REFORMULATION-CONTRASTIVE LEARNING FOR MIXED INTEGER PROGRAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixed-integer linear programs (MILP) model many real-world decision problems, motivating machine-learning methods that exploit recurring structure to accelerate MILP solving. MILPs can admit many equivalent formulations: integrality-preserving changes of variables and the addition of redundant constraints can alter their formulations while preserving the optimization problem. We leverage these reformulations as a source of self-supervision for learning general-purpose representations of MILP variables and constraints. We characterize the affine reformulations that are valid for every input instance, and distinguish re-descriptions, which leave variables unchanged, from substitutions, which transform them predictably. Building on equivariant self-supervised learning, we introduce ReMILP (reformulation-contrastive MILP representation learning), which jointly trains a graph neural network and a hypernetwork to predict how variable embeddings transform under changes of variables. Without solver-derived labels, ReMILP learns representations that exhibit the intended invariance and equivariance on unseen problem classes. Across binary solution, constraint activity and integrality gap prediction, these representations carry task-relevant information when frozen and provide a useful initialization for fine-tuning.

1 INTRODUCTION

Mixed-integer linear programs (MILPs) are a flexible modeling framework for challenging decision problems arising in domains such as energy, transportation and logistics. Solving large MILPs remains computationally demanding, motivating a growing body of work that integrates machine learning into the solving pipeline, for instance to guide branching or to predict high-quality partial solutions (Gasse et al., 2019; Paulus et al., 2022; Huang et al., 2024a). These approaches motivate the development of general-purpose MILP representations that capture reusable problem structure beyond a single solver task.

A mixed-integer linear program has no canonical formulation. The same problem can be expressed using different units, binary conventions or variable parametrizations. The resulting formulations have different coefficients and bounds, but their solution sets are related by an affine map. This type of equivalence has been leveraged for contrastive representation learning on other structured modalities, including source code (Jain et al., 2021), boolean satisfiability (Duan et al., 2022), and logic circuits (Wang et al., 2025). We extend this idea to MILPs, using equivalent reformulations as a source of self-supervision to pre-train an encoder that produces variable and constraint representations reusable across downstream MILP tasks. The closest self-supervised pre-training approach, FORGE (Shafi & Kadioglu, 2026), instead learns MILP representations through vector-quantized graph reconstruction. Reformulations have also been used for contrastive learning in MILP branching (Lin et al., 2024; Lu et al., 2026), but only as an auxiliary to supervised imitation learning and with transformations chosen to preserve the branching target. In contrast, we use reformulation equivalence itself as the supervision signal and consider a broader family of transformations.

This broader family introduces an important subtlety. *Re-descriptions* alter the constraint system while leaving the variables unchanged, so their embeddings should remain invariant. *Substitutions*, i.e. type-preserving affine changes of variables, instead modify the variables themselves, so their embeddings should transform predictably. A standard contrastive objective would enforce invariance to both. We therefore adopt equivariant self-supervised learning (Garrido et al., 2023), requiring variable

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

embeddings to be *invariant* to re-descriptions and *equivariant* to substitutions. We characterize all valid affine reformulations and introduce ReMILP, which jointly learns a MILP encoder and the action of local substitutions on its variable embeddings, without solver-derived labels.

Our contributions are as follows:

- We give necessary and sufficient conditions for a substitution or a re-description to be valid on every instance (Theorems 4.1 and 4.2), and show that every valid affine augmentation rule is a composition of both (Theorem 4.3). In particular, integer variables admit arbitrary unimodular changes of variable, and redundant constraints can be constructed from any nonnegative combination of existing constraints with loosened right-hand sides.
- We introduce ReMILP (reformulation-contrastive MILP representation learning), which jointly trains a graph neural network to produce variable embeddings that are invariant to re-descriptions and equivariant to substitutions, and a hypernetwork to predict how these embeddings transform under the corresponding changes of variables.
- We validate the learned invariance and equivariance and evaluate downstream transfer on binary solution, constraint activity, and integrality gap prediction across Distributional MIPLIB (Huang et al., 2024b) and MILP-Evolve (Li et al., 2025). In both frozen and fine-tuned regimes, we compare ReMILP with FORGE (Shafi & Kadioglu, 2026) and random or training-from-scratch baselines.

2 Related work

Learning for MILP solving. Graph neural networks operating on the MILP representation of Gasse et al. (2019) have been used to learn heuristic components of branch-and-bound solvers, such as branching policies (Gasse et al., 2019; Gupta et al., 2020; Parsonson et al., 2023) or cut selection rules (Paulus et al., 2022; Wang et al., 2023). Another line of work predicts properties of optimal solutions to restrict the search space before or during solving. Most approaches predict binary variable values at an optimum and use confident predictions to delimit the solver's search region (Ding et al., 2020; Nair et al., 2020; Han et al., 2023; Huang et al., 2024a; Liu et al., 2025). Similarly, on the dual side, inequality constraints predicted to be tight at an optimum are replaced by equalities (Li et al., 2026). These approaches are supervised and task specific. We use binary solution, constraint activity and integrality gap prediction as downstream tasks to assess both the information captured by pre-trained representations and their value as an initialisation for fine-tuning.

MILP representation learning. Few works aim to learn MILP representations that are both general-purpose and independent of solver labels. Multi-task and cross-distribution encoders share representations across tasks or domains, but are still trained on solver-derived labels (Cai et al., 2025; Pu et al., 2025). FORGE (Shafi & Kadioglu, 2026) is, to our knowledge, the only self-supervised method for general-purpose MILP representations to date, and relies on a vector-quantized graph reconstruction objective. We pursue the same goal, but derive the self-supervision signal from equivalent reformulations.

Contrastive learning with augmented MILPs. Prior work uses reformulations to augment strong branching imitation learning, adding a contrastive term to the supervised loss. Lin et al. (2024) shift variables to generate instances that preserve strong-branching labels and contrast the graph-level embeddings. Lu et al. (2026) use sign flips, integer shifts, and redundant constraints obtained by summing pairs of constraints. Their contrastive loss does not directly pair reformulations. Instead, it clusters Branch-and-bound nodes by features and pulls together nodes within each cluster. Beyond these practical augmentations, they prove that any block-triangular change of variables with a signed permutation integer block and an invertible continuous block provides an equivalent instance.

Equivariant self-supervised learning. The tension between invariance and equivariance has been studied in visual self-supervised learning, where invariant objectives may discard information about the transformations relating two views. Dangovski et al. (2022) retain such information by predicting the applied transformation from the transformed embedding, making the representation sensitive to it without prescribing how it should change. Devillers & Lefort (2023) and Garrido et al. (2023) instead learn the action itself: the former conditions a linear predictor on the embedding and transformation parameters, while the latter maps these parameters to a hypernetwork that outputs the predictor

weights, preventing them from being ignored. We adopt the hypernetwork formulation because it separates transformation-specific actions from what is shared across transformations. Unlike these works, we use no separate invariance term: a single contrastive objective conditioned on the substitution enforces equivariance to substitutions and invariance to re-descriptions.

3 PRELIMINARIES

3.1 BACKGROUND AND NOTATION

Fix $n \geq 1$ and $0 \leq k \leq n$, and let $\mathbb{V} \coloneqq \mathbb{R}^{n-k} \times \mathbb{Z}^k$ be the space of our decision variables, with $n-k$ continuous variables, and k integer ones. We introduce the notation $[p] \coloneqq \{1, \dots, p\}$ and denote by $\boldsymbol{e}_p$ the p-th element of the canonical basis of $\mathbb{R}^n$ or $\mathbb{R}^m$ depending on the context: $\boldsymbol{e}_p = (0, \dots, 0, 1, 0, \dots, 0)^\top$ where the unique 1 lies at position p.

Definition 3.1 (MILP instance)**.** A *mixed integer linear programming instance* over $\mathbb{V}$ with $m \geq 2$ constraints is defined as a tuple:

$$M = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \in \mathcal{M}_m \coloneqq \mathbb{R}^{m \times n} \times \mathbb{R}^m \times \mathbb{R}^n \tag{1}$$

representing the following optimization problem, wherein a linear objective $\boldsymbol{x} \mapsto \boldsymbol{c}^\top \boldsymbol{x}$ is minimized under linear inequality constraints ($\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$), with the k last variables being integer valued ($x \in \mathbb{V}$):

$$\min_{\boldsymbol{x} \in \mathbb{V}} \boldsymbol{c}^\top \boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}. \tag{2}$$

We will refer to $F(M) \coloneqq \{\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\} \cap \mathbb{V}$ as the feasible region of M. M is said to be *feasible* if its feasible region is nonempty, and we designate by $\mathrm{OPT}(M) \in \mathbb{R} \cup \{-\infty\}$ its optimal value in that case. It is additionally said to be *bounded* if its optimal value is bounded.

3.2 EQUIVALENCE–PRESERVING DATA AUGMENTATIONS

Definition 3.2 (Equivalence)**.** Instances $M = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \in \mathcal{M}_m$ and $M' = (\boldsymbol{A}', \boldsymbol{b}', \boldsymbol{c}') \in \mathcal{M}_{m'}$ are *equivalent* if there are an affine bijection φ of $\mathbb{R}^n$ and $\gamma \in \mathbb{R}$ such that:

- $\varphi(F(M)) = F(M')$, and
- $\boldsymbol{c}'^\top \varphi(\boldsymbol{x}) = \boldsymbol{c}^\top \boldsymbol{x} + \gamma$ for all $\boldsymbol{x} \in F(M)$.

In that case, we call the pair (φ, γ) a *witness* for the equivalence of M and M', and refer to M' as a *reformulation* of M.

Equivalent instances are either both infeasible, both unbounded, or both have bounded optimal solutions. In the latter case, φ maps the optimal solutions of M onto those of M', and $\mathrm{OPT}(M') = \mathrm{OPT}(M) + \gamma$.

Definition 3.3 (Augmentation rules)**.** An *augmentation rule* of shape $m \to m'$ is a map $\Phi : \mathcal{M}_m \to \mathcal{M}_{m'}$. It is *valid* if $\Phi(M)$ is equivalent to M for every $M \in \mathcal{M}_m$, and it is affine if every component of $\Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ is an affine function of the components of $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$.

A data augmentation pipeline acting on instances of all sizes can be viewed as a family of rules, with one rule for each $m \geq 2$. The pipeline is valid if and only if every rule in this family is valid. We can therefore study augmentation rules of a fixed shape without loss of generality.

These definitions come with some restrictions: two instances can't be equivalent if their feasible regions do not live on the same space $\mathbb{V}$, which means that augmentations that increase or decrease the number of variables in an instance[1] do not fit into our framework.

3.3 BIPARTITE GRAPH ENCODING

Following Gasse et al. (2019), we represent an instance $M = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \in \mathcal{M}_m$ as a bipartite graph $\mathcal{G} = (V \sqcup C, E)$, with one node $i \in V$ per variable and one node $j \in C$ per constraint. Variable i

[1] An example of such an augmentation would be breaking an integer variable bounded by N into $\lceil \log_b(N) \rceil$ integer variables using its base b representation.

has features $(c_i, \mathbf{1}_{i>n-k})$, constraint j has feature b_j, and $E = \{(i,j) : A_{j,i} \neq 0\}$, with edge feature $A_{j,i}$. A GNN $f_{\boldsymbol{\theta}}$ maps this graph to node embeddings. We denote by $\boldsymbol{h}_i(M) \in \mathbb{R}^d$ the embedding of variable i and collect the variable embeddings in the matrix

$$\boldsymbol{H}_{\boldsymbol{\theta}}(M) := [\boldsymbol{h}_1(M), \ldots, \boldsymbol{h}_n(M)]^\top \in \mathbb{R}^{n \times d}. \tag{3}$$

4 METHOD

We learn general-purpose MILP representations from equivalent formulations. We first show that valid affine reformulations decompose into re-descriptions, which leave variables unchanged, and substitutions, which transform them predictably. ReMILP uses this structure as a self-supervised inductive bias, training variable embeddings to be invariant to re-descriptions and equivariant to substitutions. ReMILP realizes this principle with a contrastive objective that, without solver-derived labels, pre-trains an encoder to capture reusable MILP structure before task-specific adaptation.

4.1 VALID AUGMENTATION RULES

Two important families of data augmentation rules satisfying Definition 3.3 are *substitutions* and *re-descriptions*. A *substitution* $\mathsf{S}(\boldsymbol{T}, \boldsymbol{q})$, with $\boldsymbol{T} \in \mathrm{GL}_n(\mathbb{R})$ and $\boldsymbol{q} \in \mathbb{R}^n$, applies the change of variables $\boldsymbol{x} = \boldsymbol{T}\boldsymbol{y} + \boldsymbol{q}$ to instances with any number of constraints:

$$\mathsf{S}(\boldsymbol{T}, \boldsymbol{q}) : \ (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \longmapsto \big(\boldsymbol{A}\boldsymbol{T}, \ \boldsymbol{b} - \boldsymbol{A}\boldsymbol{q}, \ \boldsymbol{T}^\top \boldsymbol{c}\big). \tag{4}$$

A *constraint rewriting* $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{g})$, with $\boldsymbol{\Lambda} \in \mathbb{R}^{m' \times m}$ and an arbitrary map $\boldsymbol{g} : \mathbb{R}^m \to \mathbb{R}^{m'}$, rewrites the constraint system and keeps variables and objective:

$$\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{g}) : \ (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \longmapsto \big(\boldsymbol{\Lambda}\boldsymbol{A}, \ \boldsymbol{g}(\boldsymbol{b}), \ \boldsymbol{c}\big). \tag{5}$$

We define a *re-description* as a constraint rewriting that preserves the feasible set of every instance. Every re-description is a valid rule and the converse holds for affine $\boldsymbol{g}$.

Theorem 4.1 (Substitutions)**.** *The substitution $\mathsf{S}(\boldsymbol{T}, \boldsymbol{q})$ is valid if and only if $\boldsymbol{q} \in \mathbb{V}$ and*

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_{\mathrm{CC}} & \boldsymbol{T}_{\mathrm{CI}} \\ \boldsymbol{0} & \boldsymbol{T}_{\mathrm{II}} \end{pmatrix}, \qquad \boldsymbol{T}_{\mathrm{CC}} \in \mathrm{GL}_{n-k}(\mathbb{R}), \quad \boldsymbol{T}_{\mathrm{CI}} \in \mathbb{R}^{(n-k)\times k}, \quad \boldsymbol{T}_{\mathrm{II}} \in \mathrm{GL}_k(\mathbb{Z}).$$

As such, the admissible transformations of integer variables are not limited to the shifts and signed permutations studied by Lin et al. (2024) and Lu et al. (2026), but include arbitrary unimodular maps.

Theorem 4.2 (Re-descriptions)**.** *The constraint rewriting $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{g})$ is a re-description if and only if $\boldsymbol{\Lambda} \geq 0$ and, for every $\boldsymbol{b} \in \mathbb{R}^m$,*

- *$\boldsymbol{g}(\boldsymbol{b}) \geq \boldsymbol{\Lambda}\boldsymbol{b}$, and*
- *for all $p \in [m]$, there are $i \in [m']$ and $\theta > 0$ such that $\boldsymbol{\lambda}_i^\top = \theta \boldsymbol{e}_p^\top$ and $g_i(\boldsymbol{b}) = \theta b_p$.*

For affine $\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}$ this means that up to row permutation, $\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Pi} \\ \boldsymbol{N} \end{pmatrix}$ and $\boldsymbol{s} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{s}' \end{pmatrix}$ where $\boldsymbol{\Pi}$ is a positive generalized permutation matrix, $\boldsymbol{N} \geq 0$ and $\boldsymbol{s}' \geq 0$.

A re-description therefore has to keep a positively rescaled copy of every constraint, and may additionally append nonnegative combinations of constraints with loosened right-hand sides.

A composite of a valid substitution and a re-description is itself a valid augmentation rule:

$$\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}) \circ \mathsf{S}(\boldsymbol{T}, \boldsymbol{q}) : \ (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \longmapsto \big(\boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{T}, \ \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s} - \boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{q}, \ \boldsymbol{T}^\top \boldsymbol{c}\big). \tag{6}$$

Theorem 4.3 (Completeness)**.** *An affine rule $\Phi : \mathcal{M}_m \to \mathcal{M}_{m'}$ is valid if and only if $\Phi = \mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}) \circ \mathsf{S}(\boldsymbol{T}, \boldsymbol{q})$ with $(\boldsymbol{T}, \boldsymbol{q})$ as in Theorem 4.1 and $(\boldsymbol{\Lambda}, \boldsymbol{s})$ as in Theorem 4.2.*

Thus, Theorems 4.1–4.3 characterize all affine augmentation rules valid for every MILP instance: each decomposes into a re-description and a substitution. Proofs are provided in Section A. This characterization provides a natural inductive bias for MILP representation learning: knowing how each part of a reformulation acts on the problem tells us how embeddings should respond to it.

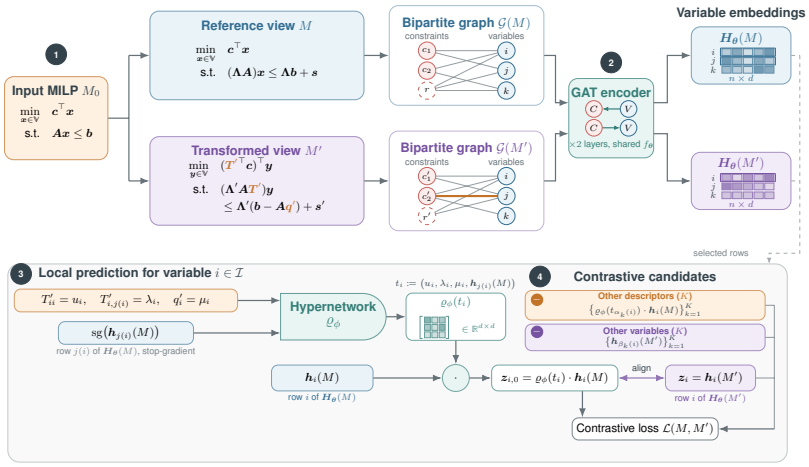


Figure 1: **ReMILP pre-training overview**. **(1)** Independent re-descriptions (Theorem 4.2) form two views of the same MILP instance M_0, with a substitution (Theorem 4.1) applied to the second view (10). **(2)** A shared two-layer GAT encodes their bipartite graphs. **(3)** The hypernetwork uses local substitution parameters and the variable partner embedding to produce a linear operator predicting $\boldsymbol{h}_i(M')$ from $\boldsymbol{h}_i(M)$ (11). **(4)** The pre-training contrastive loss makes this prediction the preferred match to $\boldsymbol{h}_i(M')$ over predictions from other descriptors and embeddings of other variables (13).

