix A. Proofs for Section 4

In this section, we prove the results in Section 4, i.e., graph neural networks can approximate \(\Phi_{\text{feas}}, \Phi_{\text{obj}},\) and \(\Phi_{\text{soln}}\), on finite datasets of unfoldable MILPs with arbitrarily small error. We would consider more general results on finite-measure subset of \(G_{m,n} \times H_V^m \times H_W^n\) which may involve infinite elements.

In our settings, foldability is actually a concept only depending on the MILP instance. In the main text, we utilize the tool of the WL test to define foldability here for readability. Here we present a more complex but equivalent definition merely with the language of MILP.

**Definition A.1 (Foldable MILP).** We say that the MILP problem (2.1) can be folded, or is foldable, if there exist \(I = \{I_1, I_2, \ldots, I_s\}\) and \(J = \{J_1, J_2, \ldots, J_t\}\) that are partitions of \(\{1, 2, \ldots, m\}\) and \(\{1, 2, \ldots, n\}\), respectively, such that at least one of \(I\) and \(J\) is nontrivial, i.e., \(s < m\) or \(t < n\), and that the followings hold for any \(p \in \{1, 2, \ldots, s\}\) and \(q \in \{1, 2, \ldots, t\}\): vertices in \(\{v_i : i \in I_p\}\) share the same feature in \(H_V\), vertices in \(\{w_j : j \in J_q\}\) share the same feature in \(H_W\), and all column sums (and row sums) of the submatrix \((A_{i,j})_{i \in I_p, j \in J_q}\) are the same.

According to [12, Appendix A], the color refinement procedure of the WL test converges to the coarsest partitions \(I\) and \(J\) satisfying the conditions in Definition A.1 assuming no collision happens. Therefore, one can see that Definition 4.1 and Definition A.1 are equivalent.

Before processing to establish the approximation theorems for GNNs on \(G_{m,n} \times H_V^m \times H_W^n\), let us define the topology and the measure on \(G_{m,n} \times H_V^m \times H_W^n\).

**Topology and measure.** The space \(G_{m,n} \times H_V^m \times H_W^n\) can be equipped with some natural topology and measure. In the construction of \(G_{m,n} \times H_V^m \times H_W^n\), 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 $G_{m,n} \times H_{m}^{V} \times H_{n}^{W}$ are defined. We use the notation $\text{Meas}(\cdot)$ for the measure on $G_{m,n} \times H_{m}^{V} \times H_{n}^{W}$.

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.2.** Let $X \subset G_{m,n} \times H_{m}^{V} \times H_{n}^{W} \setminus \mathcal{D}_{\text{foldable}}$ be measurable with finite measure. For any $\epsilon > 0$, there exists some $F \in \mathcal{F}_{GNN}$ such that

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

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

**Theorem A.3.** Let $X \subset G_{m,n} \times H_{m}^{V} \times H_{n}^{W} \setminus \mathcal{D}_{\text{foldable}}$ 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

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

(ii) There exists some $F_{2} \in \mathcal{F}_{GNN}$ such that

$$\text{Meas}\left(\{(G, H) \in X \cap \Phi_{\text{obj}}^{-1}(\mathbb{R}) : |F_{2}(G, H) - \Phi_{\text{obj}}(G, H)| > \delta\}\right) < \epsilon.$$

**Theorem A.4.** Let $X \subset \mathcal{D}_{\text{solut}} \setminus \mathcal{D}_{\text{foldable}} \subset G_{m,n} \times H_{m}^{V} \times H_{n}^{W} \setminus \mathcal{D}_{\text{foldable}}$ be measurable with finite measure. For any $\epsilon, \delta > 0$, there exists $F_{W} \in \mathcal{F}_{GNN}^{W}$ such that

$$\text{Meas}\left(\{(G, H) \in X : \|F_{W}(G, H) - \Phi_{\text{solut}}(G, H)\| > \delta\}\right) < \epsilon.$$

The proof framework is similar to those in [12] 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 $\Phi_{\text{feas}}$ and $\Phi_{\text{obj}}$ are both measurable in the following two lemmas. The optimal solution mapping $\Phi_{\text{solut}}$ will be defined rigorously and proved as measurable in Section C.

**Lemma A.5.** The feasibility mapping $\Phi_{\text{feas}} : G_{m,n} \times H_{m}^{V} \times H_{n}^{W} \rightarrow \{0, 1\}$ 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 [12] Lemma F.1]. 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 [12] Lemma F.1] 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. \(\square\)

**Lemma A.6.** The optimal objective value mapping $\Phi_{\text{obj}} : G_{m,n} \times H_{m}^{V} \times H_{n}^{W} \rightarrow \mathbb{R} \cup \{-\infty, \infty\}$ is measurable.
Theorem A.7 (Lusin’s theorem [15, Theorem 1.14]). Let \( \mu \) be a Borel regular measure on \( \mathbb{R}^n \) and let \( f : \mathbb{R}^n \to \mathbb{R}^m \) be \( \mu \)-measurable. Then for any \( \mu \)-measurable \( X \subset \mathbb{R}^n \) with \( \mu(X) < \infty \) and any \( \epsilon > 0 \), there exists a compact set \( E \subset X \) with \( \mu(X \setminus E) < \epsilon \), such that \( f|_E \) is continuous.

The main tool in the proof of Theorem A.2 and Theorem A.3 is the universal approximation property of \( F_{GNN} \) stated below. Notice that the separation power of GNNs is the same as that of WL test. Theorem A.8 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 [35, Section 5.7].

Theorem A.8 (Theorem 4.3 in [12]). Let \( X \subset G_{m,n} \times H^V_m \times H^W_n \) be a compact set. For any \( \Phi \in C(X, \mathbb{R}) \) satisfying

\[
\Phi(G, H) = \Phi(\hat{G}, \hat{H}), \quad \forall (G, H) \sim (\hat{G}, \hat{H}),
\]

and any \( \epsilon > 0 \), there exists \( F \in F_{GNN} \) such that

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

Let us remark that the proof of [12, Theorem 4.3] does not depend on the specific choice of the feature spaces \( H^V \) and \( H^W \), which only requires that \( H^V \) and \( H^W \) are both topological linear spaces. Therefore, Theorem A.8 works for MILP-graphs with additional vertices features.

