BEYOND FEASIBILITY: SOLUTION-LANDSCAPE PRESERVING GENERATION OF MIXED-INTEGER LINEAR PROGRAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world Mixed-Integer Linear Programming (MILP) instances are scarce and costly to collect, limiting the training and evaluation of learning-based MILP solvers. Existing instance generation methods mainly preserve structural similarity, computational difficulty, feasibility, and boundedness. However, these properties do not ensure solution-landscape preservation: a generated instance may remain structurally similar and feasible while substantially altering the representative solutions and their distribution compared with those of the source instance. We propose Solution-Landscape Preserving Generation of Mixed-Integer Linear Programs (SLP-MILP), a controllable generation framework that explicitly incorporates source solution behavior into the generation process. SLP-MILP constructs solution-aware constraint groups based on the activation relationships between solutions and constraints as well as the constraint structure and regenerates selected groups through solution-conditioned masked generative reconstruction under an explicit structural modification budget. For each regenerated constraint direction, its right-hand side is determined by aligning the constraint-response profile with that of the original constraint on the source solution-landscape. A separate conformal retention safeguard prevents the aligned constraints from discarding excessive source probability mass. Experiments on multiple MILP families demonstrate the effectiveness of SLP-MILP in preserving solution-landscape characteristics while maintaining structural similarity.

1 INTRODUCTION

Mixed-Integer Linear Programming (MILP) is a foundation of modern operations research and optimization and has been widely applied to routing (Pillac et al., 2013), planning (Vasishth & Gigras, 2014), and job-shop scheduling (Abdullah & Abdolrazzagh-Nezhad, 2014), among other domains.

Traditional solution methods rely mainly on branch-and-bound solvers (Gurobi Optimization, LLC, 2026; SCIP, 2025; Ge et al., 2023). In recent years, with the development of deep learning, a large number of learning-based optimization methods have emerged. Owing to the advantages of machine learning (Fan et al., 2023) in pattern recognition and feature extraction, it has provided substantial help for optimization problems, especially large-scale ones (Nocedal, 2016). At the same time, learning-based MILP solving methods have become increasingly data-dependent: whether training learned solvers, performing hyperparameter tuning, or constructing stable baselines, all require a sufficient number of well-distributed MILP instances. However, real-world instances are usually difficult to release publicly at scale (Li et al., 2023), partly because instance construction and collection are costly, and partly because they involve commercially sensitive data, privacy, and domain knowledge. Therefore, generating representative MILP instances from limited real data is an important problem for supporting learning-based optimization and solver development.

To address this problem, existing LP/MILP instance generation methods fall into three categories. The first constructs specialized strategies to generate problem instances based on problem-specific characteristics and domain knowledge (Balas & Ho, 1980; Wiel & Sahinidis, 1995). Recently, methods in this category have also emerged that construct new instances based on block-structural characteristics of instance coefficients (Liu et al., 2024). The second directly constructs new instances

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.

by sampling a hyperparameter encoding space under fixed partial parameter characteristics (Bowly et al., 2019). The third employs deep learning-based methods to construct MILP instance generation frameworks. Although the third category emerged relatively late, existing work has progressively improved the quality of MILP generation in many aspects: from preserving structural statistics and graph structure (Geng et al., 2023), to preserving solving hardness (Guo et al., 2024), and further to explicitly guaranteeing feasibility/boundedness (Zhang et al., 2024).

Although these methods can generate usable instances, they largely ignore the distribution of instance solutions, which may matters when making decisions (Elmachtoub & Grigas, 2022). A generator may substantially alter most constraints while retaining only such trivial witnesses, even then, we still have $\mathcal{F}(\mathcal{P}') \neq \varnothing$, but this does not indicate that it has preserved the typical feasible solutions, near-optimal solutions, variable coordination patterns, and the structure of the observable solution space in the source instance. Therefore, $\mathcal{F}(\mathcal{P}') \neq \varnothing \not\Rightarrow \pi_{\mathcal{P}'} \approx \pi_{\mathcal{P}}$.

We argue that the evaluation target of high-fidelity MILP generation should not be limited to "whether the instance remains solvable," but should further focus on whether the generated instance still induces a solution-landscape whose feasible-solution structure is similar to that of the source instance. Therefore, we propose **SLP-MILP**, a **S**olution-**L**andscape **P**reserving **G**eneration of **M**ixed-**I**nteger **L**inear **P**rograms. SLP-MILP constructs solution-aware constraint groups from diverse source solutions, and uses a solution-conditioned masked generator to reconstruct selected groups under a prescribed modification ratio. For each generated constraint direction, it sets the RHS by matching its multi-quantile response profile to that of the original constraint under the source solution distribution. Because profile alignment alone may yield overly restrictive interacting constraints, a separate conformal safeguard bounds the loss of source probability mass. **The main contributions of the paper are:**

1) Unlike previous research papers that focus primarily on instance similarity and feasibility, we propose SLP-MILP, which preserves both structural and solution-distribution similarity by generating instances conditioned on source solution-landscape.
2) We conducts MILP instance generation tests on multiple general datasets, verifying the effectiveness of the proposed method from two aspects: traditional graph structure and the solution-landscape related metrics proposed in this paper.
3) We verify that SLP-MILP has good application potential in downstream task especially when it depends on solution distribution.