4.2 Invariance and Equivariance

As introduced in Section 3.3, a GNN maps each instance M to its matrix of variable embeddings $\boldsymbol{H}_{\boldsymbol{\theta}}(M)$. We use the reformulations characterized above to define an inductive bias on these embeddings. Valid re-descriptions do not modify an instance's feasible region nor its optimal solutions, so we encourage *invariance*:

$$\boldsymbol{H}_{\boldsymbol{\theta}}\left(\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{g}) \cdot M\right) = \boldsymbol{H}_{\boldsymbol{\theta}}(M). \tag{7}$$

Substitutions, via variable scaling or transvection, instead change the feasible region and optimal solutions in a known way. The variable embeddings should thus be sensitive to substitutions, which we express through *equivariance*:

$$\boldsymbol{H}_{\boldsymbol{\theta}}\left(\mathsf{S}(\boldsymbol{T}, \boldsymbol{q}) \cdot M\right) = \rho(\boldsymbol{T}, \boldsymbol{q}) \cdot \boldsymbol{H}_{\boldsymbol{\theta}}(M), \tag{8}$$

where ρ maps each substitution $(\boldsymbol{T}, \boldsymbol{q})$ to a linear operator predicting how the variable embeddings transform. Intuitively, ρ can be seen as a representation of the group of valid substitutions, though we require it neither to be a homomorphism nor to output invertible matrices. These transformation laws encourage the representation space to capture MILP-specific structure without solver-derived labels.

4.3 Reformulation-Contrastive Learning

We turn the invariance and equivariance inductive biases into a self-supervised objective using pairs of reformulated MILP instances. Figure 1 summarizes ReMILP pre-training. Since general substitutions can densify the MILP graph and make GNN inference prohibitively expensive, we restrict the substitution matrix T to be sparse.

Concretely, we partition the variables into two sets $[n] = \mathcal{I} \sqcup \mathcal{J}$ and transform only variables in $\mathcal{I}$. For each $i \in \mathcal{I}$, we choose a partner $j(i) \in \mathcal{J}$ and apply the local substitution

$$x_i \leftarrow u_i x_i + \lambda_i x_{j(i)} + \mu_i. \tag{9}$$

These local substitutions together define a sparse matrix $\boldsymbol{T}'$ and a translation vector $\boldsymbol{q}'$: $T'_{ii} = u_i$, $T'_{i,j(i)} = \lambda_i$, and $q'_i = \mu_i$ for $i \in \mathcal{I}$. For $j \in \mathcal{J}$, $T'_{jj} = 1$ and $q'_j = 0$. All other components of $\boldsymbol{T}'$ are zero. We choose the parameters to satisfy the conditions of Theorem 4.1. In particular, if x_i is an integer, $u_i \in \{-1, 1\}$ and the only nontrivial substitution we apply to a binary variable is $x_i \leftarrow 1 - x_i$. The way these augmentations are sampled is detailed in Section D.5.

Given an input instance M_0, we construct two views. For the first, we apply a re-description to M_0. For the second, we first apply the substitution $\mathsf{S}(\boldsymbol{T}', \boldsymbol{q}')$, then independently sample a re-description of the resulting instance. Each re-description keeps the existing constraints and adds redundant constraints formed from random nonnegative combinations with relaxed right-hand sides. The matrices $\boldsymbol{\Lambda}, \boldsymbol{\Lambda}'$ specify the row combinations, and $\boldsymbol{s}, \boldsymbol{s}'$ specify the bound relaxations. Thus,

$$\begin{aligned} M &= \mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}) \cdot M_0, \\ M' &= \mathsf{R}(\boldsymbol{\Lambda}', \boldsymbol{\Lambda}'\boldsymbol{b} + \boldsymbol{s}') \circ \mathsf{S}(\boldsymbol{T}', \boldsymbol{q}') \cdot M_0. \end{aligned} \tag{10}$$

We predict the embedding of variable i in M' from its embedding in M. Rather than conditioning on the full substitution, we use its local parameters (u_i, λ_i, μ_i) together with the embedding $\boldsymbol{h}_{j(i)}(M)$ of the partner variable involved in the local substitution of x_i. Inspired by the hypernetwork approach of Garrido et al. (2023), we map these local inputs to an affine operator that predicts the transformed embedding of variable i:

$$\boldsymbol{h}_i(M') \approx \varrho_\phi\big(u_i, \lambda_i, \mu_i, \mathrm{sg}\big(\boldsymbol{h}_{j(i)}(M)\big)\big) \cdot \boldsymbol{h}_i(M), \tag{11}$$

where sg denotes stop-gradient. We write $t_i \coloneqq (u_i, \lambda_i, \mu_i, \mathrm{sg}(\boldsymbol{h}_{j(i)}(M)))$ for the hypernetwork's input. More details about the hypernetwork are available in Section D.4.

We adapt the normalized temperature-scaled cross-entropy (NT-Xent) objective (Chen et al., 2020) to align $\varrho_\phi(t_i) \cdot \boldsymbol{h}_i(M)$ with $\boldsymbol{h}_i(M')$ for each $i \in \mathcal{I}$. We use two types of negatives: predictions obtained with an incorrect transformation descriptor and embeddings of incorrect variables in the transformed view. For $K \geq 1$, we sample indices $\alpha_1(i), \ldots, \alpha_K(i)$ uniformly from $\mathcal{I} \setminus \{i\}$ and independently sample indices $\beta_1(i), \ldots, \beta_K(i)$ uniformly from $[n] \setminus \{i\}$. Let $\boldsymbol{z}_i \coloneqq \boldsymbol{h}_i(M')$ be the anchor. Its positive and the two sets of negatives are

$$\boldsymbol{z}_{i,k} \coloneqq \begin{cases} \varrho_\phi(t_i) \cdot \boldsymbol{h}_i(M) & \text{if } k = 0, \\ \varrho_\phi\left(t_{\alpha_k(i)}\right) \cdot \boldsymbol{h}_i(M) & \text{if } 1 \leq k \leq K, \\ \boldsymbol{h}_{\beta_{k-K}(i)}(M') & \text{if } K+1 \leq k \leq 2K. \end{cases} \tag{12}$$

Denoting cosine similarity by $\mathrm{sim}(\cdot, \cdot)$, our contrastive loss is

$$\mathcal{L}(M, M') = -\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \log \left(\frac{\exp\left(\mathrm{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_{i,0}\right)/\tau\right)}{\sum_{k=0}^{2K} \exp\left(\mathrm{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_{i,k}\right)/\tau\right)} \right). \tag{13}$$

By jointly training $f_{\boldsymbol{\theta}}$ and ϱ_ϕ, this objective encourages variable embeddings to be stable under re-descriptions and to transform predictably under substitutions, as we show empirically in Section 5.2.1.

5 Empirical Evaluation

We evaluate the representations learned by ReMILP on several classes and difficulties of MILP instances to investigate whether the learned invariance and equivariance to relevant reformulations provide a useful prior over the space of MILPs for downstream tasks.

5.1 Experimental setup

Baselines We compare ReMILP with FORGE†, our re-implementation of the vector-quantized graph autoencoder of Shafi & Kadioglu (2026), and a randomly initialized encoder. All three use the same graph encoder backbone in order to fairly assess the contribution of each pre-training method.

Pre-training protocol We use 29 000 MILP-Evolve instances (Li et al., 2025) to pre-train the encoder. Both ReMILP and FORGE† are pre-trained for 20 000 steps with a learning rate of 10^{-3}. The training procedure is described in more detail in Section D.2.

Architecture The graph encoder uses two layers of four-head GATv2 attention (Brody et al., 2022) with 128-dimensional hidden states on the MILP bipartite graph. Each layer updates constraints from variables, then variables from constraints (Gasse et al., 2019). Jumping Knowledge (Xu et al., 2018) combines layer representations.

Downstream tasks We evaluate the learned representations on three tasks: integrality-gap prediction at the instance level, binary-solution prediction at the variable level, and constraint-activity prediction at the constraint level. For integrality-gap prediction, we use 12 500 held-out MILP-Evolve instances (Li et al., 2025). For the binary solutions and constraint activity tasks, we use ten problem classes from the Distributional MIPLIB (D-MIPLIB) dataset (Huang et al., 2024b), comprising 23 instance families. We describe the datasets in more detail in Section C.

Downstream training and tuning We use two-layer MLP prediction heads for all downstream tasks. For binary-solution and constraint-activity prediction, the head maps each variable or constraint embedding, respectively, to a node-level prediction. For integrality-gap prediction, it maps an instance embedding obtained by attention pooling over both node types to a scalar. Using task-specific labels, we consider two settings: *frozen*, where only the prediction head is trained, and *fine-tuned*, where the head and graph encoder are trained jointly. Further details are provided in Section D.

Metrics For the integrality-gap prediction task, the target is $\frac{|\mathrm{LP}-\mathrm{IP}|}{\max\{|\mathrm{LP}|,|\mathrm{IP}|,10^{-5}\}}$, where LP and IP are the respective optima of the LP relaxation and the MILP. We report test mean absolute error (MAE). For the node-level tasks, we compute a pool of high-quality solutions for each instance. For binary-solution prediction, the target for each binary variable is the weighted fraction of solutions in which it equals one, capturing how often it is selected among good solutions. For constraint-activity prediction, the target is the weighted fraction of solutions in which each constraint is binding. More details are provided in Section C. We evaluate both tasks using the Kullback-Leibler (KL) divergence between these target probabilities and the predictions. To aggregate across instance families, we divide each model's test KL on an instance family by that of a baseline on the same family and take the geometric mean of these ratios. The baseline is the random encoder in the frozen setting and the supervised encoder trained from scratch in the fine-tuned setting.

5.2 Results

The introduction motivates ReMILP as a way to learn general-purpose MILP representations from equivalent formulations. We therefore ask whether the learned representations (i) retain their invariance and equivariance on unseen classes, (ii) carry task-relevant information when frozen, and (iii) provide a useful initialization for fine-tuning.

5.2.1 Invariance and equivariance on unseen classes

We perform nearest-neighbor retrieval between each D-MIPLIB test instance and one of its reformulations. For invariance, we apply a re-description to each test instance. For every variable, we use its original embedding as a query, rank all variable embeddings in the re-described instance by cosine similarity, and record the rank of the corresponding variable. For equivariance, we instead apply a substitution. For each substituted variable, we again rank all embeddings in the reformulated instance, but use as the query either its original embedding, corresponding to an invariant prediction, or the embedding predicted by ReMILP's hypernetwork with input the applied substitution. We record the rank of the corresponding substituted variable. Table 1 reports the average Top-1 retrieval accuracy and mean reciprocal rank (MRR) across all 23 D-MIPLIB test splits.

Table 1: **Invariance and equivariance properties.** Nearest-neighbor retrieval after re-description (invariance) or substitution (equivariance) on D-MIPLIB. For equivariance, "+ hyper" uses the hypernetwork prediction. Other rows use the original embedding. Mean ± sd over 3 seeds.

	Invariance ↑		Equivariance ↑	
Encoder	Top-1	MRR	Top-1	MRR
Random	0.289 ± 0.009	0.375 ± 0.011	0.091 ± 0.011	0.121 ± 0.013
FORGE†	0.120 ± 0.040	0.171 ± 0.049	0.017 ± 0.003	0.026 ± 0.004
ReMILP (ours)	0.383 ± 0.093	0.454 ± 0.089	0.081 ± 0.014	0.105 ± 0.016
ReMILP+hyper (ours)	–	–	$\mathbf{0.162 \pm 0.005}$	$\mathbf{0.276 \pm 0.004}$

Under re-description, ReMILP achieves the strongest retrieval performance, indicating that its embeddings are more stable to changes in constraint description. It improves slightly over the random encoder, while FORGE† performs worse than random. Under substitution, treating the embeddings as invariant performs poorly, whereas the hypernetwork substantially improves retrieval. Together, these results support the intended distinction on unseen classes: re-descriptions should preserve embeddings, while substitutions should transform them predictably.

5.2.2 Task-relevant information in frozen representations

We now ask whether pre-training produces representations that are directly useful for downstream prediction, before any task-specific adaptation of the encoder. We therefore evaluate the encoders in the frozen regime. For the node-level tasks, we report pure ILP and MILP families separately to examine whether this transfer depends on the presence of continuous variables. Tables 2 and 3 summarize the results.

Both pre-trained encoders carry more task-relevant information than the random encoder baseline, whose frozen embeddings are random projections of each node's two-hop neighborhood. This indicates that self-supervised pre-training extracts structure that is useful for downstream prediction even without adapting the encoder. Beyond this common benefit, ReMILP improves over FORGE† on both binary solution prediction and integrality-gap prediction, with the largest margin on the latter: ReMILP achieves a 24.5% lower error. Since FORGE† uses its own pooler based on codeword histograms, we additionally evaluate its frozen instance-level representations using the same attention pooler as Random and ReMILP. Its performance remains essentially unchanged, which confirms that the observed difference is not an artifact of the pooling mechanism.

The node-level results reveal a further distinction across tasks and problem types. On binary solution prediction, ReMILP reduces the normalized KL divergence relative to FORGE† by 3.4% on pure ILPs and 8.9% on MILP families. In contrast, the two encoders are indistinguishable within one standard deviation on constraint-activity prediction. These results are consistent with the structure of our pre-training objective: constraint embeddings are shaped only indirectly through message passing. The larger gain on MILPs is also consistent with richer substitutions for continuous variables.

Table 2: **Downstream node-level performance under frozen encoders**. Test KL divergence normalized by the corresponding random encoder result for each instance family and aggregated using the geometric mean. Values below 1 indicate improvement over the random encoder (↓).

Task	Type	ReMILP (ours)	FORGE†
Binary solution	ILP	$\mathbf{0.887 \pm 0.016}$	0.918 ± 0.007
	MILP	$\mathbf{0.688 \pm 0.050}$	0.755 ± 0.012
Constraint activity	ILP	0.871 ± 0.007	0.882 ± 0.018
	MILP	0.855 ± 0.028	0.863 ± 0.024

5.2.3 Pre-trained encoders as an initialization

When the encoder is fine-tuned jointly with the head, Figures 2 and 3 show two effects: faster convergence and, for some problem classes, better final performance. On pure ILP families, the benefit is primarily optimization speed. Both pre-trained encoders outperform training from scratch early in fine-tuning, but all three models are within 1% of each other by the end of the fine-tuning budget. Pre-training thus provides a better starting point on these instances, without improving the final solution reached by supervised training. In contrast, on MILP families the advantage persists throughout the fine-tuning budget on both tasks. This effect is clearest for MILP binary solution prediction: ReMILP reaches the final performance of the encoder trained from scratch after only 6 400 steps and continues improving beyond it, whereas FORGE† reaches the same level later, after 13 000 steps. Constraint activity follows the same pattern, with a smaller final gap.

Integrality-gap prediction provides a strong difference in the fine-tuning regime. ReMILP reaches the final MAE of the encoder trained from scratch after 19 500 of the 100 000 fine-tuning steps and ultimately improves on it by 16%, whereas FORGE† converges to a higher value. Moreover, fine-tuning changes ReMILP only marginally relative to its frozen performance (0.040 ± 0.002

to 0.038 ± 0.001), while training the encoder from a random initialization provides a substantial improvement over the corresponding frozen encoder. This suggests that much of the information required for integrality-gap prediction is already present in ReMILP's pre-trained representations, with fine-tuning primarily refining rather than constructing it.

The role of pre-training depends on the downstream regime. On pure ILPs, it mainly accelerates supervised learning. On mixed-integer and instance-level tasks, ReMILP can additionally provide task-relevant structure that remains beneficial after full fine-tuning.

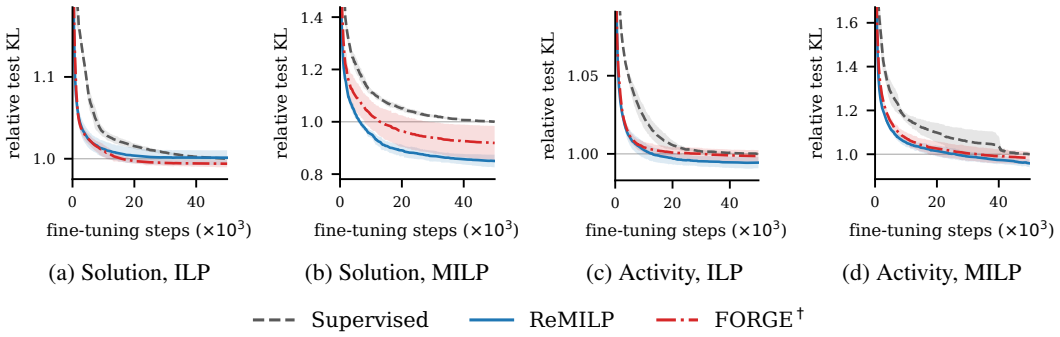


Figure 2: **Fine-tuning on binary solution and constraint activity prediction.** Test KL throughout fine-tuning, aggregated by geometric mean across the corresponding D-MIPLIB families and normalized by the final performance of the encoder trained from scratch ("Supervised").

Table 3: **Integrality gap under frozen encoders**. Test MAE on MILP-Evolve.

Encoder	Test MAE ↓
Random	0.063 ± 0.008
ReMILP (ours)	$\mathbf{0.040 \pm 0.002}$
FORGE†	0.053 ± 0.007
FORGE† + attn.	0.054 ± 0.002

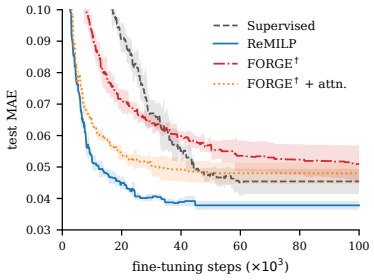


Figure 3: **Fine-tuning on integrality gap.** Test MAE on MILP-Evolve

6 Conclusion

We introduced a theory-driven approach to self-supervised representation learning for MILPs. We characterized all valid affine augmentation rules, showing that they decompose into re-descriptions and substitutions. This characterization extends prior MILP augmentations to richer variable transformations, including integer transvections, and provides a natural inductive bias for variable embeddings: invariance to re-description and equivariance to substitutions. ReMILP instantiates this principle by pre-training on equivalent reformulation pairs. A contrastive objective jointly trains the encoder and hypernetwork to predict how local substitutions transform variable embeddings, without solver-derived labels. Empirically, ReMILP's learned representations exhibit stronger invariance and equivariance than the baselines on unseen classes, retain task-relevant information when frozen, and provide a useful initialisation for fine-tuning. These results suggest that known reformulation behaviors can structure the latent space in ways that benefit several downstream MILP tasks. More broadly, this work points to a promising direction for MILP representation learning, where known problem structure shapes latent representations through equivariant self-supervision.