For the sake of completeness, we outline the proof of [12, Theorem 4.3]: Let \( \pi : G_{m,n} \times H^V_m \times H^W_n \to G_{m,n} \times H^V_m \times H^W_n / \sim \) be the quotient. Then for any continuous function \( F : G_{m,n} \times H^V_m \times H^W_n \to \mathbb{R} \), there exists a unique \( \tilde{F} : G_{m,n} \times H^V_m \times H^W_n / \sim \to \mathbb{R} \) such that \( F = \tilde{F} \circ \pi \). Note that \( \tilde{F}_{GNN} := \{ \tilde{F} : F \in F_{GNN} \} \) separates points in \( G_{m,n} \times H^V_m \times H^W_n / \sim \) by Theorem 3.1. By the Stone-Weierstrass theorem, there exists \( \tilde{F} \in \tilde{F}_{GNN} \) with \( \sum_{(G,H) \in X} |\tilde{F}(\pi(G, H)) - F(\pi(G, H))| < \epsilon \), which then implies \( \epsilon \).

Similar universal approximation results for GNNs and invariant target mappings can also be found in earlier literature; see e.g. [5,18]. With all the preparation above, we can then proceed to present the proof of Theorem A.2.

For applying Theorem A.8 one need to verify that the separation power of \( \Phi_{\text{cas}} \) and \( \Phi_{\text{obj}} \) are bounded by that of WL test for unfoldable MILP instances, i.e., the condition (A.1). For any \((G,H), (\hat{G}, \hat{H}) \in G_{m,n} \times H^V_m \times H^W_n / \mathcal{D}_{\text{foldable}}\), 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 have the following lemma:

**Lemma A.9.** For any \((G, H), (\hat{G}, \hat{H}) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \setminus \mathcal{D}_{\text{foldable}},\) it holds that \((G, H) \sim (\hat{G}, \hat{H})\) 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 \(\sigma_W \in S_n\).

Let us remark that Lemma A.9 is true for any vertex feature spaces because the following proof does not assume the structure of the feature spaces.

**Proof.** It’s trivial that the isomorphism of \((G, H)\) and \((\hat{G}, \hat{H})\) implies \((G, H) \sim (\hat{G}, \hat{H})\), and it suffices to show \((G, H) \sim (\hat{G}, \hat{H})\) implies the isomorphism.

Applying Algorithm 1 on \((G, H)\) and \((\hat{G}, \hat{H})\) with a common set of hash functions, we define the outcomes respectively:

\[
\begin{align*}
\{\{C_i^{L,V}\}_{i=0}^{m}, \{C_j^{L,W}\}_{j=0}^{n} : & = \text{WL}_\text{MLP}((G, H), L), \\
\{\{\hat{C}_i^{L,V}\}_{i=0}^{m}, \{\hat{C}_j^{L,W}\}_{j=0}^{n} : & = \text{WL}_\text{MLP}((\hat{G}, \hat{H}), L).
\end{align*}
\]

Then \((G, H) \sim (\hat{G}, \hat{H})\) can be written as

\[
(A.3) \quad \{\{C_i^{L,V}\}_{i=0}^{m} = \{\{\hat{C}_i^{L,V}\}_{i=0}^{m}, \text{ and } \{\{C_j^{L,W}\}_{j=0}^{n} = \{\{\hat{C}_j^{L,W}\}_{j=0}^{n}
\]

for all \(L \in \mathbb{N}\) and all hash functions. We define hash functions used in this proof.

- We take hash functions \(\text{HASH}_0^V, \text{HASH}_0^W\) such that they are injective on the following domain:

\[
\{h_i^V, h_j^W, \hat{h}_i^V, \hat{h}_j^W : 1 \leq i \leq m, 1 \leq j \leq n\}.
\]

Note that \(\{\cdot\}\) means a set, but not a multiset. The above set collects all possible values of the vertex features in the given two graphs \((G, H)\) and \((\hat{G}, \hat{H})\). Such hash functions exist because the number of vertices in the two graphs must be finite.

- For \(l \in \{1, 2, \cdots, L\}\), we define a set that collects all possible vertex colors at the \(l\) - 1-th iteration,

\[
C_{l-1} := \{C_i^{l-1,V}, C_j^{l-1,W}, \hat{C}_i^{l-1,V}, \hat{C}_j^{l-1,W} : 1 \leq i \leq m, 1 \leq j \leq n\}.
\]

Note that \(|C_{l-1}|\) is finite. The two sets \(\{\text{HASH}_l^V(C) : C \in C_{l-1}\}\) and \(\{\text{HASH}_l^W(C) : C \in C_{l-1}\}\) can be linearly independent for some hash functions \(\text{HASH}_l^V, \text{HASH}_l^W\). This is because that one can take the output spaces of \(\text{HASH}_l^V, \text{HASH}_l^W\) as linear spaces of dimension greater than \(|C_{l-1}|\). Given such functions \(\text{HASH}_l^V, \text{HASH}_l^W\), we consider the following set

\[
\left\{\left(\sum_{j=1}^{n} E_{i,j} \text{HASH}_l^W(C_j^{l-1,W})\right) : 1 \leq i \leq m\right\}
\]
Due to the statement assumption that (A.4) of the above hash functions, one can obtain that
\[ \mathcal{E}_{i,j}^{d-1,V} \sum_{j=1}^{n} \hat{E}_{i,j} \text{HASH}_{L,W}^{i} \left( \hat{C}_{j}^{d-1,W} \right) : 1 \leq i \leq m \]
\[ \mathcal{E}_{j}^{d-1,W} \sum_{i=1}^{n} \hat{E}_{i,j} \text{HASH}_{L,V}^{i} \left( \hat{C}_{i}^{d-1,V} \right) : 1 \leq j \leq n \]
\[ \mathcal{E}_{j}^{d-1,W} \sum_{i=1}^{n} \hat{E}_{i,j} \text{HASH}_{L,V}^{i} \left( \hat{C}_{i}^{d-1,V} \right) : 1 \leq j \leq n \],

which is a finite set given any two graphs \((G, H)\) and \((\hat{G}, \hat{H})\). Thus, one can pick hash functions \(\text{HASH}_{L,V}, \text{HASH}_{L,W}\) that are injective on the above set.