AI USE STATEMENT

In this work, we used generative AI tools in developing the proofs of Theorems 4.1 to 4.3. Several key results were first proven by AI, and then verified and simplified by the authors. LLMs were additionally used in code generation, as well as drafting and editing parts of this paper. We have reviewed all AI-assisted work, and we take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

While it is not our immediate focus, our work may contribute to improving the scalability of exact MILP solvers, which support decision-making in sectors such as energy, transportation, healthcare and defense. In such a scenario, faster solving could reduce operational costs and resource consumption in these applications. As with any optimization tool, its societal effect would depend on the objectives and constraints chosen by those who deploy it, and we do not foresee risks specific to our method.

REPRODUCIBILITY STATEMENT

In order to ensure that our results are easy to reproduce, we make the processed datasets we use in our experiments available at https://huggingface.co/datasets/anon5090/remilp-data and we include a reproducible Pixi (Arts et al.) environment alongside our source code which can be found in the supplementary materials. This minimizes the time needed for environment setup and data processing. We also detail our implementation in Section 5.1 and Section D.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Ruben Arts, Bas Zalmstra, Wolf Vollprecht, Tim de Jager, Nichita Morcotilo, and Julian Hofer. pixi. URL https://github.com/prefix-dev/pixi/releases/tag/v0.81.0.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Junyang Cai, Taoan Huang, and Bistra Dilkina. Multi-task representation learning for mixed integer linear programming. In Guido Tack (ed.), *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 134–151, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-95973-8.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljacic. Equivariant self-supervised learning: Encouraging equivariance in representations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gKLAAfiytI.

Alexandre Devillers and Mathieu Lefort. Equimod: An equivariance module to improve visual instance discrimination. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=eDLwjKmtYFt.

Jian-Ya Ding, Chao Zhang, Lei Shen, Shengyin Li, Bing Wang, Yinghui Xu, and Le Song. Accelerating primal solution findings for mixed integer programs based on solution prediction. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pp. 1452–1459, 2020.

Haonan Duan, Pashootan Vaezipoor, Max B Paulus, Yangjun Ruan, and Chris Maddison. Augment with care: Contrastive learning for combinatorial problems. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5627–5642. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/duan22b.html.

Quentin Garrido, Laurent Najman, and Yann Lecun. Self-supervised learning of split invariant equivariant representations. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10975–10996. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/garrido23b.html.

Maxime Gasse, Didier Chetelat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/d14c2267d848abeb81fd590f371d39bd-Paper.pdf.

Prateek Gupta, Maxime Gasse, Elias Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18087–18097. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d1e946f4e67db4b362ad23818a6fb78a-Paper.pdf.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL https://www.gurobi.com.

Qingyu Han, Linxin Yang, Qian Chen, Xiang Zhou, Dong Zhang, Akang Wang, Ruoyu Sun, and Xiaodong Luo. A GNN-guided predict-and-search framework for mixed-integer linear programming. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=pHMpgT5xWaE.

Taoan Huang, Aaron M Ferber, Arman Zharmagambetov, Yuandong Tian, and Bistra Dilkina. Contrastive predict-and-search for mixed integer linear programs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 19757–19771. PMLR, 21–27 Jul 2024a. URL https://proceedings.mlr.press/v235/huang24f.html.

Weimin Huang, Taoan Huang, Aaron M Ferber, and Bistra Dilkina. Distributional miplib: a multi-domain library for advancing ml-guided milp methods, 2024b. URL https://arxiv.org/abs/2406.06954.

Paras Jain, Ajay Jain, Tianjun Zhang, Pieter Abbeel, Joseph Gonzalez, and Ion Stoica. Contrastive code representation learning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5954–5971, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.482. URL https://aclanthology.org/2021.emnlp-main.482/.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Sirui Li, Janardhan Kulkarni, Ishai Menache, Cathy Wu, and Beibin Li. Towards foundation models for mixed integer linear programming. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 88590–88638, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/dc594b8712f5d3ea1904092d74739d25-Paper-Conference.pdf.

Yixuan Li, Ran Hou, Yu Ding, Shisi Guan, Jiahui Duan, Xiongwei Han, Tao Zhong, Vincent Chau, Weiwei Wu, Zhiyuan Liu, et al. Constraint matters: Multi-modal representation for reducing mixed-integer linear programming. In *International Conference on Learning Representations*, volume 2026, pp. 22344–22370, 2026.

Jiacheng Lin, Meng XU, Zhihua Xiong, and Huangang Wang. Cambranch: Contrastive learning with augmented milps for branching. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 28544–28567, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/7b39f4512a2e3899edcc59c7501f3cd4-Paper-Conference.pdf.

Haoyang Liu, Jie Wang, Zijie Geng, Xijun Li, Yuxuan Zong, Fangzhou Zhu, Jianye HAO, and Feng Wu. Apollo-milp: An alternating prediction-correction neural solving framework for mixed-integer linear programming. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 94513–94540, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/ebcdb4d0372ebbdd0e6a7dce9662b6b4-Paper-Conference.pdf.

Tongkai Lu, Shuai Ma, and Chongyang Tao. Dynamic stratified contrastive learning with upstream augmentation for MILP branching. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=LDNH3sQtMW.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid Von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

Christopher W F Parsonson, Alexandre Laterre, and Thomas D Barrett. Reinforcement learning for branch-and-bound optimisation using retrospective trajectories. *Proc. Conf. AAAI Artif. Intell.*, 37 (4):4061–4069, June 2023.

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

Tianle Pu, Zijie Geng, Haoyang Liu, Shixuan Liu, Jie Wang, Li Zeng, Chao Chen, and Changjun Fan. Rome: Domain-robust mixture-of-experts for milp solution prediction across domains. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 33633–33661. Curran Associates, Inc., 2025. doi: 10.52202/085713-1131. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3071ed272658c4418309962ba7b94ec8-Paper-Conference.pdf.

Zohair Shafi and Serdar Kadioglu. Forge: Foundational optimization representations from graph embeddings, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=7Uo1yRWUpo.

Zhihai Wang, Xijun Li, Jie Wang, Yufei Kuang, Mingxuan Yuan, Jia Zeng, Yongdong Zhang, and Feng Wu. Learning cut selection for mixed-integer linear programming via hierarchical sequence model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Zob4P9bRNcK.

Ziyi Wang, Chen Bai, Zhuolun He, Guangliang Zhang, Qiang Xu, Tsung-Yi Ho, Yu Huang, and Bei Yu. Fgnn2: A powerful pretraining framework for learning the logic functionality of circuits. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 44(1):227–240, January 2025. ISSN 0278-0070. doi: 10.1109/TCAD.2024.3434464. URL https://doi.org/10.1109/TCAD.2024.3434464.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/xu18c.html.

A OMITTED PROOFS

Throughout this section, n, k, m' and $m \geq 2$ are fixed. Recall that $\boldsymbol{e}_j$ is used to denote the j-th vector of the canonical basis of $\mathbb{R}^n$, $\mathbb{R}^m$, $\mathbb{R}^k$ or $\mathbb{R}^{n-k}$ depending on the context.

The last k of any n coordinates are the integer ones, and a subscript C or I selects the corresponding parts of a vector or matrix. For example, a vector $\boldsymbol{x} \in \mathbb{R}^n$ is decomposed in the following manner: $\boldsymbol{x} = (\boldsymbol{x}_\mathrm{C}, \boldsymbol{x}_\mathrm{I})$ with $\boldsymbol{x}_\mathrm{C} \in \mathbb{R}^{n-k}$ and $\boldsymbol{x}_\mathrm{I} \in \mathbb{R}^k$. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, we decompose it as $\boldsymbol{A} = [\boldsymbol{A}_\mathrm{C}\ \boldsymbol{A}_\mathrm{I}]$ where $\boldsymbol{A}_\mathrm{C} \in \mathbb{R}^{m\times(n-k)}$ and $\boldsymbol{A}_\mathrm{I} \in \mathbb{R}^{m\times k}$, so that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}_\mathrm{C}\boldsymbol{x}_\mathrm{C} + \boldsymbol{A}_\mathrm{I}\boldsymbol{x}_\mathrm{I}$. Similarly, an $n \times n$ matrix $\boldsymbol{T}$ can be decomposed into four blocks:

$$\begin{array}{c} \\ n-k \\ k \end{array} \begin{array}{c} \begin{array}{cc} n-k & k \end{array} \\ \begin{pmatrix} \boldsymbol{T}_\mathrm{CC} & \boldsymbol{T}_\mathrm{CI} \\ \boldsymbol{T}_\mathrm{IC} & \boldsymbol{T}_\mathrm{II} \end{pmatrix} \end{array}$$

With the block notation established, we start off by proving some key lemmas.

A.1 PRELIMINARIES

Lemma A.1 (Affine self-maps of $\mathbb{V}$)**.** *Let $\psi(\boldsymbol{x}) = \boldsymbol{S}\boldsymbol{x} + \boldsymbol{r}$ be an affine map of $\mathbb{R}^n$ with $\psi(\mathbb{V}) = \mathbb{V}$. Then $\boldsymbol{S}_\mathrm{IC} = \boldsymbol{0}$, $\boldsymbol{S}_\mathrm{CC} \in \mathrm{GL}_{n-k}(\mathbb{R})$, $\boldsymbol{S}_\mathrm{II} \in \mathrm{GL}_k(\mathbb{Z})$ and $\boldsymbol{r} \in \mathbb{V}$. In particular ψ is a bijection of $\mathbb{R}^n$. Conversely, every ψ of this form satisfies $\psi(\mathbb{V}) = \mathbb{V}$, and so does ψ^{-1}.*

Proof. For every $\boldsymbol{x}_\mathrm{C} \in \mathbb{R}^{n-k}$ the integer block of $\psi(\boldsymbol{x}_\mathrm{C}, \boldsymbol{0})$, namely $\boldsymbol{S}_\mathrm{IC}\boldsymbol{x}_\mathrm{C} + \boldsymbol{r}_\mathrm{I}$, lies in $\mathbb{Z}^k$. A continuous map from $\mathbb{R}^{n-k}$ to $\mathbb{Z}^k$ is constant, so $\boldsymbol{S}_\mathrm{IC} = \boldsymbol{0}$ and $\boldsymbol{r}_\mathrm{I} \in \mathbb{Z}^k$. The integer block of $\psi(\boldsymbol{0}, \boldsymbol{e}_j)$ is $\boldsymbol{S}_\mathrm{II}\boldsymbol{e}_j + \boldsymbol{r}_\mathrm{I} \in \mathbb{Z}^k$, so $\boldsymbol{S}_\mathrm{II}$ is an integer matrix. For every $\boldsymbol{w} \in \mathbb{V}$ there is $\boldsymbol{x} \in \mathbb{V}$ with $\psi(\boldsymbol{x}) = \boldsymbol{w}$, and then $\boldsymbol{w}_\mathrm{I} = \boldsymbol{S}_\mathrm{II}\boldsymbol{x}_\mathrm{I} + \boldsymbol{r}_\mathrm{I}$. Hence $\boldsymbol{S}_\mathrm{II}\mathbb{Z}^k = \mathbb{Z}^k$, which proves that $\boldsymbol{S}_\mathrm{II}$ is unimodular. Fixing $\boldsymbol{w}_\mathrm{I}$, which fixes $\boldsymbol{x}_\mathrm{I} = \boldsymbol{S}_\mathrm{II}^{-1}(\boldsymbol{w}_\mathrm{I} - \boldsymbol{r}_\mathrm{I})$, and letting $\boldsymbol{w}_\mathrm{C}$ range over $\mathbb{R}^{n-k}$ in $\boldsymbol{w}_\mathrm{C} = \boldsymbol{S}_\mathrm{CC}\boldsymbol{x}_\mathrm{C} + \boldsymbol{S}_\mathrm{CI}\boldsymbol{x}_\mathrm{I} + \boldsymbol{r}_\mathrm{C}$ shows that $\boldsymbol{S}_\mathrm{CC}$ is surjective, hence invertible. Thus $\det \boldsymbol{S} = \det \boldsymbol{S}_\mathrm{CC} \det \boldsymbol{S}_\mathrm{II} \neq 0$ and $\boldsymbol{r} \in \mathbb{V}$.

Conversely, if $\boldsymbol{S}$ and $\boldsymbol{r}$ have the stated form, the integer block of $\psi(\boldsymbol{x})$ is $\boldsymbol{S}_\mathrm{II}\boldsymbol{x}_\mathrm{I} + \boldsymbol{r}_\mathrm{I} \in \mathbb{Z}^k$ for $\boldsymbol{x} \in \mathbb{V}$, so $\psi(\mathbb{V}) \subseteq \mathbb{V}$. The inverse $\psi^{-1}(\boldsymbol{y}) = \boldsymbol{S}^{-1}\boldsymbol{y} - \boldsymbol{S}^{-1}\boldsymbol{r}$ has $\boldsymbol{S}^{-1} = \begin{pmatrix} \boldsymbol{S}_\mathrm{CC}^{-1} & -\boldsymbol{S}_\mathrm{CC}^{-1}\boldsymbol{S}_\mathrm{CI}\boldsymbol{S}_\mathrm{II}^{-1} \\ \boldsymbol{0} & \boldsymbol{S}_\mathrm{II}^{-1} \end{pmatrix}$, of the same form because $\boldsymbol{S}_\mathrm{II}^{-1}$ is again unimodular, and $\boldsymbol{S}^{-1}\boldsymbol{r} \in \mathbb{V}$. Hence also $\psi^{-1}(\mathbb{V}) \subseteq \mathbb{V}$, and $\psi(\mathbb{V}) = \mathbb{V}$. □

Lemma A.2 (Objective rigidity)**.** *Let (φ, γ) witness the equivalence of $M = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ and $M' = (\boldsymbol{A}', \boldsymbol{b}', \boldsymbol{c}')$, with $\varphi(\boldsymbol{x}) = \boldsymbol{S}\boldsymbol{x} + \boldsymbol{r}$, and suppose that $F(M)$ affinely spans $\mathbb{R}^n$. Then $\boldsymbol{S}^\top \boldsymbol{c}' = \boldsymbol{c}$. In particular, if $\boldsymbol{c}' = \boldsymbol{c}$, then $\boldsymbol{c}^\top \varphi(\boldsymbol{x}) = \boldsymbol{c}^\top(\boldsymbol{x} + \boldsymbol{r})$ for all $\boldsymbol{x} \in \mathbb{R}^n$.*

Proof. The affine functions $\boldsymbol{x} \mapsto \boldsymbol{c}'^\top(\boldsymbol{S}\boldsymbol{x} + \boldsymbol{r})$ and $\boldsymbol{x} \mapsto \boldsymbol{c}^\top\boldsymbol{x} + \gamma$ agree on $F(M)$, hence everywhere, since $F(M)$ affinely spans $\mathbb{R}^n$. In particular, $\boldsymbol{S}^\top\boldsymbol{c}' = \boldsymbol{c}$. If $\boldsymbol{c}' = \boldsymbol{c}$ then $\boldsymbol{c}^\top\boldsymbol{S}\boldsymbol{x} = \boldsymbol{c}^\top\boldsymbol{x}$ for all $\boldsymbol{x}$. □

Lemma A.3 (Unconstrained feasible sets)**.** *Let φ be an affine bijection of $\mathbb{R}^n$, and let $\boldsymbol{A}' \in \mathbb{R}^{m'\times n}$, $\boldsymbol{b}' \in \mathbb{R}^{m'}$ be such that $\boldsymbol{A}'\boldsymbol{y} \leq \boldsymbol{b}'$ for every $\boldsymbol{y} \in \varphi(\mathbb{V})$. Then $\boldsymbol{A}' = \boldsymbol{0}$ and $\boldsymbol{b}' \geq 0$. In particular, if an instance M' is equivalent to an instance whose feasible set is $\mathbb{V}$, then $F(M') = \mathbb{V}$.*

Proof. The polyhedron $P' = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{A}'\boldsymbol{y} \leq \boldsymbol{b}'\}$ is convex and contains $\varphi(\mathbb{V})$, so it contains $\operatorname{conv}\varphi(\mathbb{V}) = \varphi(\operatorname{conv}\mathbb{V}) = \varphi(\mathbb{R}^n) = \mathbb{R}^n$. A nonzero row of $\boldsymbol{A}'$ would make P' a proper subset of $\mathbb{R}^n$, so $\boldsymbol{A}' = \boldsymbol{0}$, and then $\boldsymbol{0} \leq \boldsymbol{b}'$. For the second statement, $F(M') = P' \cap \mathbb{V} = \mathbb{V}$. □

A.2 PROOF OF THEOREM 4.1

Theorem 4.1 (Substitutions)**.** *The substitution $\mathsf{S}(\boldsymbol{T}, \boldsymbol{q})$ is valid if and only if $\boldsymbol{q} \in \mathbb{V}$ and*

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_\mathrm{CC} & \boldsymbol{T}_\mathrm{CI} \\ \boldsymbol{0} & \boldsymbol{T}_\mathrm{II} \end{pmatrix}, \qquad \boldsymbol{T}_\mathrm{CC} \in \mathrm{GL}_{n-k}(\mathbb{R}), \quad \boldsymbol{T}_\mathrm{CI} \in \mathbb{R}^{(n-k)\times k}, \quad \boldsymbol{T}_\mathrm{II} \in \mathrm{GL}_k(\mathbb{Z}).$$

Proof. Write $\psi(\boldsymbol{x}) := \boldsymbol{T}^{-1}(\boldsymbol{x} - \boldsymbol{q})$ and $P := \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$. For every instance, substituting $\boldsymbol{x} = \boldsymbol{T}\boldsymbol{y} + \boldsymbol{q}$ gives

$$F\big(\mathsf{S}(\boldsymbol{T}, \boldsymbol{q}) \cdot M\big) = \{\boldsymbol{y} \in \mathbb{V} \mid \boldsymbol{A}(\boldsymbol{T}\boldsymbol{y} + \boldsymbol{q}) \leq \boldsymbol{b}\} = \psi(P) \cap \mathbb{V}. \tag{14}$$

Sufficiency. By Lemma A.1, the map $\boldsymbol{y} \mapsto \boldsymbol{T}\boldsymbol{y} + \boldsymbol{q}$ sends $\mathbb{V}$ onto $\mathbb{V}$, and so does its inverse ψ. Since ψ is a bijection, $\psi(F(M)) = \psi(P \cap \mathbb{V}) = \psi(P) \cap \psi(\mathbb{V}) = \psi(P) \cap \mathbb{V}$, which is $F(\mathsf{S}(\boldsymbol{T}, \boldsymbol{q}) \cdot M)$ by (14). As for the objective value map, we have the following identity:

$$(\boldsymbol{T}^\top \boldsymbol{c})^\top \psi(\boldsymbol{x}) = \boldsymbol{c}^\top(\boldsymbol{x} - \boldsymbol{q}) = \boldsymbol{c}^\top \boldsymbol{x} - \boldsymbol{c}^\top \boldsymbol{q}$$