Due to the statement assumption that \((G, H)\) and \((\hat{G}, \hat{H})\) are both unfoldable and the definition of the above hash functions, one can obtain that
\[ C_{i}^{L,V} \neq C_{i'}^{L,V}, \quad \forall i \neq i', \quad C_{j}^{L,W} \neq C_{j'}^{L,W}, \quad \forall j \neq j', \]
\[ \hat{C}_{i}^{L,V} \neq \hat{C}_{i'}^{L,V}, \quad \forall i \neq i', \quad \hat{C}_{j}^{L,W} \neq \hat{C}_{j'}^{L,W}, \quad \forall j \neq j', \]
for some \(L \in \mathbb{N}\). In another word, in either \((G, H)\) or \((\hat{G}, \hat{H})\), each vertex has an unique color.

Combining \((A.3)\) and \((A.4)\), we have that there exists unique \(\sigma_{V} \in S_{m}\) and \(\sigma_{W} \in S_{n}\) such that
\[ \hat{C}_{i}^{L,V} = C_{\sigma_{V}(i)}^{L,V}, \quad \hat{C}_{j}^{L,W} = C_{\sigma_{W}(j)}^{L,W} \quad \forall i = 1, 2, \ldots, m, j = 1, 2, \ldots, n. \]

In addition, the vertices with the same colors much have the same neighbour information, including the colors of the neighbourhood and weights of the associated edges. Rigorously, since \(\text{HASH}_{L,V}, \text{HASH}_{L,W}\) are both injective, one will obtain that
\[ \hat{C}_{i}^{d-1,V} = C_{\sigma_{V}(i)}^{d-1,V}, \quad \hat{C}_{j}^{d-1,W} = C_{\sigma_{W}(j)}^{d-1,W} \]
\[ \sum_{j=1}^{n} \hat{E}_{i,j} \text{HASH}_{L,W}^{i} \left( \hat{C}_{j}^{d-1,W} \right) = \sum_{j=1}^{n} E_{\sigma_{V}(i),j} \text{HASH}_{L,W}^{i} \left( C_{j}^{d-1,W} \right) \]
\[ \sum_{i=1}^{n} \hat{E}_{i,j} \text{HASH}_{L,V}^{i} \left( \hat{C}_{i}^{d-1,V} \right) = \sum_{i=1}^{n} E_{i,\sigma_{W}(j)} \text{HASH}_{L,V}^{i} \left( C_{i}^{d-1,V} \right) \]

Now let us consider the second line of the above equations,
\[ \sum_{j=1}^{n} \hat{E}_{i,j} \text{HASH}_{L,W}^{i} \left( \hat{C}_{j}^{d-1,W} \right) = \sum_{j=1}^{n} E_{\sigma_{V}(i),\sigma_{W}(j)} \text{HASH}_{L,W}^{i} \left( C_{\sigma_{W}(j)}^{d-1,W} \right) \]
\[ = \sum_{j=1}^{n} E_{\sigma_{V}(i),\sigma_{W}(j)} \text{HASH}_{L,W}^{i} \left( \hat{C}_{j}^{d-1,W} \right) \]

Therefore, it holds that
\[ \sum_{j=1}^{n} \left( \hat{E}_{i,j} - E_{\sigma_{V}(i),\sigma_{W}(j)} \right) \text{HASH}_{L,W}^{i} \left( \hat{C}_{j}^{d-1,W} \right) = 0. \]
This implies that \( \hat{E}_{i,j} = E_{\sigma_V(i), \sigma_W(j)} \) because \{\text{HASH}'_{L,W}(C) : C \in C_{L-1} \} \) are linearly independent. Then one can conclude that \((\sigma_V, \sigma_W) \ast (G, H) = (\hat{G}, \hat{H})\). \( \square \)

Now we can proceed to present the proof of Theorem A.2.

**Proof of Theorem A.2** According to Lemma A.5, 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 \( E \subset X \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \setminus \text{foldable} \) such that \( \Phi_{\text{feas}}|_E \) is continuous and that \( \text{Meas}(X \setminus E) < \epsilon \). For any \((G, H), (\hat{G}, \hat{H}) \in E\), if \((G, H) \sim (\hat{G}, \hat{H})\), Lemma A.9 guarantees that \( \Phi_{\text{feas}}(G, H) = \Phi_{\text{feas}}(\hat{G}, \hat{H}) \). Then using Theorem A.8 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

\[
\text{Meas}\left(\{(G, H) \in X : \|^F(G, H) - \Phi_{\text{feas}}(G, H)^2 < 1/2\} \right) \leq \text{Meas}(X \setminus E) < \epsilon.
\]

\( \square \)

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

**Proof of Theorem A.3** \( \mathcal{D} \) is finite and 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.2 using Lemma A.9 and Theorem A.8.

**Proof of Theorem A.4** \( \Phi_{\text{obj}} \) is measurable by Lemma A.6, which implies that \( \mathbb{I}_{\Phi_{\text{obj}}(\cdot)} \in \mathcal{R} \) and \( \Phi_{\text{obj}} \), respectively.

**Proof of Theorem A.5** \( \mathcal{D} \) is compact and \( \mathbb{I}_{\Phi_{\text{obj}}(\cdot)} \in \mathcal{R} \) 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 \( \Phi_{\text{obj}} \) restricted on \( \mathcal{D} \cap \Phi_{\text{obj}}^{-1}(\mathbb{R}) \) is continuous. Then one can use Lemma A.9 and Theorem A.8 to obtain the desired results.

Now we consider the optimal solution mapping \( \Phi_{\text{solu}} \) and Theorem A.4 for proving which one needs a universal approximation result of \( \mathcal{F}_{\text{GNN}} \) for equivariant functions. For stating the theorem rigorously, we define another equivalence relationship on \( \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W : (G, H) \sim (\hat{G}, \hat{H}) \) 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 \( \mathcal{F}_{\text{GNN}} \) is stated as follows, which guarantees that \( \mathcal{F}_{\text{GNN}} \) 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.10** (Theorem E.1 in [12]). 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 \( \Phi \in C(X, \mathbb{R}^n) \) satisfies the followings:
(i) For any \( \sigma_V \in S_m, \sigma_W \in S_n, \) and \((G, H) \in X\), it holds that
\[
\Phi((\sigma_V, \sigma_W) \ast (G, H)) = \sigma_W(\Phi(G, H)).
\]

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

(iii) Given any \((G, H) \in X\) and any \( j, j' \in \{1, 2, \ldots, n\}\), if \( C_j^l \overset{W}{\sim} C_{j'}^l \) 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 \( F_W \in \mathcal{F}_G^{\mathrm{GNN}} \) such that
\[
\sup_{(G,H) \in X} \| \Phi(G, H) - F_W(G, H) \| < \epsilon.
\]

Similar to Theorem A.8 Theorem A.10 is also true for all topological linear spaces \( \mathcal{H}^V \) and \( \mathcal{H}^W \). We refer to [5, 18] for other results on the closure of equivariant GNNs. Now we can prove Theorem A.4 and Corollary 4.4.

**Proof of Theorem A.4** We can assume that \( X \subset \mathcal{D}_{\text{solv}} \backslash \mathcal{D}_{\text{foldable}} \subset \mathcal{G}_{m,n} \times \mathcal{H}^V \mathcal{H}^W \backslash \mathcal{D}_{\text{foldable}} \) is close under the action of \( S_m \times S_n \), otherwise one can replace \( X \) with \( \bigcup_{(\sigma_V, \sigma_W) \in S_m \times S_n} (\sigma_V, \sigma_W) \ast X \). The solution mapping \( \Phi_{\text{solv}} : \mathcal{D}_{\text{solv}} \backslash \mathcal{D}_{\text{foldable}} \) is measurable by Lemma \( \text{C.5} \). According to Lusin’s theorem, there exists a compact \( E' \subset X \subset \mathcal{D}_{\text{solv}} \backslash \mathcal{D}_{\text{foldable}} \) with \( \text{Meas}(X \backslash E') < \epsilon/|S_m \times S_n| \) such that \( \Phi_{\text{solv}}|_{E'} \) is continuous. Let us set
\[
E = \bigcap_{(\sigma_V, \sigma_W) \in S_m \times S_n} (\sigma_V, \sigma_W) \ast E' \subset X,
\]
which is compact and is closed under the action of \( S_m \times S_n \). Furthermore, it holds that
\[
\text{Meas}(X \backslash E) = \text{Meas} \left( X \backslash \bigcap_{(\sigma_V, \sigma_W) \in S_m \times S_n} (\sigma_V, \sigma_W) \ast E' \right) \leq \sum_{(\sigma_V, \sigma_W) \in S_m \times S_n} \text{Meas}(X \backslash (\sigma_V, \sigma_W) \ast E') = |S_m \times S_n| \cdot \text{Meas}(X \backslash E') < \epsilon.
\]

For any \((G, H), (\hat{G}, \hat{H}) \in E\), if \((G, H) \overset{W}{\sim} (\hat{G}, \hat{H})\), similar to Lemma A.9 we know that there exists \( \sigma_V \in S_m \) such that \( (\sigma_V, \text{Id}) \ast (G, H) = (\hat{G}, \hat{H}) \). Then by the construction of \( \Phi_{\text{solv}} \) in Section C one has that \( \Phi_{\text{solv}}(G, H) = \Phi_{\text{solv}}(\hat{G}, \hat{H}) \in \mathbb{R}^n \). Condition (i) in Theorem A.10 is satisfied by the definition of \( \Phi_{\text{solv}} \) (see Section C) and Condition (iii) in Theorem A.10 follows from the fact that WL test yields discrete coloring for any graph in \( \mathcal{G}_{m,n} \mathcal{H}^V \mathcal{H}^W \backslash \mathcal{D}_{\text{foldable}} \). Then applying Theorem A.10 one can conclude that for any \( \delta > 0 \), there exists \( F_W \in \mathcal{F}_G^{\mathrm{GNN}} \) with
\[
\sup_{(G,H) \in E} \| F_W(G, H) - \Phi_{\text{solv}}(G, H) \| < \delta.
\]

Therefore, it holds that
\[
\text{Meas} \left( \{ (G, H) \in X : \| F_W(G, H) - \Phi_{\text{solv}}(G, H) \| > \delta \} \right) \leq \text{Meas}(X \backslash E) < \epsilon,
\]
which completes the proof. \( \square \)
Proof of Theorem 4.4. Without loss of generality, we can assume that the finite dataset $\mathcal{D} \subset \mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}}$ is closed under the action of $S_m \times S_n$. Otherwise, we can replace $\mathcal{D}$ using $S_m(\sigma_V, \sigma_W) \in S_m \times S_n \ast \mathcal{D}$.

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.4 with Theorem A.10. □

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 the following concepts are modified as follows in this section due to the random feature technique:

- The space of MILP-graph. Each vertex in the MILP-graph is equipped with an additional feature and the entire random vector $\omega$ is sampled from $\omega \in \Omega := [0, 1]^m \times [0, 1]^n$. More specifically, we write:

$$\omega := (\omega^V, \omega^W) := (\omega^V_1, \cdots, \omega^V_m, \omega^W_1, \cdots, \omega^W_n).$$

Equipped with a random feature, the new vertex features are defined with:

$$h^V_{i,R} := (h^V_i, \omega^V_i), \ h^W_{j,R} := (h^W_j, \omega^W_j), \ \forall \ 1 \leq i \leq m, 1 \leq j \leq n.$$ 

The corresponding vertex feature spaces are defined with:

$$\mathcal{H}^V_m := \mathcal{H}^V_m \times [0, 1]^m = (\mathcal{H}^V \times [0, 1])^m, \ \mathcal{H}^W_n := \mathcal{H}^W_n \times [0, 1]^n = (\mathcal{H}^W \times [0, 1])^n.$$ 

The corresponding graph space is $\mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \times \Omega$, and it is isomorphic to the space defined in the main text:

$$\mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \times \Omega \cong \mathcal{G}_{m,n} \times \mathcal{H}^V_{m,R} \times \mathcal{H}^W_{n,R}.$$ 

Note that the two spaces are actually the same, merely defined in different way. In the main text, we adopt the former due to its simplicity. In the proof, we may either use the former or the latter according to the context.

- Topology and measure. We equip $\Omega$ with the standard topology and the Lebesgue measure.

Then the overall space $\mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \times \Omega$ can be naturally equipped with the product topology and the product measure.