Necessity of the block form of $\boldsymbol{T}$. For $\boldsymbol{c} \in \mathbb{R}^n$ the unconstrained instance $M_{\boldsymbol{c}} = (\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{c})$ and its image $(\boldsymbol{0}, \boldsymbol{0}, \boldsymbol{T}^\top \boldsymbol{c})$ both have feasible set $\mathbb{V}$. Let $(\varphi_{\boldsymbol{c}}, \gamma_{\boldsymbol{c}})$ be a witness and $\boldsymbol{S}_{\boldsymbol{c}}$ the linear part of $\varphi_{\boldsymbol{c}}$. Then $\varphi_{\boldsymbol{c}}(\mathbb{V}) = \mathbb{V}$, so $\boldsymbol{S}_{\boldsymbol{c}}$ is block upper triangular with a unimodular integer block by Lemma A.1, and $\boldsymbol{S}_{\boldsymbol{c}}^\top \boldsymbol{T}^\top \boldsymbol{c} = \boldsymbol{c}$ by Lemma A.2. As $\boldsymbol{c}$ ranges over $\mathbb{R}^n$ so does $\boldsymbol{d} := \boldsymbol{T}^\top \boldsymbol{c}$; hence for every $\boldsymbol{d} \in \mathbb{R}^n$ there is a block upper triangular $\boldsymbol{S}$ with unimodular integer block such that $\boldsymbol{S}^\top \boldsymbol{d} = \boldsymbol{T}^{-\top} \boldsymbol{d}$. For such $\boldsymbol{S}$ and $\boldsymbol{d} = (\boldsymbol{0}, \boldsymbol{d}_{\mathrm{I}})$ we have $\boldsymbol{S}^\top \boldsymbol{d} = (\boldsymbol{0}, \boldsymbol{S}_{\mathrm{II}}^\top \boldsymbol{d}_{\mathrm{I}})$. Writing $\boldsymbol{T}^{-\top} = \left(\begin{smallmatrix} \boldsymbol{X} & \boldsymbol{Y} \\ \boldsymbol{Z} & \boldsymbol{W} \end{smallmatrix}\right)$, we obtain $\boldsymbol{Y}\boldsymbol{d}_{\mathrm{I}} = \boldsymbol{0}$ for all $\boldsymbol{d}_{\mathrm{I}}$, so $\boldsymbol{Y} = \boldsymbol{0}$, and

$$\mathbb{R}^k = \bigcup_{\boldsymbol{U} \in \mathrm{GL}_k(\mathbb{Z})} \ker\left(\boldsymbol{W} - \boldsymbol{U}^\top\right).$$

A proper linear subspace has Lebesgue measure zero, so a countable union of them is not $\mathbb{R}^k$ and one of these kernels is all of $\mathbb{R}^k$, that is, $\boldsymbol{W} = \boldsymbol{U}^\top$ for some $\boldsymbol{U} \in \mathrm{GL}_k(\mathbb{Z})$. Therefore $\boldsymbol{T}^{-1} = \left(\begin{smallmatrix} \boldsymbol{X}^\top & \boldsymbol{Z}^\top \\ \boldsymbol{0} & \boldsymbol{U} \end{smallmatrix}\right)$, with $\boldsymbol{X}$ invertible because $\boldsymbol{T}$ is, and by Lemma A.1 its inverse $\boldsymbol{T}$ has the same form, with integer block $\boldsymbol{U}^{-1} \in \mathrm{GL}_k(\mathbb{Z})$.

Necessity of $\boldsymbol{q} \in \mathbb{V}$. Let j be an integer coordinate and $\rho \in (0, 1)$. Consider the instance whose first two constraints are $x_j \le \rho$ and $-x_j \le \rho$, with all further constraints (if $m > 2$) equal to $\boldsymbol{0}^\top \boldsymbol{x} \le 0$. Its feasible set contains $\boldsymbol{0}$. By (14) its image has feasible set $\{\boldsymbol{y} \in \mathbb{V} \mid |(\boldsymbol{T}\boldsymbol{y})_j + q_j| \le \rho\}$. By the block form, $(\boldsymbol{T}\boldsymbol{y})_j$ is an integer combination of the entries of $\boldsymbol{y}_{\mathrm{I}}$, hence an integer. If $q_j \notin \mathbb{Z}$ and $\rho < \mathrm{dist}(q_j, \mathbb{Z})$, the image is infeasible, and no bijection maps a nonempty set onto the empty set. Hence $\boldsymbol{q}_{\mathrm{I}} \in \mathbb{Z}^k$, that is, $\boldsymbol{q} \in \mathbb{V}$. □

A.3 Proof of Theorem 4.2

We start off by proving a different necessary and sufficient condition from the one stated in Theorem 4.2, which is expressed in terms of the existence of a map from the new constraint system back to the old one, and then we establish the equivalence of the two.

Lemma A.4. *The constraint rewriting $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{g})$ is a re-description if and only if $\boldsymbol{\Lambda} \ge 0$ and, for every $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{g}(\boldsymbol{b}) \ge \boldsymbol{\Lambda}\boldsymbol{b}$ and there is $\boldsymbol{\Gamma}_{\boldsymbol{b}} \in \mathbb{R}^{m \times m'}$ with $\boldsymbol{\Gamma}_{\boldsymbol{b}} \ge 0$, $\boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{\Lambda} = \boldsymbol{I}_m$ and $\boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{b}$.*

Proof. Write $F = \{\boldsymbol{x} \in \mathbb{V} \mid \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}\}$ and $F' = \{\boldsymbol{x} \in \mathbb{V} \mid \boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{g}(\boldsymbol{b})\}$.

Sufficiency. Fix $(\boldsymbol{A}, \boldsymbol{b})$. If $\boldsymbol{x} \in F$, then $\boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{\Lambda}\boldsymbol{b} \le \boldsymbol{g}(\boldsymbol{b})$ because $\boldsymbol{\Lambda} \ge 0$, so $\boldsymbol{x} \in F'$. If $\boldsymbol{x} \in F'$, then $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{b}$ because $\boldsymbol{\Gamma}_{\boldsymbol{b}} \ge 0$, so $\boldsymbol{x} \in F$.

Necessity. Assume $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{g})$ is a re-description, so that $F = F'$ for every $(\boldsymbol{A}, \boldsymbol{b})$. Each counterexample below exhibits an instance on which the point $\boldsymbol{e}_1 \in \mathbb{V}$ lies in exactly one of F and F'. All of them use a constraint matrix of the form $\boldsymbol{A} = \boldsymbol{u}\,\boldsymbol{e}_1^\top$, the matrix whose first column is $\boldsymbol{u}$ and whose other columns vanish. Notice that such a matrix satisfies $\boldsymbol{A}\boldsymbol{e}_1 = \boldsymbol{u}$.

Nonnegativity of $\boldsymbol{\Lambda}$. Suppose $\Lambda_{i,l} < 0$. Let $\boldsymbol{b} = \boldsymbol{0}$ and $\boldsymbol{u} = -N\boldsymbol{e}_l$ with $N > 0$. Then $F = \{\boldsymbol{x} \in \mathbb{V} \mid x_1 \ge 0\} \ni \boldsymbol{e}_1$, while row i of the new system at $\boldsymbol{e}_1$ reads $|\Lambda_{i,l}|N \le g_i(\boldsymbol{0})$, which fails for large N.

The bound $\boldsymbol{g}(\boldsymbol{b}) \ge \boldsymbol{\Lambda}\boldsymbol{b}$. Suppose $g_i(\boldsymbol{b}) < \boldsymbol{\lambda}_i^\top \boldsymbol{b}$ and let $\boldsymbol{u} = \boldsymbol{b}$. Every original constraint holds with equality at $\boldsymbol{e}_1$, so $\boldsymbol{e}_1 \in F$, but row i of the new constraint system evaluated at $\boldsymbol{e}_1$ reads $\boldsymbol{\lambda}_i^\top \boldsymbol{b} \le g_i(\boldsymbol{b})$, which is a contradiction.

Existence of $\boldsymbol{\Gamma}_{\boldsymbol{b}}$. Fix $\boldsymbol{b}$ and let $\mathcal{Z} = \{i \mid g_i(\boldsymbol{b}) = \boldsymbol{\lambda}_i^\top \boldsymbol{b}\}$ be the rows on which $\boldsymbol{g}(\boldsymbol{b})$ is tight. Row p of the condition $\boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{\Lambda} = \boldsymbol{I}_m$ and $\boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{b}$ holds iff there exists a $\boldsymbol{\gamma} \ge 0$ such that $\boldsymbol{\Lambda}^\top \boldsymbol{\gamma} = \boldsymbol{e}_p$

and $\boldsymbol{\gamma}^\top \boldsymbol{g}(\boldsymbol{b}) = b_p$. For any $\boldsymbol{\gamma}$ with the first two properties, $\boldsymbol{\gamma}^\top \boldsymbol{\Lambda}\boldsymbol{b} = (\boldsymbol{\Lambda}^\top\boldsymbol{\gamma})^\top \boldsymbol{b} = b_p$, so that

$$\boldsymbol{\gamma}^\top \boldsymbol{g}(\boldsymbol{b}) - b_p = \boldsymbol{\gamma}^\top \big(\boldsymbol{g}(\boldsymbol{b}) - \boldsymbol{\Lambda}\boldsymbol{b}\big) = \sum_i \gamma_i \big(g_i(\boldsymbol{b}) - \boldsymbol{\lambda}_i^\top \boldsymbol{b}\big)$$

This is a sum of nonnegative terms. It vanishes if and only if $\gamma_i = 0$ for every $i \notin \mathcal{Z}$. Row p of the condition is therefore equivalent to the existence of $\boldsymbol{\gamma} \geq 0$ supported on $\mathcal{Z}$ with $\boldsymbol{\Lambda}^\top\boldsymbol{\gamma} = \boldsymbol{e}_p$.

Suppose that for some $p \in [m]$ there is no such $\boldsymbol{\gamma}$. By Farkas' lemma, there is $\boldsymbol{y} \in \mathbb{R}^m$ with $\boldsymbol{\lambda}_i^\top \boldsymbol{y} \leq 0$ for all $i \in \mathcal{Z}$ and $y_p > 0$. Let $\boldsymbol{u} = \boldsymbol{b} + \varepsilon \boldsymbol{y}$ with $\varepsilon > 0$. The original constraint p evaluated at $\boldsymbol{e}_1$ reads $b_p + \varepsilon y_p \leq b_p$, which does not hold, so $\boldsymbol{e}_1 \notin F$. On the other hand, the new row i evaluated at $\boldsymbol{e}_1$ reads $\boldsymbol{\lambda}_i^\top \boldsymbol{b} + \varepsilon \boldsymbol{\lambda}_i^\top \boldsymbol{y} \leq g_i(\boldsymbol{b})$. This holds for $i \in \mathcal{Z}$ because there $g_i(\boldsymbol{b}) = \boldsymbol{\lambda}_i^\top \boldsymbol{b}$ and $\boldsymbol{\lambda}_i^\top \boldsymbol{y} \leq 0$, and for $i \notin \mathcal{Z}$ once ε is small because $g_i(\boldsymbol{b}) > \boldsymbol{\lambda}_i^\top \boldsymbol{b}$. So $\boldsymbol{e}_1 \in F' \setminus F$, a contradiction. Stacking the vectors $\boldsymbol{\gamma}$ obtained for $p = 1, \ldots, m$ as rows gives $\boldsymbol{\Gamma}_{\boldsymbol{b}}$. □

Theorem 4.2 (Re-descriptions). *The constraint rewriting* $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{g})$ *is a re-description if and only if* $\boldsymbol{\Lambda} \geq 0$ *and, for every* $\boldsymbol{b} \in \mathbb{R}^m$,

- $\boldsymbol{g}(\boldsymbol{b}) \geq \boldsymbol{\Lambda}\boldsymbol{b}$, *and*
- *for all* $p \in [m]$, *there are* $i \in [m']$ *and* $\theta > 0$ *such that* $\boldsymbol{\lambda}_i^\top = \theta \boldsymbol{e}_p^\top$ *and* $g_i(\boldsymbol{b}) = \theta b_p$.

For affine $\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}$ *this means that up to row permutation,* $\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Pi} \\ \boldsymbol{N} \end{pmatrix}$ *and* $\boldsymbol{s} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{s}' \end{pmatrix}$ *where* $\boldsymbol{\Pi}$ *is a positive generalized permutation matrix,* $\boldsymbol{N} \geq 0$ *and* $\boldsymbol{s}' \geq 0$.

Proof. By Lemma A.4 it suffices to show that, for $\boldsymbol{\Lambda} \geq 0$ and $\boldsymbol{g}(\boldsymbol{b}) \geq \boldsymbol{\Lambda}\boldsymbol{b}$, a certificate $\boldsymbol{\Gamma}_{\boldsymbol{b}}$ exists if and only if every constraint has a tight positive copy.

If row i has $\boldsymbol{\lambda}_i^\top = \theta \boldsymbol{e}_p^\top$ with $\theta > 0$ and $g_i(\boldsymbol{b}) = \theta b_p$, put $1/\theta$ in position (p, i) of $\boldsymbol{\Gamma}_{\boldsymbol{b}}$. Doing this for every p and filling the rest with zeros gives $\boldsymbol{\Gamma}_{\boldsymbol{b}} \geq 0$ with $\boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{\Lambda} = \boldsymbol{I}$ and $\boldsymbol{\Gamma}_{\boldsymbol{b}}\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{b}$.

Conversely, row p of $\boldsymbol{\Gamma}_{\boldsymbol{b}}$ is a vector $\boldsymbol{\gamma} \geq 0$ with $\boldsymbol{e}_p^\top = \sum_i \gamma_i \boldsymbol{\lambda}_i^\top$ and $\boldsymbol{\gamma}^\top \boldsymbol{g}(\boldsymbol{b}) = b_p$. As in the proof of Lemma A.4, the latter says that $\boldsymbol{\gamma}$ is supported on the rows with $g_i(\boldsymbol{b}) = \boldsymbol{\lambda}_i^\top \boldsymbol{b}$. For every coordinate $l \neq p$, the sum $\sum_i \gamma_i \Lambda_{i,l} = 0$ has nonnegative terms, so $\gamma_i \Lambda_{i,l} = 0$ for all i. Coordinate p gives $\sum_i \gamma_i \Lambda_{i,p} = 1$, so some i has $\gamma_i \Lambda_{i,p} > 0$. For that i, $\gamma_i > 0$ forces $\Lambda_{i,l} = 0$ for all $l \neq p$, so $\boldsymbol{\lambda}_i^\top = \Lambda_{i,p} \boldsymbol{e}_p^\top$ with $\Lambda_{i,p} > 0$, and $g_i(\boldsymbol{b}) = \boldsymbol{\lambda}_i^\top \boldsymbol{b}$ because $\boldsymbol{\gamma}$ is supported on such rows.

For affine $\boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}$, the first condition is equivalent to $\boldsymbol{s} \geq 0$, and a *copy row* of p, that is, a row i with $\boldsymbol{\lambda}_i^\top = \theta \boldsymbol{e}_p^\top$ and $\theta > 0$, is tight if and only if $s_i = 0$, independently of $\boldsymbol{b}$. Choosing one tight copy for each p gives m distinct rows, which after permuting output rows form $\boldsymbol{D}\boldsymbol{\Pi}$ with $\boldsymbol{s} = \boldsymbol{0}$ on them. The remaining rows form $\boldsymbol{N} \geq 0$ with $\boldsymbol{s}' \geq 0$. The converse is immediate. □

A.4 Proof of Theorem 4.3

Theorem 4.3 (Completeness). *An affine rule* $\Phi : \mathcal{M}_m \to \mathcal{M}_{m'}$ *is valid if and only if* $\Phi = \mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}) \circ \mathsf{S}(\boldsymbol{T}, \boldsymbol{q})$ *with* $(\boldsymbol{T}, \boldsymbol{q})$ *as in Theorem 4.1 and* $(\boldsymbol{\Lambda}, \boldsymbol{s})$ *as in Theorem 4.2.*

Sufficiency. Let $(\boldsymbol{T}, \boldsymbol{q})$ and $(\boldsymbol{\Lambda}, \boldsymbol{s})$ satisfy the conditions of Theorems 4.1 and 4.2. The composition is affine. Next, observe that the pair $(\psi, -\boldsymbol{c}^\top \boldsymbol{q})$ with $\psi(\boldsymbol{x}) = \boldsymbol{T}^{-1}(\boldsymbol{x} - \boldsymbol{q})$ witnesses the equivalence of M and $\mathsf{S}(\boldsymbol{T}, \boldsymbol{q}) \cdot M$, and that the identity witnesses the equivalence of $\mathsf{S}(\boldsymbol{T}, \boldsymbol{q}) \cdot M$ and $\Phi(M)$. Composing the two, $(\psi, -\boldsymbol{c}^\top \boldsymbol{q})$ is a witness on every instance.

Necessity. Let Φ be affine and valid, and write $\Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) = (\boldsymbol{A}', \boldsymbol{b}', \boldsymbol{c}')$, $F = F(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ and $F' = F(\Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}))$. A composition of valid rules is valid, since witnesses compose. Since Φ is affine, there are linear maps $\boldsymbol{L}, \boldsymbol{L}_{\boldsymbol{b}}, \boldsymbol{L}_{\boldsymbol{c}}, \boldsymbol{h}, \boldsymbol{K}_{\boldsymbol{A}}$, matrices $\boldsymbol{G}, \boldsymbol{G}_{\boldsymbol{c}}, \boldsymbol{K}, \boldsymbol{K}_{\boldsymbol{b}}$ and constants $\boldsymbol{A}_0, \boldsymbol{s}, \boldsymbol{c}_0$ with

$$\begin{aligned} \boldsymbol{A}' = \boldsymbol{A}'(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) &= \boldsymbol{L}(\boldsymbol{A}) &&+ \boldsymbol{L}_{\boldsymbol{b}}(\boldsymbol{b}) + \boldsymbol{L}_{\boldsymbol{c}}(\boldsymbol{c}) + \boldsymbol{A}_0, \\ \boldsymbol{b}' = \boldsymbol{b}'(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) &= \boldsymbol{h}(\boldsymbol{A}) &&+ \boldsymbol{G}\boldsymbol{b} \quad + \boldsymbol{G}_{\boldsymbol{c}}\boldsymbol{c} \quad + \boldsymbol{s}, \\ \boldsymbol{c}' = \boldsymbol{c}'(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) &= \boldsymbol{K}_{\boldsymbol{A}}(\boldsymbol{A}) &&+ \boldsymbol{K}_{\boldsymbol{b}}\boldsymbol{b} \quad + \boldsymbol{K}\boldsymbol{c} \quad + \boldsymbol{c}_0. \end{aligned} \tag{15}$$

The proof eliminates or pins down these terms one at a time until only the form of Theorem 4.3 is left.