- Key mappings. On the space (with random features) $\mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \times \Omega$, the three key mappings can be defined with

$$\Phi_{\text{feas}}(G, H, \omega) := \Phi_{\text{feas}}(G, H), \ \Phi_{\text{obj}}(G, H, \omega) := \Phi_{\text{obj}}(G, H), \ \Phi_{\text{solu}}(G, H, \omega) := \Phi_{\text{solu}}(G, H),$$

for all $\omega \in \Omega$. This is due to the fact that the feasibility, objective and optimal solution only depend on the MILP instance and they are independent of the random features appended.
• Invariance and equivariance. For any permutations $\sigma_V \in S_m$ and $\sigma_W \in S_n$, $(\sigma_V, \sigma_W) \ast (G, H, \omega)$ denotes the reordered MILP-graph (equipped with random features). We say a function $F_R$ is permutation invariant if

$$F_R((\sigma_V, \sigma_W) \ast (G, H, \omega)) = F_R(G, H, \omega),$$

for all $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H, \omega) \in G_{m,n} \times H_{m}^V \times H_{m}^W$. And we say a function $F_{W,R}$ is permutation equivariant if

$$F_{W,R}((\sigma_V, \sigma_W) \ast (G, H, \omega)) = \sigma_W(F_{W,R}(G, H, \omega)),$$

for all $\sigma_V \in S_m$, $\sigma_W \in S_n$, and $(G, H, \omega) \in G_{m,n} \times H_{m}^V \times H_{m}^W$. It’s clear that $\Phi_{\text{feas}}$ and $\Phi_{\text{soln}}$ is permutation equivariant. Furthermore, it holds that any $F_R \in F_{GNN}$ is invariant and any $F_{W,R} \in F_{GNN}$ is equivariant.

• The WL test. In the WL test, vertex features $h^V_i, h^W_j$ are replaced with $h^{V,R}_i, h^{W,R}_j$, respectively, and the corresponding vertex colors are denoted by $C^{V,R}_i, C^{W,R}_j$ in the $l$-th iteration.

Initialization: $C^{0,V,R}_i = \text{HASH}_{0,V}(h^{V,R}_i), \quad C^{0,W,R}_j = \text{HASH}_{0,W}(h^{W,R}_j)$.

(B.2) For $l = 1, 2, \cdots, L$, $C^{l,V,R}_i = \text{HASH}_{l,V} \left( C^{l-1,V,R}_i, \sum_{j=1}^{n} E_{i,j} \text{HASH}_{l,W} \left( C^{l-1,W,R}_j \right) \right)$.

For $l = 1, 2, \cdots, L$, $C^{l,W,R}_j = \text{HASH}_{l,W} \left( C^{l-1,W,R}_j, \sum_{i=1}^{m} E_{i,j} \text{HASH}_{l,V} \left( C^{l-1,V,R}_i \right) \right)$.

Note that the update form of (B.2) is equal to that of Algorithm 1 and the only difference is the initial vertex features.

• The equivalence relationship. Now let us define the equivalence relationships $\sim$ and $\sim_W$ on the space equipped with random features $G_{m,n} \times H_{m}^V \times H_{m}^W \times \Omega$. Given two MILP-graphs with random features $(G, H, \omega)$ and $(\hat{G}, \hat{H}, \hat{\omega})$, the outcomes of the WL test are $\{(C^{L,V,R}_i)\}_{i=0}^{m}, \{(C^{L,W,R}_j)\}_{j=0}^{n}$ and $\{(\hat{C}^{L,V,R}_i)\}_{i=0}^{m}, \{(\hat{C}^{L,W,R}_j)\}_{j=0}^{n}$, respectively. We denote $(G, H, \omega) \sim (\hat{G}, \hat{H}, \hat{\omega})$ if

$$\{(C^{L,V,R}_i)\}_{i=0}^{m} = \{(\hat{C}^{L,V,R}_i)\}_{i=0}^{m} \text{ and } \{(C^{L,W,R}_j)\}_{j=0}^{n} = \{(\hat{C}^{L,W,R}_j)\}_{j=0}^{n},$$

for all for any $L \in \mathbb{N}$ and any hash functions. We say $(G, H, \omega) \sim_W (\hat{G}, \hat{H}, \hat{\omega})$ if $(G, H, \omega) \sim (\hat{G}, \hat{H}, \hat{\omega})$ and in addition, $C^{L,W,R}_j = \hat{C}^{L,W,R}_j$, $\forall j \in \{1, 2, \ldots, n\}$, for any $L \in \mathbb{N}$ and any hash functions.

• Foldability. Suppose the vertex colors corresponding to $(G, H, \omega)$ are $C^{L,V,R}_i, C^{0,W,R}_j$. Then we say $(G, H, \omega)$ is foldable if there exist $1 \leq i, i' \leq m$ or $1 \leq j, j' \leq n$ such that

$$C^{l,V,R}_i = C^{l,V,R}_{i'}, \quad i \neq i', \quad \text{or} \quad C^{l,W,R}_j = C^{l,W,R}_{j'}, \quad j \neq j'.$$

for all $l \in \mathbb{N}$ and any hash functions. By merely replacing $h^{V}_i, h^{W}_j$ with $h^{V,R}_i, h^{W,R}_j$, we define all the required preliminaries. According to the remarks following Theorem A.8, Lemma A.9, and Theorem A.10 we are able
to directly apply these theorems and lemmas on the space $G_{m,n} \times H_{m}^{V,R} \times H_{n}^{W,R}$, where $H_{m}^{V,R}$ corresponds to $H_{m}^{V}$ and $H_{n}^{W,R}$ corresponds to $H_{n}^{W}$.

**Proof of Theorem 5.1.** Define

$$\Omega_{F} = \{ \omega \in \Omega : \exists i \neq i' \in \{1, 2, \ldots, m\}, \text{ s.t. } \omega_i^V = \omega_{i'}^V,$$

$$\text{or } \exists j \neq j' \in \{1, 2, \ldots, n\}, \text{ s.t. } \omega_j^W = \omega_{j'}^W \}.$$  

(B.3)

Given any $(G, H) \in G_{m,n} \times H_{m}^{V} \times H_{n}^{W}$, as long as $\omega \notin \Omega_{F}$, different vertices in $V$ or $W$ are equipped with different vertex features:

$$h_{i}^{V,R} \neq h_{i'}^{V,R}, \forall i \neq i',$$

$$h_{j}^{W,R} \neq h_{j'}^{W,R}, \forall j \neq j'$$

and consequently, each of the vertices possesses an unique initial color:

$$C_{i}^{0,V,R} \neq C_{i'}^{0,V,R}, \forall i \neq i', \quad C_{j}^{0,W,R} \neq C_{j'}^{0,W,R}, \forall j \neq j',$$

for some $\text{HASH}_{0,V}$ and $\text{HASH}_{0,W}$. In another word, although $(G, H)$ may be foldable, $(G, H, \omega)$ must be unfoldable as long as $\omega \notin \Omega_{F}$.