Lemma A.5. *In (15), $\boldsymbol{c}_0 = \mathbf{0}$, $\boldsymbol{K}_{\boldsymbol{A}} = 0$, $\boldsymbol{K}_{\boldsymbol{b}} = \mathbf{0}$ and $\boldsymbol{K} = \boldsymbol{T}^\top$ for a matrix $\boldsymbol{T}$ of the block form of Theorem 4.1. If moreover $\boldsymbol{c}' = \boldsymbol{c}$ on every instance, then $\boldsymbol{A}_0 = \mathbf{0}$, $\boldsymbol{L}_{\boldsymbol{b}} = 0$, $\boldsymbol{L}_{\boldsymbol{c}} = 0$, $\boldsymbol{G}_{\boldsymbol{c}} = \mathbf{0}$, $\boldsymbol{G} \geq 0$ and $\boldsymbol{s} \geq 0$.*

Proof. The instance $(\mathbf{0}, \mathbf{0}, \boldsymbol{c})$ has $F = \mathbb{V}$, so $F' = \mathbb{V}$ by Lemma A.3. A witness maps $\mathbb{V}$ onto $\mathbb{V}$, so its linear part $\boldsymbol{S}_{\boldsymbol{c}}$ has the block form by Lemma A.1, and $\boldsymbol{S}_{\boldsymbol{c}}^\top \boldsymbol{c}' = \boldsymbol{c}$ by Lemma A.2. On this instance $\boldsymbol{c}' = \boldsymbol{K}\boldsymbol{c} + \boldsymbol{c}_0$, so $\boldsymbol{S}_{\boldsymbol{c}}^\top(\boldsymbol{K}\boldsymbol{c} + \boldsymbol{c}_0) = \boldsymbol{c}$. At $\boldsymbol{c} = \mathbf{0}$ this gives $\boldsymbol{c}_0 = \mathbf{0}$, and then $\boldsymbol{S}_{\boldsymbol{c}}^\top \boldsymbol{K}\boldsymbol{c} = \boldsymbol{c}$ for all $\boldsymbol{c}$, so $\boldsymbol{K}$ is invertible. For $\boldsymbol{T} := \boldsymbol{K}^\top$ we can apply the argument from the necessity half of Section A.2 verbatim in order to prove that $\boldsymbol{T}$ has the block form of Theorem 4.1.

Next let $\boldsymbol{c} = \mathbf{0}$ and $(\boldsymbol{A}, \boldsymbol{b})$ in the open set

$$\Omega := \left\{ (\boldsymbol{A}, \boldsymbol{b}) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m \;\middle|\; \boldsymbol{b} > \mathbf{1},\ \max_{p,j} |A_{p,j}| < 1 \right\}$$

Then F contains $\mathbf{0}, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_n$ and affinely spans $\mathbb{R}^n$, so Lemma A.2 gives $\boldsymbol{c}' = \mathbf{0}$, that is, $\boldsymbol{K}_{\boldsymbol{A}}(\boldsymbol{A}) + \boldsymbol{K}_{\boldsymbol{b}}\boldsymbol{b} = \mathbf{0}$ on Ω. A linear map vanishing on an open set vanishes everywhere, so $\boldsymbol{K}_{\boldsymbol{A}} = 0$ and $\boldsymbol{K}_{\boldsymbol{b}} = \mathbf{0}$.

Now consider the case $\boldsymbol{c}' = \boldsymbol{c}$. For $\boldsymbol{b} \geq 0$ the instance $(\mathbf{0}, \boldsymbol{b}, \boldsymbol{c})$ has $F = \mathbb{V}$, so Lemma A.3 gives $\boldsymbol{A}'(\mathbf{0}, \boldsymbol{b}, \boldsymbol{c}) = \mathbf{0}$ and $\boldsymbol{b}'(\mathbf{0}, \boldsymbol{b}, \boldsymbol{c}) \geq 0$. The affine map $\boldsymbol{A}'$ vanishes on $\{\mathbf{0}\} \times \mathbb{R}^m_{\geq 0} \times \mathbb{R}^n$, hence on its affine span $\{\mathbf{0}\} \times \mathbb{R}^m \times \mathbb{R}^n$, that is, $\boldsymbol{L}_{\boldsymbol{b}}(\boldsymbol{b}) + \boldsymbol{L}_{\boldsymbol{c}}(\boldsymbol{c}) + \boldsymbol{A}_0 = \mathbf{0}$ for all $\boldsymbol{b}$ and $\boldsymbol{c}$, so $\boldsymbol{A}_0 = \mathbf{0}$, $\boldsymbol{L}_{\boldsymbol{b}} = 0$ and $\boldsymbol{L}_{\boldsymbol{c}} = 0$. Likewise $\boldsymbol{G}\boldsymbol{b} + \boldsymbol{G}_{\boldsymbol{c}}\boldsymbol{c} + \boldsymbol{s} \geq 0$ for all $\boldsymbol{b} \geq 0$ and all $\boldsymbol{c}$. At $\boldsymbol{b} = \mathbf{0}$, scaling $\boldsymbol{c}$ gives $\boldsymbol{G}_{\boldsymbol{c}} = \mathbf{0}$ and then $\boldsymbol{s} \geq 0$. Taking $\boldsymbol{b} = N\boldsymbol{e}_p$ with $N \to \infty$ gives $\boldsymbol{G}\boldsymbol{e}_p \geq 0$, so $\boldsymbol{G} \geq 0$. □

By Lemma A.1 and Theorem 4.1, $\mathsf{S}(\boldsymbol{T}^{-1}, \mathbf{0}) = \mathsf{S}(\boldsymbol{T}, \mathbf{0})^{-1}$ is valid. We replace Φ by $\Phi \circ \mathsf{S}(\boldsymbol{T}^{-1}, \mathbf{0})$, keeping the notation of (15) for the new rule. Its objective component is $\boldsymbol{K}\boldsymbol{T}^{-\top}\boldsymbol{c} = \boldsymbol{c}$, so from now on $\boldsymbol{c}' = \boldsymbol{c}$, and Lemma A.5 gives

$$\Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) = \big(\boldsymbol{L}(\boldsymbol{A}),\ \boldsymbol{h}(\boldsymbol{A}) + \boldsymbol{G}\boldsymbol{b} + \boldsymbol{s},\ \boldsymbol{c}\big), \qquad \boldsymbol{G} \geq 0,\ \boldsymbol{s} \geq 0. \tag{16}$$

In particular F' depends only on $(\boldsymbol{A}, \boldsymbol{b})$.

Lemma A.6. *There is $\boldsymbol{\Lambda} \geq 0$ with $\boldsymbol{L}(\boldsymbol{A}) = \boldsymbol{\Lambda}\boldsymbol{A}$ for all $\boldsymbol{A}$.*

Proof. Fix $p \in [m]$, $\boldsymbol{a} \neq \mathbf{0}$ and $\beta \in \mathbb{R}$, and let $\boldsymbol{A} = \boldsymbol{e}_p\boldsymbol{a}^\top$ and $\boldsymbol{b} = \beta\boldsymbol{e}_p$, so that $F = \{\boldsymbol{x} \in \mathbb{V} \mid \boldsymbol{a}^\top\boldsymbol{x} \leq \beta\}$. Then $\operatorname{conv} F$ contains the half-space $H = \{\boldsymbol{x} \mid \boldsymbol{a}^\top\boldsymbol{x} \leq \beta - \|\boldsymbol{a}\|_1\}$. Indeed, for any $\boldsymbol{x} \in H$, every vertex $\boldsymbol{z}$ of the unit cube $\lfloor \boldsymbol{x} \rfloor + [0,1]^n$ is an integer vector with $\boldsymbol{a}^\top\boldsymbol{z} \leq \boldsymbol{a}^\top\boldsymbol{x} + \|\boldsymbol{a}\|_1 \leq \beta$, so it lies in F. In particular F affinely spans $\mathbb{R}^n$. Let (φ, γ) be a witness of $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{a}) \sim \Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{a})$, whose objectives are both $\boldsymbol{a}$, and let $\boldsymbol{r}$ be the translation part of φ. By Lemma A.2, $\boldsymbol{a}^\top\varphi(\boldsymbol{x}) = \boldsymbol{a}^\top(\boldsymbol{x} + \boldsymbol{r})$ for all $\boldsymbol{x} \in \mathbb{R}^n$, and comparing with the value identity $\boldsymbol{a}^\top\varphi(\boldsymbol{x}) = \boldsymbol{a}^\top\boldsymbol{x} + \gamma$ on F gives $\gamma = \boldsymbol{a}^\top\boldsymbol{r}$. Hence $\varphi(H)$ is the half-space $H' := \{\boldsymbol{y} \mid \boldsymbol{a}^\top\boldsymbol{y} \leq \beta - \|\boldsymbol{a}\|_1 + \gamma\}$. As φ is affine, $\operatorname{conv} F' = \varphi(\operatorname{conv} F) \supseteq H'$.

Write $\boldsymbol{a}_i'^\top$ for row i of $\boldsymbol{L}(\boldsymbol{A})$ and b_i' for component i of $\boldsymbol{b}'$. The constraint $\boldsymbol{a}_i'^\top\boldsymbol{y} \leq b_i'$ of the image holds on F', hence on $\operatorname{conv} F'$, thus on H'. We claim that $\boldsymbol{a}_i' = \mu_i\boldsymbol{a}$ for some $\mu_i \geq 0$. Let $\boldsymbol{v} \in \mathbb{R}^n$ with $\boldsymbol{a}^\top\boldsymbol{v} \leq 0$. For $\boldsymbol{y} \in H'$ and $N \in \mathbb{N}$ the point $\boldsymbol{y} + N\boldsymbol{v}$ lies in H', so $\boldsymbol{a}_i'^\top\boldsymbol{y} + N\boldsymbol{a}_i'^\top\boldsymbol{v} \leq b_i'$ for all N, which forces $\boldsymbol{a}_i'^\top\boldsymbol{v} \leq 0$. Applying this to $\boldsymbol{v}$ and $-\boldsymbol{v}$ for every $\boldsymbol{v} \perp \boldsymbol{a}$ gives $\boldsymbol{a}_i' \perp \boldsymbol{a}^\perp$, so $\boldsymbol{a}_i' = \mu_i\boldsymbol{a}$ for some $\mu_i \in \mathbb{R}$, and applying it to $\boldsymbol{v} = -\boldsymbol{a}$ gives $\mu_i \geq 0$.

The map $\boldsymbol{a} \mapsto \boldsymbol{a}_i'$ is linear, since $\boldsymbol{L}$ is, and by the claim it sends every nonzero vector to a nonnegative multiple of itself. Such a map is $\Lambda_{i,p}\boldsymbol{I}$ for a scalar $\Lambda_{i,p} \geq 0$: if $n = 1$ every linear map of $\mathbb{R}$ is a scalar, which the claim makes nonnegative, and if $n \geq 2$, for linearly independent $\boldsymbol{a}, \boldsymbol{u}$ with multipliers μ, ν, the image $\mu\boldsymbol{a} + \nu\boldsymbol{u}$ of $\boldsymbol{a} + \boldsymbol{u}$ is a multiple of $\boldsymbol{a} + \boldsymbol{u}$ only if $\mu = \nu$. So row i of $\boldsymbol{L}(\boldsymbol{e}_p\boldsymbol{a}^\top)$ is $\Lambda_{i,p}\boldsymbol{a}^\top$ for every $\boldsymbol{a}$. As $\boldsymbol{A} = \sum_p \boldsymbol{e}_p\boldsymbol{a}_p^\top$, row i of $\boldsymbol{L}(\boldsymbol{A})$ is $\sum_p \Lambda_{i,p}\boldsymbol{a}_p^\top = \boldsymbol{\lambda}_i^\top\boldsymbol{A}$. □

By Lemma A.6, (16) becomes

$$\Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) = \big(\boldsymbol{\Lambda}\boldsymbol{A},\ \boldsymbol{h}(\boldsymbol{A}) + \boldsymbol{G}\boldsymbol{b} + \boldsymbol{s},\ \boldsymbol{c}\big), \qquad \boldsymbol{\Lambda} \geq 0. \tag{17}$$

Hyperplane instances. Let $\mathcal{T} \subseteq \mathbb{R}^m$ be the set of vectors with no zero entry and entries of both signs. It is open since its complement is the union of closed sets, and nonempty since $m \geq 2$. Fix $\boldsymbol{t} \in \mathcal{T}$ and consider, for $\xi \in \mathbb{R}$ and $\boldsymbol{a} \in \mathbb{R}^n$, the instance $(\boldsymbol{t}\,\boldsymbol{a}^\top,\ \xi\boldsymbol{t})$. Its row p reads $t_p \boldsymbol{a}^\top \boldsymbol{x} \leq \xi t_p$, and both signs of t_p occur, so $F = \{\boldsymbol{x} \in \mathbb{V} \mid \boldsymbol{a}^\top \boldsymbol{x} = \xi\}$ is a hyperplane intersected with $\mathbb{V}$. By (17), the image has constraint matrix $\boldsymbol{\Lambda} \boldsymbol{t} \boldsymbol{a}^\top$ and right-hand side $\boldsymbol{h}(\boldsymbol{t}\boldsymbol{a}^\top) + \xi \boldsymbol{G}\boldsymbol{t} + \boldsymbol{s}$, so its row i reads $\ell_i\, \boldsymbol{a}^\top \boldsymbol{y} \leq \rho_i(\xi, \boldsymbol{a})$ with

$$\ell_i := \boldsymbol{\lambda}_i^\top \boldsymbol{t}, \qquad \rho_i(\xi, \boldsymbol{a}) := \xi\, \boldsymbol{g}_i^\top \boldsymbol{t} + h_i(\boldsymbol{t}\,\boldsymbol{a}^\top) + s_i,$$

where $\boldsymbol{g}_i^\top$ is the i-th row of $\boldsymbol{G}$ and h_i the i-th entry of $\boldsymbol{h}$. Each ρ_i is an affine function of $(\xi, \boldsymbol{a})$, and $F' = \{\boldsymbol{y} \in \mathbb{V} \mid \ell_i\, \boldsymbol{a}^\top \boldsymbol{y} \leq \rho_i(\xi, \boldsymbol{a})\ \forall i\}$. Let $\Gamma_{\boldsymbol{a}} := \boldsymbol{a}^\top \mathbb{V}$, which contains $a_j \mathbb{Z}$ for every j.

Lemma A.7. *Let $\boldsymbol{t} \in \mathcal{T}$. There are $\kappa(\boldsymbol{t}) \in \mathbb{R}$ and $\boldsymbol{q}(\boldsymbol{t}) \in \mathbb{R}^n$ such that*

$$\boldsymbol{G}\boldsymbol{t} = \kappa(\boldsymbol{t})\, \boldsymbol{\Lambda}\boldsymbol{t} \qquad \text{and} \qquad \boldsymbol{h}(\boldsymbol{t}\,\boldsymbol{a}^\top) = -\boldsymbol{\Lambda}\boldsymbol{t}\ \boldsymbol{q}(\boldsymbol{t})^\top \boldsymbol{a} \quad \text{for all } \boldsymbol{a} \in \mathbb{R}^n.$$

Moreover, some row i has $\ell_i > 0$ and $s_i = 0$, and some row has $\ell_i < 0$.

Proof. Let $\boldsymbol{a} \neq \boldsymbol{0}$ and $\xi \in \Gamma_{\boldsymbol{a}}$, so that $F \neq \emptyset$ and $\boldsymbol{a}^\top F = \{\xi\}$. A witness (φ, γ) of $(\boldsymbol{t}\boldsymbol{a}^\top, \xi\boldsymbol{t}, \boldsymbol{a}) \sim \Phi(\boldsymbol{t}\boldsymbol{a}^\top, \xi\boldsymbol{t}, \boldsymbol{a})$ satisfies $\boldsymbol{a}^\top \varphi(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{x} + \gamma$ on F and maps F onto F', so $\boldsymbol{a}^\top F' = \{\xi + \gamma\}$ is a single point. Suppose $\ell_i \leq 0$ for all i. Pick $\boldsymbol{y} \in F'$, an index j with $a_j \neq 0$, and $\boldsymbol{v} := \operatorname{sign}(a_j) \cdot \boldsymbol{e}_j$ so that $\boldsymbol{a}^\top \boldsymbol{v} = |a_j|$ is satisfied. Then $\boldsymbol{y} + N\boldsymbol{v} \in F'$ for every $N \in \mathbb{N}$, and $\boldsymbol{a}^\top(\boldsymbol{y} + N\boldsymbol{v}) = \boldsymbol{a}^\top \boldsymbol{y} + N|a_j|$ takes infinitely many values, a contradiction. So some $\ell_i > 0$, and choosing $\boldsymbol{v} = -\operatorname{sign}(a_j) \cdot \boldsymbol{e}_j$ instead shows that some $\ell_i < 0$. Hence

$$H_+(\xi, \boldsymbol{a}) := \min_{\ell_i > 0} \frac{\rho_i(\xi, \boldsymbol{a})}{\ell_i}, \qquad H_-(\xi, \boldsymbol{a}) := \max_{\ell_i < 0} \frac{\rho_i(\xi, \boldsymbol{a})}{\ell_i}$$

are finite, H_+ is concave, H_- is convex, and $F' = \{\boldsymbol{y} \in \mathbb{V} \mid H_- \leq \boldsymbol{a}^\top \boldsymbol{y} \leq H_+\}$ provided $\rho_i \geq 0$ for the rows with $\ell_i = 0$, and $F' = \emptyset$ otherwise. Since $F' \neq \emptyset$, $\rho_i \geq 0$ for $\ell_i = 0$ and $\Delta := H_+ - H_- \geq 0$. Since $\boldsymbol{a}^\top F' = \Gamma_{\boldsymbol{a}} \cap [H_-, H_+]$ is a single point and $\Gamma_{\boldsymbol{a}} \supseteq a_j \mathbb{Z}$, we also have $\Delta < 2|a_j|$ for every j with $a_j \neq 0$.