Therefore, for any $(G, H, \omega), (\hat{G}, \hat{H}, \hat{\omega}) \in G_{m,n} \times H_{m}^{V} \times H_{n}^{W} \times (\Omega \backslash \Omega_{F})$, it holds that both $(G, H, \omega)$ and $(\hat{G}, \hat{H}, \hat{\omega})$ are unfoldable. Applying Lemma A.9 we have $(G, H, \omega) \sim (\hat{G}, \hat{H}, \hat{\omega})$ 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 $\sigma_{W} \in S_{n}$. The invariance property of $\Phi_{\text{feas}}$ implies that

$$\Phi_{\text{feas}}(G, H, \omega) = \Phi_{\text{feas}}(\hat{G}, \hat{H}, \hat{\omega}), \forall (G, H, \omega) \sim (\hat{G}, \hat{H}, \hat{\omega}) \in G_{m,n} \times H_{m}^{V} \times H_{n}^{W} \times (\Omega \backslash \Omega_{F}).$$

In addition, the probability of $\omega \in \Omega_{F}$ is zero due to the uniform distribution of $\omega : P(\Omega_{F}) = 0$. For any $\epsilon > 0$, there exists a compact subset $\Omega_{c} \subset \Omega \backslash \Omega_{F}$ such that

$$P(\Omega \backslash \Omega_{c}) = P((\Omega \backslash \Omega_{F}) \backslash \Omega_{c}) < \epsilon.$$ 

Note that $D \times \Omega_{c}$ is also compact and that $\Phi_{\text{feas}}$ is continuous on $D \times \Omega_{c}$. The continuity of $\Phi_{\text{feas}}|_{D \times \Omega_{c}}$ follows from the condition that $D$ is a finite dataset and that the fact that any mapping with discrete finite domain is continuous (note that $\Phi_{\text{feas}}(G, H, \omega)$ is independent of $\omega \in \Omega_{c}$). Applying Theorem A.8 for $\Phi_{\text{feas}}$ and $D \times \Omega_{c} \subset G_{m,n} \times H_{m}^{V,R} \times H_{n}^{W,R}$, one can conclude the existence of $F_{R} \in F_{\text{GNN}}^{R}$ with

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

Since $\Phi_{\text{feas}}$ is independent of the random features $\omega$, the above equation can be rewritten as

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

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

$$P\left(I_{F_{R}(G,H)>1/2} \neq \Phi_{\text{feas}}(G, H)\right) \leq P(\Omega \backslash \Omega_{c}) < \epsilon.$$ 

**Proof of Theorem 5.2.** The results can be obtained by applying similar techniques as in the proof of Theorem 5.1 for $I_{\Phi_{\text{obj}}(\cdot)} \in \mathbb{R}$ and $\Phi_{\text{obj}}$. □
Proof of Theorem 5.3. Without loss of generality, we can assume that \( D \) is closed under the action of \( S_m \times S_n \); otherwise, we can use \( \{(\sigma_V, \sigma_W) * (G, H) : (G, H) \in D, \sigma_V \in S_m, \sigma_W \in S_n\} \) instead of \( D \). Let \( \Omega_F \subset \Omega \) be the set defined in \( \text{B.3} \) which is clearly closed under the action of \( S_m \times S_n \). There exists a compact \( \Omega'_e \subset \Omega \setminus \Omega_F \) with \( \mathbb{P}(\Omega \setminus \Omega'_e) = \mathbb{P}(\Omega \setminus \Omega'_e) < \epsilon/|S_m \times S_n| \).

Define

\[
\Omega_e = \bigcap_{(\sigma_V, \sigma_W) \in S_m \times S_n} (\sigma_V, \sigma_W) * \Omega'_e \subset \Omega \setminus \Omega_F,
\]

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

\[
\mathbb{P}(\Omega \setminus \Omega_e) \leq \mathbb{P}(\Omega \setminus \bigcap_{(\sigma_V, \sigma_W) \in S_m \times S_n} (\sigma_V, \sigma_W) * \Omega'_e) \leq \sum_{(\sigma_V, \sigma_W) \in S_m \times S_n} \mathbb{P}(\Omega \setminus (\sigma_V, \sigma_W) * \Omega'_e) = |S_m \times S_n| \cdot \mathbb{P}(\Omega \setminus \Omega'_e) < \epsilon.
\]

In addition, \( D \times \Omega_e \) 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.10 for \( \Phi_{\text{sou}} \) and \( D \times \Omega_e \). Condition (i) holds automatically by the definition of the optimal solution mapping. For Condition (ii), given any \((G, H, \omega) \in D \times \Omega_e \) with \((G, H, \omega) \in \mathcal{W} \) \((G, \hat{H}, \hat{\omega}) \), since graphs with vertex features in \( D \times \Omega_e \subset \mathcal{G}_{m,n} \times \mathcal{H}_V^m \times \mathcal{H}_W^n \times (\Omega \setminus \Omega_F) \) cannot be folded, similar to Lemma A.9, one can conclude that \((G, \hat{H}, \hat{\omega}) = (\sigma_V, \text{Id}) * (G, H, \omega) \) for some \( \sigma_V \in S_n \), which leads to \( \Phi_{\text{sou}}(G, H) = \Phi_{\text{sou}}(G, \hat{H}) \). Condition (iii) also follows from the fact that any graph in \( D \times \Omega_e \) cannot be folded and WL test yields discrete coloring. Therefore, Theorem A.10 applies and there exists \( F_{W,R} \in \mathcal{F}^{W,R}_{\text{GNN}} \) such that

\[
\sup_{(G, H, \omega) \in D \times \Omega_e} \|F_{W,R}(G, H, \omega) - \Phi_{\text{sou}}(G, H)\| < \delta,
\]

which implies that

\[
\mathbb{P}(\|F_{W,R}(G, H) - \Phi_{\text{sou}}(G, H)\| > \delta) \leq \mathbb{P}(\Omega \setminus \Omega_e) < \epsilon, \quad \forall (G, H) \in D.
\]

\[\square\]

Remark B.1. A more deeper observation is that, Theorem 5.1, Theorem 5.2, and Theorem A.3 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 [12, Appendix C]), and that one iteration in WL test suffices to yield a discrete coloring for MILP-graphs in \( \mathcal{G}_{m,n} \times \mathcal{H}_V \times \mathcal{H}_W \times (\Omega \setminus \Omega_F) \).