These inequalities hold on $\mathcal{S} := \{(\xi, \boldsymbol{a}) \mid \boldsymbol{a} \neq \boldsymbol{0},\ \xi \in \Gamma_{\boldsymbol{a}}\}$. In order to extend them to $\mathbb{R} \times \mathbb{R}^n$, notice that every $(\xi, \boldsymbol{a})$ is the midpoint of $(2\xi, \boldsymbol{a}_+)$ and $(0, \boldsymbol{a}_-)$, where $\boldsymbol{a}_\pm := \boldsymbol{a} \pm (2\xi - a_1)\boldsymbol{e}_1$, and both points lie in $\mathcal{S}$ whenever $\boldsymbol{a}_+, \boldsymbol{a}_- \neq \boldsymbol{0}$, because $\boldsymbol{a}_+^\top \boldsymbol{e}_1 = 2\xi$ by construction and $0 \in \Gamma_{\boldsymbol{a}_-}$. By concavity of Δ and affineness of ρ_i, $\Delta \geq 0$ and $\rho_i \geq 0$ (for $\ell_i = 0$) hold at all such points. The excluded points, where $\boldsymbol{a}_+ = \boldsymbol{0}$ or $\boldsymbol{a}_- = \boldsymbol{0}$, have $\boldsymbol{a} = \alpha \boldsymbol{e}_1$ and $\xi \in \{0, \alpha\}$ for some $\alpha \in \mathbb{R}$, so they lie on two lines, and the inequalities hold everywhere by continuity. A concave function bounded below on $\mathbb{R}^{1+n}$ is constant, and so is an affine one. Evaluating Δ at $(0, \varepsilon \boldsymbol{e}_1) \in \mathcal{S}$ gives $\Delta < 2\varepsilon$ for every $\varepsilon > 0$, so $\Delta \equiv 0$, and evaluating ρ_i at $(0, \boldsymbol{0})$ gives $\rho_i \equiv s_i$ for $\ell_i = 0$.

So $H := H_+ = H_-$ is concave and convex, hence affine. At $(0, \boldsymbol{0})$, where $\rho_i = s_i$, it equals $\min_{\ell_i > 0} s_i/\ell_i \geq 0$ and $\max_{\ell_i < 0} s_i/\ell_i \leq 0$. Hence $H(0, \boldsymbol{0}) = 0$, the row attaining the minimum has $\ell_i > 0$ and $s_i = 0$, and H is linear, say $H(\xi, \boldsymbol{a}) = \kappa(\boldsymbol{t})\xi - \boldsymbol{q}(\boldsymbol{t})^\top \boldsymbol{a}$. For $\ell_i > 0$ the affine function $\rho_i/\ell_i - H$ is nonnegative, and for $\ell_i < 0$ it is nonpositive. In both cases it is constant, equal to its value s_i/ℓ_i at $(0, \boldsymbol{0})$. So $\rho_i - s_i = \ell_i H$ for every row, including those with $\ell_i = 0$, and comparing the coefficients of ξ and of $\boldsymbol{a}$ gives $\boldsymbol{g}_i^\top \boldsymbol{t} = \kappa(\boldsymbol{t})\ell_i$ and $h_i(\boldsymbol{t}\boldsymbol{a}^\top) = -\ell_i\, \boldsymbol{q}(\boldsymbol{t})^\top \boldsymbol{a}$, which is the claim row by row. □

Lemma A.8. *Every $p \in [m]$ has a copy row, that is, a row with $\boldsymbol{\lambda}_i^\top = \theta_i \boldsymbol{e}_p^\top$ and $\theta_i > 0$, with $s_i = 0$. There are $\kappa \geq 0$ and $\boldsymbol{q} \in \mathbb{R}^n$ with $\boldsymbol{G} = \kappa \boldsymbol{\Lambda}$ and $\boldsymbol{h}(\boldsymbol{A}) = -\boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{q}$ for all $\boldsymbol{A}$.*

Proof. Fix p and let $\boldsymbol{t} = \tau \boldsymbol{e}_p - \sum_{r \neq p} \boldsymbol{e}_r \in \mathcal{T}$ with $\tau > 0$, so that $\ell_i = \tau \Lambda_{i,p} - \sum_{r \neq p} \Lambda_{i,r}$. As $\boldsymbol{\Lambda} \geq 0$, if τ is smaller than every positive number among the ratios $\sum_{r \neq p} \Lambda_{i,r}/\Lambda_{i,p}$ over the rows with $\Lambda_{i,p} > 0$, then the rows with $\ell_i > 0$ are exactly the copy rows of p. By Lemma A.7 such a row exists with $s_i = 0$.

Let i and i' be copy rows of $p \neq p'$. By Lemma A.7, $\kappa(\boldsymbol{t}) = \boldsymbol{g}_i^\top \boldsymbol{t}/(\theta_i t_p) = \boldsymbol{g}_{i'}^\top \boldsymbol{t}/(\theta_{i'} t_{p'})$ for $\boldsymbol{t} \in \mathcal{T}$, so $\theta_{i'} t_{p'}\, \boldsymbol{g}_i^\top \boldsymbol{t} = \theta_i t_p\, \boldsymbol{g}_{i'}^\top \boldsymbol{t}$ on the open set $\mathcal{T}$, hence as polynomials. As t_p divides the right-hand side and not $t_{p'}$, it divides the linear form $\boldsymbol{g}_i^\top \boldsymbol{t}$, so $\boldsymbol{g}_i^\top \boldsymbol{t} = \mu t_p$ for some $\mu \in \mathbb{R}$, and $\kappa(\boldsymbol{t}) = \mu/\theta_i$ is a constant κ. The same argument applied to $-\boldsymbol{q}(\boldsymbol{t})^\top \boldsymbol{a} = h_i(\boldsymbol{t}\boldsymbol{a}^\top)/(\theta_i t_p)$, for fixed $\boldsymbol{a}$, shows that

$\boldsymbol{q}(\boldsymbol{t}) =: \boldsymbol{q}$ is constant. Now Lemma A.7 gives $\boldsymbol{g}_i^\top \boldsymbol{t} = \kappa\, \boldsymbol{\lambda}_i^\top \boldsymbol{t}$ and $h_i(\boldsymbol{t}\boldsymbol{a}^\top) = -\boldsymbol{\lambda}_i^\top (\boldsymbol{t}\boldsymbol{a}^\top)\boldsymbol{q}$ for all rows, all $\boldsymbol{t}$ in the open set $\mathcal{T}$ and all $\boldsymbol{a}$. Both sides are linear in $\boldsymbol{t}$, so the identities hold for all $\boldsymbol{t} \in \mathbb{R}^m$. The first gives $\boldsymbol{G} = \kappa\boldsymbol{\Lambda}$, and the second gives $\boldsymbol{h}(\boldsymbol{A}) = -\boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{q}$, since the matrices $\boldsymbol{t}\boldsymbol{a}^\top$ span $\mathbb{R}^{m\times n}$. Finally a copy row has $\boldsymbol{g}_i = \kappa\theta_i \boldsymbol{e}_p \geq 0$, so $\kappa \geq 0$. □

By Lemma A.8, (17) becomes

$$\Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) = \big(\boldsymbol{\Lambda}\boldsymbol{A},\ \boldsymbol{\Lambda}(\kappa\boldsymbol{b} - \boldsymbol{A}\boldsymbol{q}) + \boldsymbol{s},\ \boldsymbol{c}\big), \qquad \kappa \geq 0. \tag{18}$$

Lemma A.9. $\kappa = 1$.

Proof. Fix $\boldsymbol{t} \in \mathcal{T}$ and $\boldsymbol{b} \in \mathbb{R}^m$ with $U > L$, where $U := \min_{t_p>0} b_p/t_p$ and $L := \max_{t_p<0} b_p/t_p$, for instance $b_p = \max(t_p, 0)$. For $N \in \mathbb{N}$ the instance $(\boldsymbol{t}\,\boldsymbol{e}_1^\top,\ N\boldsymbol{b})$ has feasible set $\{\boldsymbol{x} \in \mathbb{V} \mid NL \leq x_1 \leq NU\}$. By (18), row i of its image reads $\ell_i y_1 \leq N\kappa\, \boldsymbol{\lambda}_i^\top \boldsymbol{b} - \ell_i q_1 + s_i$, so the image has feasible set $\{\boldsymbol{y} \in \mathbb{V} \mid L'_N \leq y_1 \leq U'_N\}$ with

$$U'_N = \min_{\ell_i>0} \frac{N\kappa\, \boldsymbol{\lambda}_i^\top \boldsymbol{b} + s_i}{\ell_i} - q_1, \qquad L'_N = \max_{\ell_i<0} \frac{N\kappa\, \boldsymbol{\lambda}_i^\top \boldsymbol{b} + s_i}{\ell_i} - q_1,$$

provided the rows with $\ell_i = 0$, which read $0 \leq N\kappa\, \boldsymbol{\lambda}_i^\top \boldsymbol{b} + s_i$, hold.

For every row i, $\boldsymbol{\lambda}_i^\top \boldsymbol{b} - \ell_i U = \sum_p \Lambda_{i,p} t_p (b_p/t_p - U)$ is a sum of nonnegative terms, because $b_p/t_p \geq U$ when $t_p > 0$ and $b_p/t_p \leq L < U$ when $t_p < 0$, and it vanishes for a copy row of a constraint p with $b_p/t_p = U$. Likewise $\boldsymbol{\lambda}_i^\top \boldsymbol{b} - \ell_i L \geq 0$, with equality for a copy row of a constraint p with $b_p/t_p = L$. In particular, rows with $\ell_i = 0$ have $\boldsymbol{\lambda}_i^\top \boldsymbol{b} \geq 0$, so their constraints hold. Let $C := \max_{\ell_i \neq 0} |s_i/\ell_i|$. For $\ell_i > 0$,

$$\frac{N\kappa\, \boldsymbol{\lambda}_i^\top \boldsymbol{b} + s_i}{\ell_i} = N\kappa U + N\kappa\, \frac{\boldsymbol{\lambda}_i^\top \boldsymbol{b} - \ell_i U}{\ell_i} + \frac{s_i}{\ell_i} \ \geq\ N\kappa U,$$

with the middle term vanishing for the copy row of a constraint p with $b_p/t_p = U$, so $N\kappa U - q_1 \leq U'_N \leq N\kappa U - q_1 + C$. Similarly $N\kappa L - q_1 - C \leq L'_N \leq N\kappa L - q_1$, and together $|(U'_N - L'_N) - N\kappa(U - L)| \leq 2C$.

Now take N with $N(U - L) \geq 2$, so that the feasible set is nonempty, and let $\Gamma := \boldsymbol{e}_1^\top \mathbb{V} \in \{\mathbb{R}, \mathbb{Z}\}$. A witness (φ, γ) of $(\boldsymbol{t}\boldsymbol{e}_1^\top, N\boldsymbol{b}, \boldsymbol{e}_1) \sim \Phi(\boldsymbol{t}\boldsymbol{e}_1^\top, N\boldsymbol{b}, \boldsymbol{e}_1)$ satisfies $\boldsymbol{e}_1^\top \varphi(\boldsymbol{x}) = x_1 + \gamma$ on F and maps F onto F', so the set $\boldsymbol{e}_1^\top F' = \Gamma \cap [L'_N, U'_N]$ is the translate of $\boldsymbol{e}_1^\top F = \Gamma \cap [NL, NU]$ by γ, and their diameters agree. The diameter of a nonempty set $\Gamma \cap [L, U]$ lies in $(U - L - 2,\ U - L]$, so $|(U'_N - L'_N) - N(U - L)| < 2$. With the bound above, $|N\kappa(U - L) - N(U - L)| < 2 + 2C$, and dividing by N and letting $N \to \infty$ gives $\kappa = 1$. □

By Lemma A.9, (18) reads $\Phi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) = \big(\boldsymbol{\Lambda}\boldsymbol{A},\ \boldsymbol{\Lambda}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{q}) + \boldsymbol{s},\ \boldsymbol{c}\big) = \mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}) \circ \mathsf{S}(\boldsymbol{I}, \boldsymbol{q}) \cdot (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$, since $\mathsf{S}(\boldsymbol{I}, \boldsymbol{q}) \cdot (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) = (\boldsymbol{A}, \boldsymbol{b} - \boldsymbol{A}\boldsymbol{q}, \boldsymbol{c})$. Here $\boldsymbol{\Lambda} \geq 0$, $\boldsymbol{s} \geq 0$, and by Lemma A.8 every p has a copy row i with $s_i = 0$, that is, with $g_i(\boldsymbol{b}) = \theta_i b_p$. So $(\boldsymbol{\Lambda}, \boldsymbol{s})$ satisfies the conditions of Theorem 4.2, and by the sufficiency half of Section A.3 the re-description preserves feasible sets and objectives. Hence $\mathsf{S}(\boldsymbol{I}, \boldsymbol{q}) \cdot M$ has the same feasible set and objective as $\Phi(M)$ for every M, so a witness of $M \sim \Phi(M)$ is one of $M \sim \mathsf{S}(\boldsymbol{I}, \boldsymbol{q}) \cdot M$. Thus $\mathsf{S}(\boldsymbol{I}, \boldsymbol{q})$ is valid and $\boldsymbol{q} \in \mathbb{V}$ by Theorem 4.1. Undoing the normalization, $\Phi = \mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}) \circ \mathsf{S}(\boldsymbol{I}, \boldsymbol{q}) \circ \mathsf{S}(\boldsymbol{T}, \boldsymbol{0}) = \mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s}) \circ \mathsf{S}(\boldsymbol{T}, \boldsymbol{T}\boldsymbol{q})$, as $\mathsf{S}(\boldsymbol{I}, \boldsymbol{q}) \circ \mathsf{S}(\boldsymbol{T}, \boldsymbol{0})$ maps $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ to $(\boldsymbol{A}\boldsymbol{T}, \boldsymbol{b} - \boldsymbol{A}\boldsymbol{T}\boldsymbol{q}, \boldsymbol{T}^\top \boldsymbol{c})$, and $\boldsymbol{T}\boldsymbol{q} \in \mathbb{V}$ because $\boldsymbol{T}$ has the block form. □

In particular, if a constraint rewriting $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}\boldsymbol{b} + \boldsymbol{s})$ with affine $\boldsymbol{g}$ is valid, then Theorem 4.3 writes it as $\mathsf{R}(\boldsymbol{\Lambda}', \boldsymbol{\Lambda}'\boldsymbol{b} + \boldsymbol{s}') \circ \mathsf{S}(\boldsymbol{T}, \boldsymbol{q})$ with $(\boldsymbol{\Lambda}', \boldsymbol{s}')$ as in Theorem 4.2. Comparing the two rules on all instances gives $\boldsymbol{T} = \boldsymbol{I}$, $\boldsymbol{\Lambda}' = \boldsymbol{\Lambda}$, $\boldsymbol{s}' = \boldsymbol{s}$ and $\boldsymbol{q} = \boldsymbol{0}$, so the rewriting is a re-description. This is the converse claimed after (5).

B Contribution of the substitutions

The two views of ReMILP differ by independent re-descriptions and by a substitution applied to the second one. To isolate the impact of the substitutions, as well as the equivariance machinery that we

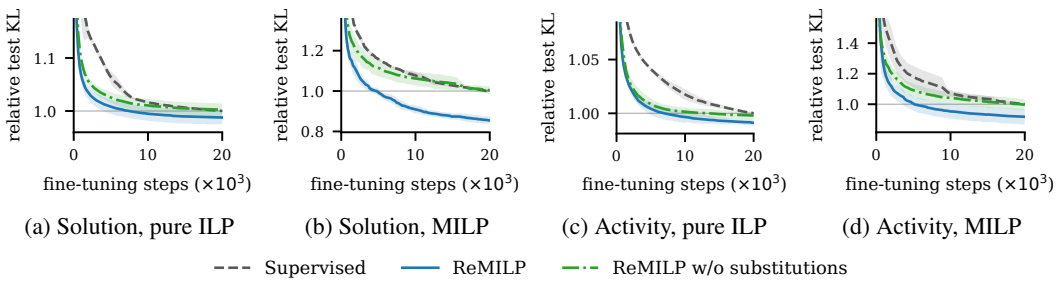


Figure 4: Fine-tuning ReMILP with and without substitutions on the node-level tasks, 20 000 steps. Each panel shows the geometric mean over the pairs of a group of the test KL at the checkpoint selected on validation, divided by the value the supervised model of the same seed reaches at the end of its budget. Shaded bands are one standard deviation over three seeds, and the horizontal line marks parity with the best supervised model.

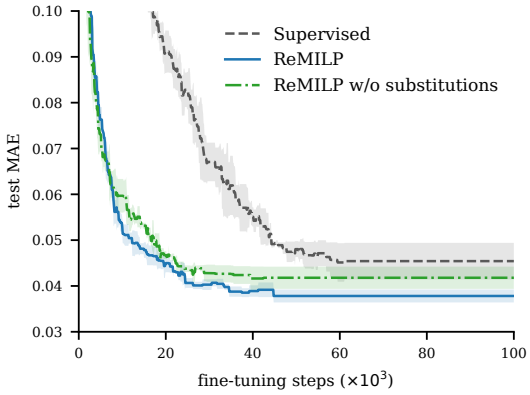


Figure 5: Fine-tuning ReMILP with and without substitutions on the integrality gap: test MAE, mean and one standard deviation over three seeds, 100 000-step budget on the full training set.

introduced to make use of them, we pre-train the same encoder without them in a purely contrastive way, so that the views differ by redundant constraints only. Each variable's embedding must then match its own embedding in the other view, and the negatives are embeddings of other variables. Node-level heads are trained for 20 000 steps and the integrality gap head for 100 000 steps. The results are reported in Figures 4 and 5.

Without substitutions the pre-trained encoder still converges faster than the supervised model, but it converges to the same value on every group, contrary to the original ReMILP which converges to 0.854 ± 0.020 of the supervised model's KL on solution prediction and 0.918 ± 0.050 on constraint activity. Redundant constraints alone thus account for the speed-up of Section 5.2.3, but the lasting advantage on mixed-integer families comes from the substitutions.

The integrality gap shows the same split between speed and final error. Without substitutions the encoder descends as fast as ReMILP and far ahead of the supervised model, but it settles at 0.042 ± 0.002, above ReMILP's 0.038 ± 0.001 and within one standard deviation of the supervised model's 0.045 ± 0.004.