Appendix C. The Optimal Solution Mapping \( \Phi_{\text{sou}} \)

In this section, we define the equivariant optimal solution mapping \( \Phi_{\text{sou}} \), and prove the measurability. The definition consists of several steps.
C.1. The sorting mapping. We first define a sorting mapping:

\[ \Phi_{\text{sort}} : \mathcal{G}_{m,n} \times \mathcal{H}^V_m \times \mathcal{H}^W_n \setminus \mathcal{D}_{\text{foldable}} \to S_n, \]

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 \((b_i, o_i), (b_{i'}, o_{i'}) \in \mathcal{H}^V = \mathbb{R} \times \{\leq, =, \geq\}\), we say \((b_i, o_i) < (b_{i'}, o_{i'})\) if one of the followings holds:
- \(b_i < b_{i'}\).
- \(b_i = b_{i'}\) and \(o_i < o_{i'}\), where \(o(\leq) = -1, o(\geq) = 1\).

(ii) For any \((c_j, l_j, u_j, r_j), (c_{j'}, l_{j'}, u_{j'}, r_{j'}) \in \mathcal{H}^W = \mathbb{R} \times (\mathbb{R} \cup \{-\infty\}) \times (\mathbb{R} \cup \{+\infty\}) \times \{0, 1\}\), we say \((c_j, l_j, u_j, r_j) < (c_{j'}, l_{j'}, u_{j'}, r_{j'})\) 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 \(r_j < r_{j'}\).

**Definition C.2.** Let \(X = \{\{x_1, x_2, \ldots, x_k\}\}\) and \(X' = \{\{x'_1, x'_2, \ldots, x'_{k'}\}\}\) be two multi-sets 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 lexicographically:

(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 \( \Phi_{\text{sort}} \) in Algorithm 2.

**Algorithm 2** The sorting mapping \( \Phi_{\text{sort}} \)

**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. \hspace{1em} Refine the ordering on \( V \) and \( W \) according to Definition C.3.
4. end for
5. Return \( \sigma_W \in S_n \) such that \( w_{\sigma_W(1)} < w_{\sigma_W(2)} < \cdots < w_{\sigma_W(n)} \).

**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 \((G, H) \notin \mathcal{D}_{\text{foldable}}\).

The sorting mapping \( \Phi_{\text{sort}} \) has some straightforward properties:

- \( \Phi_{\text{sort}} \) is equivariant:
  \[ \Phi_{\text{sort}}((\sigma_V, \sigma_W) \circ (G, H)) = (\sigma_V, \sigma_W) \circ \Phi_{\text{sort}}(G, H) = \sigma_W \circ \Phi_{\text{sort}}(G, H), \]
  for any \( \sigma_V \in \mathcal{S}_m, \sigma_W \in \mathcal{S}_n \), and \((G, H) \in \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \setminus \mathcal{D}_{\text{foldable}}\). 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.

- \( \Phi_{\text{sort}} \) is measurable, where the range \( S_n \) is equipped with the discrete measure. This is because that \( \Phi_{\text{sort}} \) is defined via finitely many comparisons.

**C.2. The optimal solution mapping.** We then define the optimal solution mapping

\[ \Phi_{\text{solu}} : \mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}} \to \mathbb{R}^n, \]

based on \( \Phi_{\text{sort}} \), where

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

with \( \mathcal{H}_n^W = (\mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \{0, 1\})^n \subset \mathcal{H}_n^W \), 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 \((G, H) \in \mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}}\), 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 \((G, H) \in \mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}}\), 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 \( \mathbb{R}^d \): \( x \preceq x' \) if and only if \( x_{\sigma(1)} x_{\sigma(2)} \cdots x_{\sigma(n)} \preceq x'_{\sigma(1)} x'_{\sigma(2)} \cdots x'_{\sigma(n)} \) in the sense of lexicographic order. For any \((G, H) \in \mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}} \subset \mathcal{G}_{m,n} \times \mathcal{H}_m^V \times \mathcal{H}_n^W \setminus \mathcal{D}_{\text{foldable}}\), as in Section C.1, the permutation \( \Phi_{\text{sort}}(G, H) \in \mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}} \).
$S_n$ 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 $\preceq$, 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:

$$\Phi_{\text{solu}}(G, H)_{\sigma(1)} = \inf \{ x_{\sigma(1)} : x \in X_{\text{solu}}(G, H) \},$$

and

$$\Phi_{\text{solu}}(G, H)_{\sigma(j)} = \inf \{ x_{\sigma(j)} : x \in X_{\text{solu}}(G, H), x_{\sigma(j')} = \Phi_{\text{solu}}(G, H)_{\sigma(j')}, j' = 1, 2, \ldots, j \},$$

for $j = 2, 3, \ldots, n$. 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.,

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

We then show the measurability of $\Phi_{\text{solu}}$.

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

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

$$\Phi_j : \iota^{-1}(\mathcal{D}_{\text{solu}} \setminus \mathcal{D}_{\text{foldable}}) \cap (\Phi_{\text{sort}} \circ \iota)^{-1}(\sigma) \to \mathbb{R},$$

$$(A, b, c, l, u) \mapsto \Phi_{\text{solu}}(\iota(A, b, c, l, u))_{\sigma(j)};$$

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

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

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

Define

$$V_{\text{feas}}(A, b, c, l, u, x) = \max \left\{ \max_{1 \leq i \leq k_1} \left( \sum_{j' = 1}^n A_{i,j'}x_{j'} - b_i \right), \max_{k_1 < i \leq k_2} \left( \sum_{j' = 1}^n A_{i,j'}x_{j'} - b_i \right) \right\},$$

$$\max_{k_2 < i \leq m} \left( b_i - \sum_{j' = 1}^n A_{i,j'}x_{j'} \right), \max_{1 \leq j' \leq n} (l_{j'} - x_{j'}), \max_{1 \leq j' \leq n} (x_{j'} - u_{j'}) \right\},$$

and

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

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 $V_{\text{feas}}$ 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, $V_{\text{solu}}$ is measurable with respect to $(A, b, c, l, u)$ for
any \( x \), by the measurability of \( \Phi_{\text{obj}} \) (see Lemma 6) and continuity of \( V_{\text{feas}} \), and is continuous with respect to \( x \).

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

- \( \Phi_{\text{solu}}(\iota(A, b, c, l, u)) < \phi \).
- \( \inf \{ x_{\sigma(1)} : x \in X_{\text{solu}}(\iota(A, b, c, l, u)) \} < \phi \).
- There exists \( r \in \mathbb{N}_+ \) and \( x \in X_{\text{solu}}(\iota(A, b, c, l, u)) \), such that \( x_{\sigma(1)} \leq \phi - 1/r \).
- There exists \( r \in \mathbb{N}_+ \), for any \( r' \in \mathbb{N}_+ \), \( x_{\sigma(1)} \leq \phi - 1/r \) 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

\[
\Phi_j^{-1}(-\infty, \phi) = \iota^{-1}(D_{\text{solu}} \setminus D_{\text{foldable}}) \cap (\Phi_{\text{sort}} \circ \iota)^{-1}(\sigma) \cap \bigcup_{r \in \mathbb{N}_+} \bigcap_{r' \in \mathbb{N}_+} \bigcup_{x \in \mathbb{Q}^k \times \mathbb{Z}^{n-k}} : V_{\text{solu}}(A, b, c, l, u, x) < 1/r' \bigg\}
\]

is measurable, which implies the measurability of \( \Phi_1 \).

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

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

for \((A, b, c, l, u) \in \iota^{-1}(D_{\text{solu}} \setminus D_{\text{foldable}}) \cap (\Phi_{\text{sort}} \circ \iota)^{-1}(\sigma)\) and \( x \in \mathbb{R}^k \times \mathbb{Z}^{n-k} \). Similar to \( V_{\text{solu}} \), \( V_{\text{solu}}^j \) is also measurable respect to \((A, b, c, l, u)\) for any \( x \) and is continuous with respect to \( x \).

For any \((A, b, c, l, u) \in \iota^{-1}(D_{\text{solu}} \setminus D_{\text{foldable}}) \cap (\Phi_{\text{sort}} \circ \iota)^{-1}(\sigma)\), the followings are equivalent:

- \( \Phi_{\text{solu}}(\iota(A, b, c, l, u))_{\sigma(j)} < \phi \).
- \( \inf \{ x_{\sigma(j)} : x \in X_{\text{solu}}(\iota(A, b, c, l, u)) \}, x_{\sigma(j')} = \Phi_{j'}(A, b, c, l, u), j' = 1, 2, \ldots, j - 1 \} \leq \phi \).
- 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 \( x_{\sigma(j)} \leq \phi - 1/r \).
- There exists \( r \in \mathbb{N}_+ \), for any \( r' \in \mathbb{N}_+ \), \( x_{\sigma(j)} \leq \phi - 1/r \) 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. \( \square \)

**Appendix D. Details of the Numerical Experiments and Extra Experiments**

**MILP instance generation.** Each instance in \( D_1 \) has 20 variables, 6 constraints and is generated with:

- For each variable, \( c_j \sim \mathcal{N}(0, 0.01), l_j, u_j \sim \mathcal{N}(0, 10) \). 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, \( o_i \sim \mathcal{U}([\leq, =, \geq]) \) 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 ($1 \leq k \leq 500$)

- 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 $x_{j_1} + x_{j_2} = 1$, $x_{j_2} + x_{j_3} = 1$, $x_{j_3} + x_{j_4} = 1$, $x_{j_4} + x_{j_5} = 1$, $x_{j_5} + x_{j_6} = 1$, $x_{j_6} + x_{j_4} = 1$.

**MLP architectures.** As we mentioned in the main text, all the learnable functions in GNN are taken as MLPs. All the learnable functions $f^{V}_{in}, f^{W}_{in}, f^{out}, f^{W}_{out}, \{f^{V}_{l}, f^{W}_{l}, g^{V}_{l}, g^{W}_{l}\}_{l=0}^{L}$ 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 [26] 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.

**Extra experiments on generalization.** We conduct some numerical experiments on generalization for the random-feature approach. The GNN/MLP architecture is the same as what we describe before with the embedding size being $d = 8$. The size of the training set is chosen from $\{10, 100, 1000\}$ and the size of the testing set is 1000, where all instances are generated in the same way as $\mathcal{D}_2$, with the only difference that we set $c_1 = \cdots = c_{20} = 0.01$ instead of $c_1 = \cdots = c_{20} = 0$. The reason is that, $c_1 = \cdots = c_{20} = 0$ makes every feasible solution as optimal and thus the labels depend on the solver’s choice. By setting $c_1 = \cdots = c_{20} = 0.01$, one can make the label depend only on the MILP problem itself, which fits our purpose, i.e., representing characteristics of MILP problems, better. In addition, our datasets for feasibility consists of 50% feasible instances and 50% infeasible ones, while other datasets are obtained by removing infeasible samples.

We report the error on training set and the testing set in Table 1. The generalization performance is good in our setting when the training size is relatively large. Although our experiments are still small-scale, these results indicate the potential of the random-feature approach in solving MILP problems in practice.

| Number of Training Samples | 10  | 100 | 1000 |
|----------------------------|-----|-----|------|
| Training Error             | 0   | 0   | 0.02 |
| Testing Error              | 0.289 | 0.104 | 0.022 |

**Table 1.** Generalization for feasibility
### Table 2. Generalization for optimal objective value

| Number of Training Samples | 10    | 100   | 1000  |
|---------------------------|-------|-------|-------|
| Training Error            | 1.2e-15 | 1.1e-06 | 1.1e-06 |
| Testing Error             | 5.9e-03 | 5.4e-06 | 1.1e-06 |

### Table 3. Generalization for optimal solution

| Number of Training Samples | 10    | 100   | 1000  |
|---------------------------|-------|-------|-------|
| Training Error            | 1.7e-03 | 7.2e-02 | 7.2e-02 |
| Testing Error             | 2.1e-01 | 7.5e-02 | 7.3e-02 |

(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