C Datasets

In this work, we make use of two datasets for training and evaluating our models:

C.1 MILP-EVOLVE

We made use of the integrality gap prediction dataset of MILP-Evolve (Li et al., 2025) for both pre-training and integrality gap prediction. MILP instance families were filtered by feasibility, objective boundedness, and primal gap convergence. Instance families with SOS or indicator constraints were eliminated. Instance families that allowed instances with zero variables or zero constraints were also filtered out. After the filtering step, only 42k instances remained, which were randomly split into a pre-training set (70%) and an evaluation set (30%). The pre-training set was further split into a 90% training subset, and a 10% subset for validation. Similarly, the evaluation set was split into training (70%), validation (10%) and test (20%) subsets, which were all used for integrality gap prediction. We additionally recomputed the integrality gap targets using the well-conditioned formula

$$\frac{|\mathrm{LP} - \mathrm{IP}|}{\max\{|\mathrm{LP}|, |\mathrm{IP}|, 10^{-5}\}}$$

rather than $\frac{|\mathrm{LP}-\mathrm{IP}|}{|\mathrm{LP}|}$ which was used by MILP-Evolve as the latter led to exploding gradients in our setup.

C.2 DISTRIBUTIONAL MIPLIB

Instance family selection We use all instance families with a difficulty of Easy, Medium or Hard and containing at least 300 instances from the Distributional MIPLIB dataset (Huang et al., 2024b). This leaves us with 23 instance families across 10 problem classes, summarized in Table 4.

Table 4: Instance families used from Distributional MIPLIB.

Problem class	Abbreviation	Easy	Medium	Hard
Combinatorial Auctions	CA	✓	✓	
Maximum Independent Set	IS	✓	✓	
Set Covering	SC	✓	✓	✓
Minimum Vertex Cover	VC	✓	✓	✓
Generalized Independent Set Problem	GISP	✓	✓	✓
Capacitated Facility Location Problem	CFLP	✓	✓	
Load Balancing	LB			✓
Neural Network Verification	NNV	✓		
Optimal Transmission Switching	OTS	✓	✓	✓
Middle-Mile Consolidation Network	MMCN		BI, BC	BI

For each family, we separate our instances into 200 for training, 50 for validation and 50 for testing, which are solved in the manner described below. CFLP, LB, NNV, OTS and MMCN medium-BC contain both integer and continuous variables. The other families are pure ILPs.

Node-level targets In order to compute the binary solution and constraint activity target values, we solve each family's 300 instances using the Gurobi solver (Gurobi Optimization, LLC, 2026). For each instance, we use a 1-hour time limit and retain the best 300 solutions during that time. Let $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(S)}$ be the solution pool of an instance and $z_s = \boldsymbol{c}^\top \boldsymbol{x}^{(s)}$ their objective values. Following Nair et al. (2020), we weight the pool by objective quality:

$$w_s \propto \exp(-z_s),$$

and define both our variable-level and constraint-level targets as marginals of this distribution:

$$\hat{y}_j = \mathbb{E}_w[x_j]$$

for binary solution prediction (the weighted frequency with which a binary variable j takes value 1), and

$$\hat{d}_i = \mathbb{P}_w\left[\,|\boldsymbol{a}_i^\top \boldsymbol{x} - b_i| \le \varepsilon\,\right]$$

for constraint activity prediction (the weighted frequency with which constraint i is binding, up to a small tolerance $\varepsilon = 10^{-6}$).

D IMPLEMENTATION DETAILS

D.1 PREPROCESSING

Before encoding, we normalize each constraint row and its bound by the same positive factor, such that every single coefficient of $\boldsymbol{A}$ and component of $\boldsymbol{b}$ is bounded by 1. This normalization is principled insofar as it preserves a MILP's feasible region. Objective coefficients are scaled in the same way, which preserves optimal solutions and integrality gap. Our implementation of the various augmentations described in this paper ensures that instances stay row-normalized, which prevents distribution shift.

D.2 TRAINING

Across pre-training and downstream tasks, all optimization is done with Adam (Kingma & Ba, 2017). We use dynamic batches with a maximum size of 200 000 nodes and 1 200 000 edges. Any instance exceeding either of the two limits is excluded from the training. All pre-training is done for 20 000 steps at a learning rate of 10^{-3}. For all downstream tasks, we use a fixed learning rate of 10^{-4}. Binary solution and constraint activity prediction are run for 50 000 gradient steps, and integrality gap prediction is run for 100 000 gradient steps. All three tasks have early stopping if the best validation loss does not improve for 5 000 consecutive steps.

D.3 INSTANCE POOLING

The integrality gap is predicted from a single embedding of the instance, obtained by pooling the node embeddings. Variables and constraints are pooled separately, with a coordinate-wise attention: every output coordinate has its own softmax over the nodes. For the variables, a two-layer MLP s maps each embedding $\boldsymbol{h}_i(M) \in \mathbb{R}^d$ to a score vector $s(\boldsymbol{h}_i(M)) \in \mathbb{R}^p$, and a linear map $\boldsymbol{W}$ maps it to a value vector $\boldsymbol{W}\boldsymbol{h}_i(M) \in \mathbb{R}^p$. The pooled variable vector is

$$\boldsymbol{z}_V(M) = \sum_{i=1}^{n} \boldsymbol{a}_i \odot \boldsymbol{W}\boldsymbol{h}_i(M), \qquad a_{i,k} = \frac{\exp\big(s_k(\boldsymbol{h}_i(M))\big)}{\sum_{i'=1}^{n} \exp\big(s_k(\boldsymbol{h}_{i'}(M))\big)},$$

so each coordinate is a convex combination of the corresponding value coordinates of the variables, weighted by that coordinate's own attention distribution. The constraint embeddings $\boldsymbol{g}_j(M)$ are pooled in the same way by separate parameters into $\boldsymbol{z}_C(M)$. The instance embedding is the output of a two-layer MLP applied to $[\boldsymbol{z}_V(M); \boldsymbol{z}_C(M)]$. We use $d = 128$, $p = 256$ for each node type, and an instance embedding of dimension 256. The pooling and the MLP are trained together with the gap head, including when the encoder is frozen. FORGE† instead represents an instance by the histogram of its nodes' codebook assignments, and the variant marked "+ attn." replaces that histogram with the pooling described here.

D.4 HYPERNETWORK

The hypernetwork ϱ_ϕ of (11) is a single affine layer. For a transformed variable $i \in \mathcal{I}$, it reads the descriptor

$$t_i = \big(\log|u_i|,\ \mathbf{1}[u_i < 0],\ \lambda_i,\ \mu_i,\ \mathrm{sg}(\boldsymbol{h}_{j(i)}(M))\big) \in \mathbb{R}^{4+d},$$

in which the scaling enters through its log-magnitude and its sign so that complements and scalings of either sign are represented on a common scale. The partner embedding is replaced by $\mathbf{0}$ when x_i has no partner, that is, for binary variables and for variables that are only scaled. The layer outputs the d^2 entries of the operator,

$$\varrho_\phi(t_i) = \mathrm{mat}\big(\boldsymbol{W}_\phi t_i + \boldsymbol{b}_\phi\big) \in \mathbb{R}^{d\times d}, \qquad \boldsymbol{W}_\phi \in \mathbb{R}^{d^2\times(4+d)},\ \boldsymbol{b}_\phi \in \mathbb{R}^{d^2},$$

where mat reshapes a vector of length d^2 into a $d \times d$ matrix. The weights $\boldsymbol{W}_\phi$ are initialised from $\mathcal{N}(0, 0.02^2)$ and the bias to $\boldsymbol{b}_\phi = \mathrm{vec}(\boldsymbol{I}_d)$, so that every operator starts close to the identity and training begins from the invariant prediction $\boldsymbol{h}_i(M') \approx \boldsymbol{h}_i(M)$. With $d = 128$ the hypernetwork has about 2.2 million parameters. Among the action negatives ($1 \le k \le K$ in (13)), those drawn from a variable with the same descriptor parameters and the same partner as i would produce the positive itself, and they are removed from the softmax. The hypernetwork is used only during pre-training, and downstream heads read the encoder's embeddings directly.

D.5 SAMPLING AUGMENTATIONS

Both views of (10) are sampled afresh for every instance at every step. Algorithm 1 draws the substitution $\mathsf{S}(\boldsymbol{T}', \boldsymbol{q}')$ of the second view, and Algorithm 2 draws each of the two re-descriptions. The fraction $|\mathcal{I}|/n$, the bounds b_u and b_λ, the fraction $m'/m - 1$ and the number r of constraints combined per redundant row are the hyperparameters of Table 5.

Every transformed variable $i \in \mathcal{I}$ receives the local substitution (9). A binary variable must remain binary, and the only substitution with this property other than the identity is the complement $x_i \leftarrow 1 - x_i$, that is $u_i = -1$, $\lambda_i = 0$ and $\mu_i = 1$, which involves no partner. A transformed binary variable is therefore complemented with probability $\frac{1}{2}$ and otherwise left unchanged, and it stays in $\mathcal{I}$ in both cases, with the identity as its substitution in the second case. Every other transformed variable takes a partner $j(i)$ drawn from $\mathcal{J}$, so no partner is itself transformed and $\boldsymbol{T}'$ has the structure required by Theorem 4.1. An integer variable takes an integer partner, which may be binary, while a continuous variable may take a partner of any type. Binary variables in $\mathcal{J}$ are therefore eligible partners of both integer and continuous variables. We write $\mathbb{Z}_b \coloneqq \{z \in \mathbb{Z} : 1 \le |z| \le \lfloor b \rfloor\}$ for the nonzero integers of absolute value at most b.

Algorithm 1 Sampling the substitution $\mathsf{S}(\boldsymbol{T}', \boldsymbol{q}')$

Require: instance M_0 over $\mathbb{V}$, fraction $\kappa = |\mathcal{I}|/n$, bounds b_u, b_λ
$\mathcal{C} \leftarrow$ variables with a nonzero coefficient in some constraint of M_0
draw $\mathcal{I} \subseteq \mathcal{C}$ uniformly without replacement, with $|\mathcal{I}| = \mathrm{round}(\kappa|\mathcal{C}|)$
$\mathcal{J} \leftarrow [n] \setminus \mathcal{I}$
for $i \in \mathcal{I}$ **do**
  if x_i is binary **then**
    $(u_i, \lambda_i, \mu_i) \leftarrow (-1, 0, 1)$ with probability $\frac{1}{2}$, else $(1, 0, 0)$
  else if x_i is integer **then**
    draw $j(i)$ uniformly among the integer variables of $\mathcal{J}$
    draw u_i uniformly from $\{-1, 1\}$, and λ_i, μ_i uniformly from $\mathbb{Z}_{b_\lambda}$
  else
    draw $j(i)$ uniformly from $\mathcal{J}$
    draw $\log|u_i|$ uniformly from $[-\log b_u, \log b_u]$ and the sign of u_i uniformly
    draw λ_i, μ_i uniformly from $[-b_\lambda, b_\lambda]$
  end if
end for
return $\boldsymbol{T}', \boldsymbol{q}'$ assembled from $(u_i, \lambda_i, \mu_i, j(i))_{i \in \mathcal{I}}$ as in (9)

A variable left without an eligible partner, which only happens when $\mathcal{J}$ holds no variable of the required type, is scaled but not transvected ($\lambda_i = \mu_i = 0$).

Each re-description keeps the m existing constraints and appends $m' - m$ redundant ones. A redundant constraint is a convex combination of r existing constraints, drawn with replacement, whose right-hand side is then relaxed (Algorithm 2). Every constraint of M_0 is scaled by a positive factor so that the largest absolute value among its coefficients and its right-hand side is 1, and the constraints of $\mathsf{S}(\boldsymbol{T}', \boldsymbol{q}') \cdot M_0$ are rescaled in the same way before its re-description is drawn. A positive scaling of the constraints is itself a re-description, and it gives $\boldsymbol{\lambda}_i^\top \boldsymbol{b} \le 1$ for every convex combination $\boldsymbol{\lambda}_i$.

Algorithm 2 Sampling the re-description $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda b} + \boldsymbol{s})$

Require: instance $M = (\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}) \in \mathcal{M}_m$ with scaled constraints, fraction $\rho = m'/m - 1$, number of combined constraints r

$m' \leftarrow m + \mathrm{round}(\rho m)$
for $i = m+1, \dots, m'$ **do**
  draw $p_1, \dots, p_r$ uniformly from $[m]$ with replacement
  draw $(w_1, \dots, w_r)$ uniformly from the probability simplex
  $\boldsymbol{\lambda}_i \leftarrow \sum_{t=1}^{r} w_t \boldsymbol{e}_{p_t}$
  draw ν_i uniformly from $[0, 1]$ and set $s_i \leftarrow \nu_i \, (1 - \boldsymbol{\lambda}_i^\top \boldsymbol{b})$
end for
$\boldsymbol{\Lambda} \leftarrow$ the rows of $\boldsymbol{I}_m$ followed by $\boldsymbol{\lambda}_{m+1}^\top, \dots, \boldsymbol{\lambda}_{m'}^\top$, and $\boldsymbol{s} \leftarrow (0, \dots, 0, s_{m+1}, \dots, s_{m'})$
return $\mathsf{R}(\boldsymbol{\Lambda}, \boldsymbol{\Lambda b} + \boldsymbol{s})$

The relaxed right-hand side $\boldsymbol{\lambda}_i^\top \boldsymbol{b} + s_i$ thus lies uniformly between the implied value $\boldsymbol{\lambda}_i^\top \boldsymbol{b}$ and 1. The identity block of $\boldsymbol{\Lambda}$ keeps a copy of every constraint of M, and $\boldsymbol{\Lambda} \geq 0$ and $\boldsymbol{s} \geq 0$, so the re-description satisfies the conditions of Theorem 4.2.

D.6 Hyperparameter tuning

We tuned the pre-training hyperparameters of ReMILP and FORGE† separately with the multivariate TPE sampler of Optuna (Akiba et al., 2019), starting with 15 random warmup trials, for a total of 56 trials per model. The learning rates and numbers of steps were fixed. Each trial pre-trains the encoder with three seeds and scores it on 14 cells, which are binary solution and constraint activity prediction on seven instance families: CA medium, SC hard, CFLP medium, OTS easy, MMCN medium-BC, NNV easy, and LB hard. On each cell the encoder is frozen and a head is trained for 5 000 steps on training instances, and the cell is scored by its best validation KL.

Let $r_{s,c}$ be that validation KL for seed s and cell c, divided by the value the random encoder reaches on the same cell and seed. Denote by G_s the geometric mean over all cells c for a fixed seed s. A trial minimizes

$$\mathrm{mean}_s \, G_s + \frac{1}{2} \, \mathrm{std}_s \, G_s + \frac{2}{|S| \, |C|} \sum_{s,c} \max(r_{s,c} - 1, 0)$$

The second term penalizes variation across seeds and the third penalizes cells on which the encoder does worse than the random encoder. Table 5 lists the search space and the selected configurations.

Table 5: **Pre-training search space and selected values.** Ranges marked log are sampled on a logarithmic scale.

Model	Hyperparameter	Search space	Selected
ReMILP	Temperature τ	$[0.05, 0.5]$, log	0.090
	Negatives of each type K	$\{7, 15, 31, 63\}$	63
	Fraction of transformed variables $\kappa = \lvert\mathcal{I}\rvert/n$	$\{0.01, 0.02, 0.05\}$	0.05
	Fraction of redundant constraints $\rho = \frac{m'}{m} - 1$	$\{0, 0.05, 0.1, 0.2\}$	0.2
	Constraints combined per redundant row	$\{2, 3\}$	2
	Bound b_λ on λ_i, μ_i	$[1.5, 6]$, log	4.01
	Bound b_u on u_i	$[1, 4]$, log	1.12
FORGE†	Codebook size	$\{64, 128, 256, 1024\}$	256
	Codebook EMA decay	$\{0.8, 0.9, 0.95, 0.99\}$	0.99
	Commitment weight	$[0.01, 2]$, log	0.046

Table 6: **Binary solution prediction.** Frozen encoders, test KL per instance family, mean ± one standard deviation over three seeds. A value is bold when it is not within one standard deviation of the runner up.

Class	Difficulty	Random	ReMILP	FORGE†
IS	easy	0.588 ± 0.000	0.588 ± 0.000	0.588 ± 0.000
	medium	0.599 ± 0.000	0.599 ± 0.000	0.599 ± 0.000
CA	easy	0.242 ± 0.001	0.212 ± 0.001	0.217 ± 0.005
	medium	0.243 ± 0.001	0.210 ± 0.001	0.216 ± 0.005
SC	easy	0.057 ± 0.000	0.044 ± 0.002	0.047 ± 0.002
	medium	0.060 ± 0.000	0.047 ± 0.002	0.052 ± 0.003
	hard	0.065 ± 0.000	$\mathbf{0.051 \pm 0.002}$	0.058 ± 0.004
VC	easy	0.574 ± 0.006	0.505 ± 0.008	0.484 ± 0.017
	medium	0.582 ± 0.006	0.512 ± 0.010	0.496 ± 0.016
	hard	0.370 ± 0.000	0.371 ± 0.000	0.370 ± 0.000
GISP	easy	0.172 ± 0.001	0.170 ± 0.000	0.170 ± 0.000
	medium	0.136 ± 0.000	0.135 ± 0.000	0.135 ± 0.000
	hard	0.117 ± 0.000	0.115 ± 0.000	0.116 ± 0.000
MMCN	medium-BI	0.310 ± 0.004	0.252 ± 0.013	0.285 ± 0.020
	hard-BI	0.336 ± 0.013	$\mathbf{0.250 \pm 0.011}$	0.283 ± 0.021
CFLP	easy	0.245 ± 0.003	0.209 ± 0.023	0.228 ± 0.016
	medium	0.213 ± 0.000	0.178 ± 0.043	0.196 ± 0.015
OTS	easy	0.293 ± 0.018	$\mathbf{0.167 \pm 0.016}$	0.210 ± 0.025
	medium	0.339 ± 0.023	$\mathbf{0.188 \pm 0.011}$	0.242 ± 0.036
	hard	0.268 ± 0.002	$\mathbf{0.245 \pm 0.007}$	0.261 ± 0.004
MMCN	medium-BC	0.071 ± 0.004	$\mathbf{0.022 \pm 0.000}$	0.025 ± 0.000
NNV	easy	0.585 ± 0.007	0.553 ± 0.006	0.555 ± 0.000
LB	hard	0.512 ± 0.039	0.436 ± 0.045	0.385 ± 0.055

E PER-FAMILY RESULTS

Tables 6 and 7 give the frozen measurements behind Table 2 without aggregation, as the test KL of each instance family rather than a ratio to the random encoder, and Figures 6 to 9 do the same for the fine-tuned runs of Figure 2. The ratios of the main text are geometric means of these normalized values over instance families.

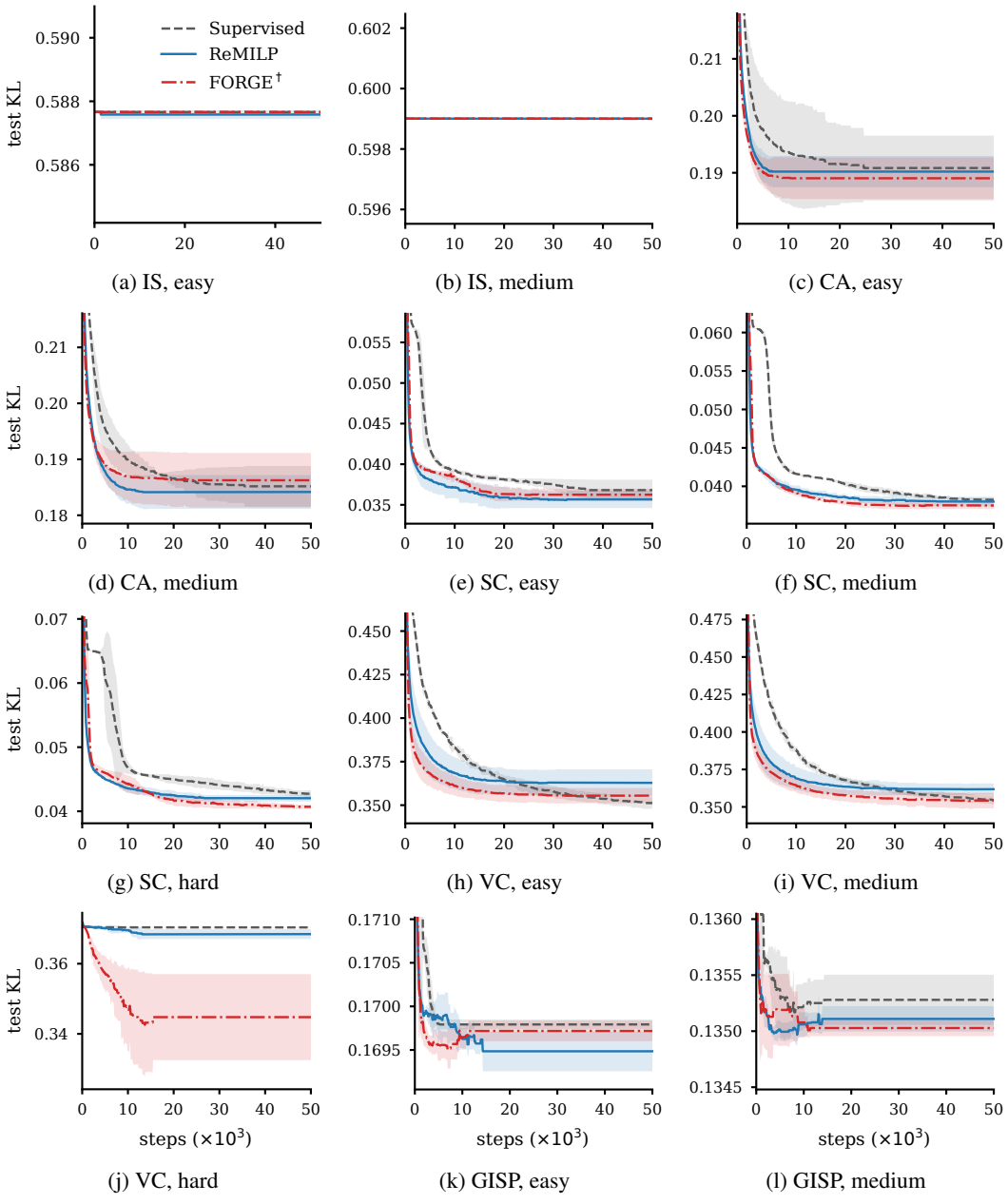
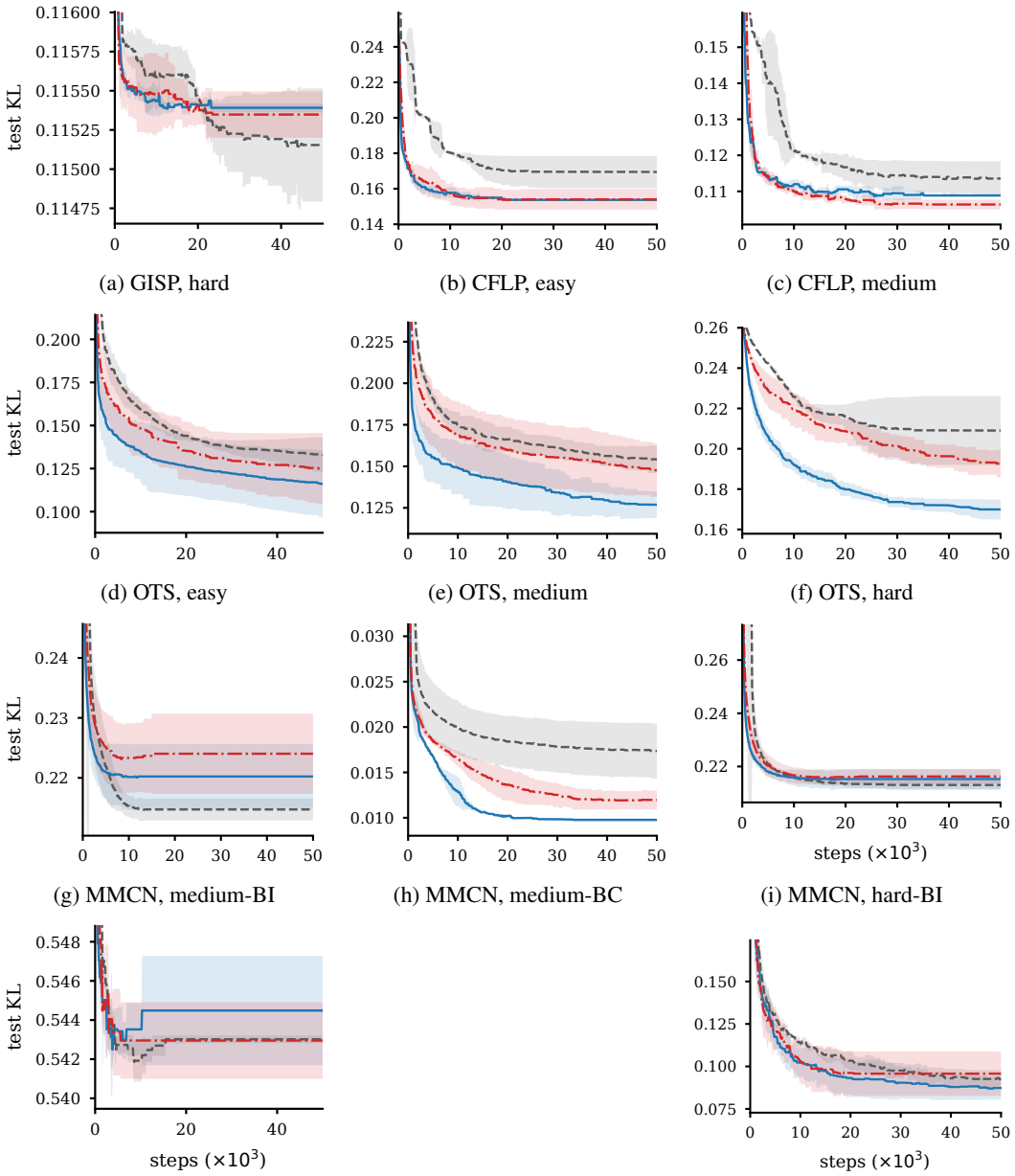


Figure 6: **Fine-tuning on binary solution prediction.** Part 1 of 2. Test KL of the checkpoint selected on validation, one panel per instance family, mean and one standard deviation over three seeds. Each panel carries its own vertical scale.



(a) GISP, hard (b) CFLP, easy (c) CFLP, medium

(d) OTS, easy (e) OTS, medium (f) OTS, hard

(g) MMCN, medium-BI (h) MMCN, medium-BC (i) MMCN, hard-BI

(j) NNV, easy (k) LB, hard

Figure 7: **Fine-tuning on binary solution prediction.** Part 2 of 2. Test KL of the checkpoint selected on validation, one panel per instance family, mean and one standard deviation over three seeds. Each panel carries its own vertical scale.

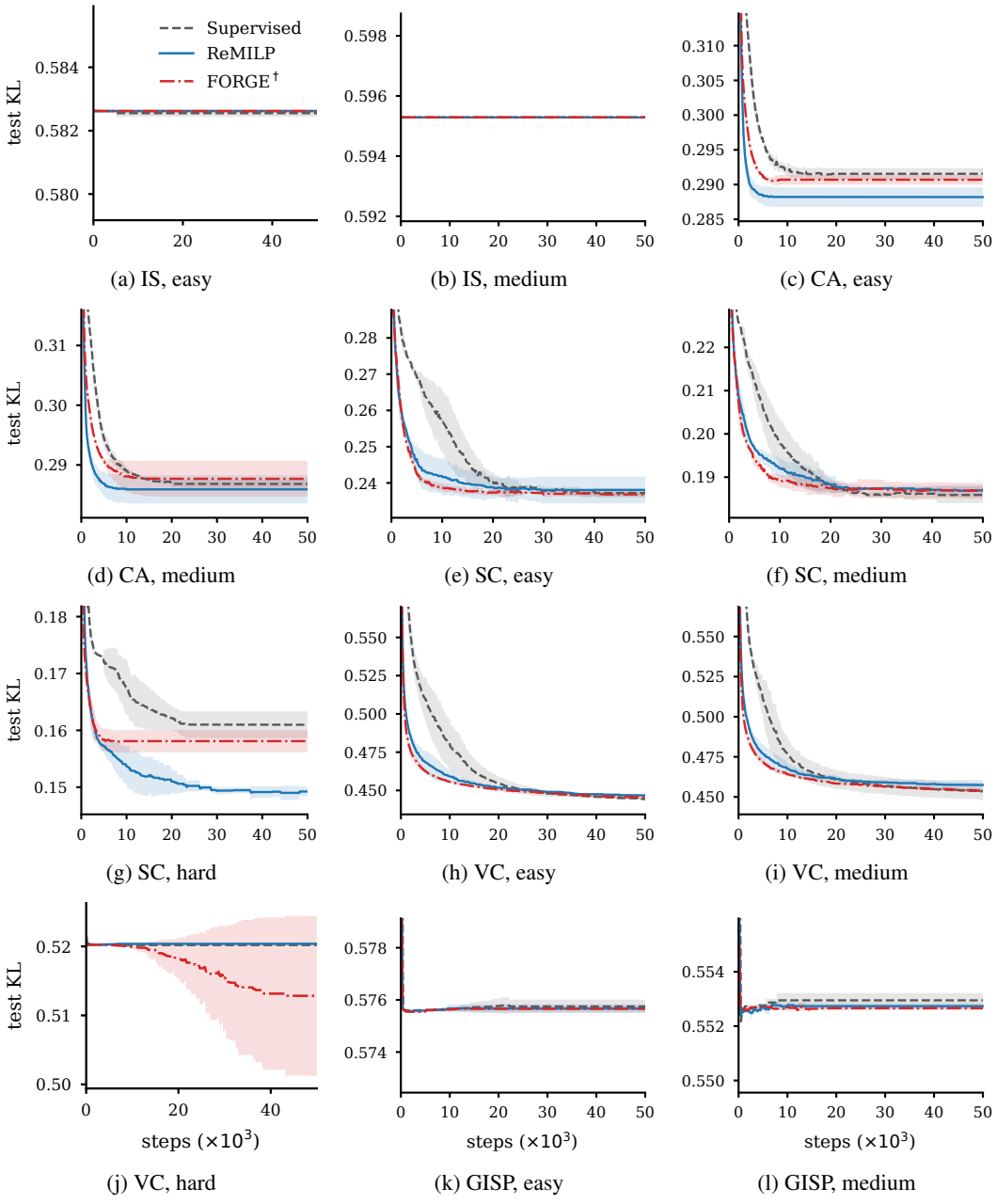
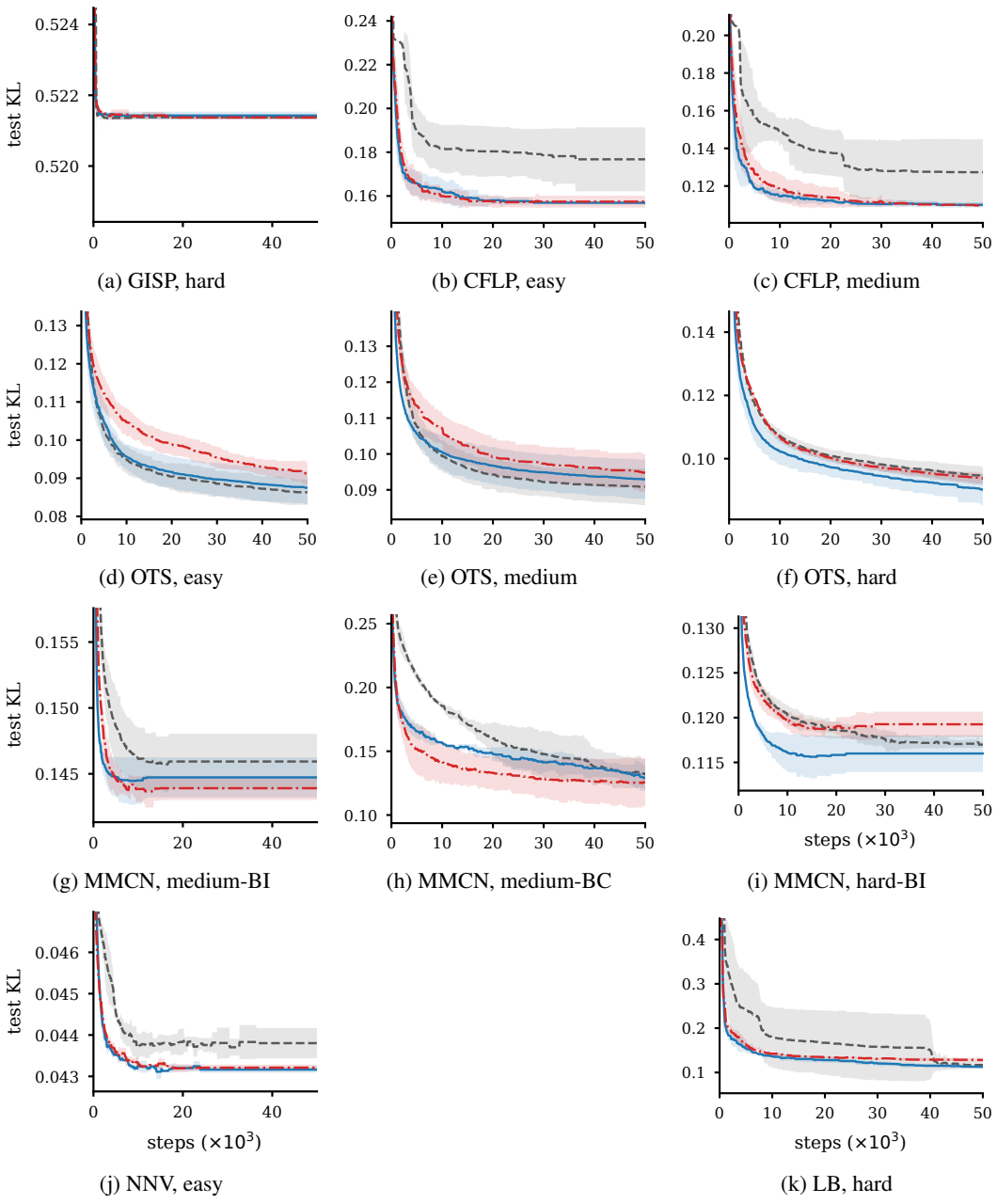


Figure 8: **Fine-tuning on constraint activity prediction.** Part 1 of 2. Test KL of the checkpoint selected on validation, one panel per instance family, mean and one standard deviation over three seeds. Each panel carries its own vertical scale.



(a) GISP, hard (b) CFLP, easy (c) CFLP, medium

(d) OTS, easy (e) OTS, medium (f) OTS, hard

(g) MMCN, medium-BI (h) MMCN, medium-BC (i) MMCN, hard-BI

(j) NNV, easy (k) LB, hard

Figure 9: **Fine-tuning on constraint activity prediction.** Part 2 of 2. Test KL of the checkpoint selected on validation, one panel per instance family, mean and one standard deviation over three seeds. Each panel carries its own vertical scale.

Table 7: Constraint activity prediction. Frozen encoders, test KL per instance family, mean $\pm$ one standard deviation over three seeds. Heads trained for 50 000 steps. The upper block holds the pairs with no continuous variables. A value is bold when not within one standard deviation of the runner up.

Class	Difficulty	Random	ReMILP	FORGE†
IS	easy	0.583 ± 0.000	0.583 ± 0.000	0.583 ± 0.000
	medium	0.595 ± 0.000	0.595 ± 0.000	0.595 ± 0.000
CA	easy	0.384 ± 0.004	0.312 ± 0.002	0.318 ± 0.008
	medium	0.394 ± 0.003	0.313 ± 0.002	0.321 ± 0.008
SC	easy	0.389 ± 0.012	0.286 ± 0.003	0.287 ± 0.003
	medium	0.322 ± 0.008	0.227 ± 0.004	0.229 ± 0.005
	hard	0.257 ± 0.002	0.175 ± 0.003	0.176 ± 0.005
VC	easy	0.631 ± 0.001	0.568 ± 0.007	0.571 ± 0.035
	medium	0.639 ± 0.001	0.577 ± 0.008	0.580 ± 0.035
	hard	0.520 ± 0.000	0.520 ± 0.000	0.520 ± 0.000
GISP	easy	0.577 ± 0.000	0.576 ± 0.000	0.576 ± 0.000
	medium	0.552 ± 0.000	0.553 ± 0.000	0.553 ± 0.000
	hard	0.523 ± 0.003	0.521 ± 0.000	0.522 ± 0.000
MMCN	medium-BI	0.183 ± 0.002	0.154 ± 0.003	0.161 ± 0.006
	hard-BI	0.158 ± 0.003	$\mathbf{0.128 \pm 0.002}$	0.136 ± 0.003
CFLP	easy	0.235 ± 0.003	0.214 ± 0.014	0.221 ± 0.005
	medium	0.208 ± 0.001	0.182 ± 0.039	0.189 ± 0.023
OTS	easy	0.150 ± 0.006	0.134 ± 0.004	0.130 ± 0.008
	medium	0.156 ± 0.005	0.135 ± 0.006	0.133 ± 0.006
	hard	0.125 ± 0.003	$\mathbf{0.109 \pm 0.002}$	0.122 ± 0.004
MMCN	medium-BC	0.311 ± 0.002	0.219 ± 0.007	0.225 ± 0.011
NNV	easy	0.049 ± 0.001	0.046 ± 0.000	0.046 ± 0.000
LB	hard	0.554 ± 0.003	0.445 ± 0.035	0.410 ± 0.020