2 Related Work

LP/MILP based on machine learning. Recent advances have further strengthened the integration of machine learning with LP/MILP optimization. Existing approaches can still be broadly divided into two categories. The first directly predicts or generates high-quality solutions and then combines the predictions with local search or conventional solvers. Representative methods include contrastive solution prediction and variable fixing (Huang et al., 2024), unsupervised differentiable solution learning (Geng et al., 2025), and iterative prediction-correction for reliable problem reduction (Liu et al., 2025).Also, generative models have been extended to jointly model continuous and integer variables (Li et al., 2026). The second category embeds ML models into optimization solvers to improve individual algorithmic components. Recent examples include predicting Lagrangian multipliers (Demelas et al., 2024), learning presolve strategies for large-scale LPs (Kuang et al., 2025), node selection in branch-and-bound (Zhang et al., 2025) and learning cutting-plane generation or configuration (Deza et al., 2025). Despite these advances, feasibility, data efficiency, and out-of-distribution generalization remain major challenges. Recent work therefore increasingly focuses on robust hybrid learning-and-optimization frameworks, such as adversarial instance augmentation for improving solver generalization (Liu et al., 2026).

LP/MILP instance generation. Existing work on MILP instance generation can be broadly categorized into knowledge-based, heuristic-based, and learning-based methods. Here, we focus on learning-based approaches. G2MILP (Geng et al., 2023) introduced a deep learning framework that generates instances by local node destruction and reconstruction while preserving graph structure, but it does not reliably ensure feasibility. DIG-MILP (Wang et al., 2024) addresses MILP data scarcity with a VAE-based generator whose primal-dual feasibility conditions guarantee feasibility

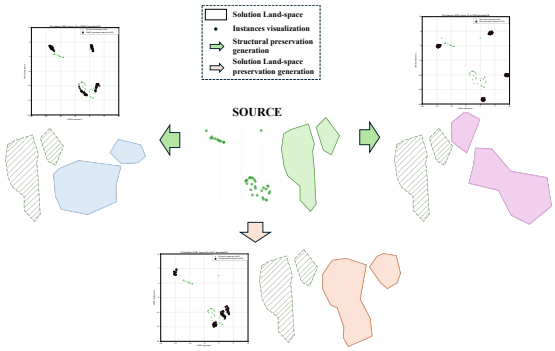


Figure 1: Difference of generation by structural preservation and solution-landscape preservation. The grid scatter plots and the colored regions represent the instance and solution-landscape distribution, respectively.

and boundedness, although its generated instances may still deviate from the training distribution. ACM-MILP (Guo et al., 2024) uses constraint communities to select nodes for masking and aims to preserve solving hardness. MILP-FBGen (Zhang et al., 2024) starts from a feasible source solution and uses a diffusion model to generate feasible instances.

In summary, the application of machine learning in the field of operations research requires a large number of instance. Existing methods primarily preserve structural similarity or feasibility, but they do not use the source solution landscape to guide generation. Our SLP-MILP accounts for both instance similarity and feasibility while preserving the source solution landscape, which is important in practical application scenarios.

3 PRELIMINARIES

3.1 MILP INSTANCE GENERATION

This paper mainly focuses on the problem of MILP instance generation. A MILP problem can be represented as $\mathcal{P} = (A, \mathbf{b}, \mathbf{c}, \mathbf{l}, \mathbf{u}, \mathcal{I})$, where $A \in \mathbb{R}^{m\times n}$ is the constraint coefficient matrix, $\mathbf{b} \in \mathbb{R}^m$ is the right-hand side, $\mathbf{c} \in \mathbb{R}^n$ is the objective coefficient vector, $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ are the variable lower and upper bounds, $\mathcal{I} \subseteq \{1, \ldots, n\}$ is the index set of integer variables, and $\mathcal{F}(\mathcal{P})$ denotes its feasible region. The corresponding optimization problem is

$$\begin{aligned} &\min_{\mathbf{x}} \quad \mathbf{c}^\top \mathbf{x}, \\ \text{s.t.} \quad &A\mathbf{x} \le \mathbf{b}, \quad \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \quad x_j \in \mathbb{Z}, \quad j \in \mathcal{I}. \end{aligned} \tag{1}$$

3.2 CONDITIONED SOLUTION DISTRIBUTION

A feasible region is a set rather than a probability distribution. There is therefore no unique objective notion of a "true feasible-solution distribution". To give solution-landscape preservation a precise meaning, we introduce a conditioned solution distribution. Following the Gibbs/energy-based view of converting an energy function into a probability distribution (LeCun et al., 2006), given an inverse temperature $\beta \ge 0$, we define

$$\pi^{\beta}_{\mathcal{P}}(\mathbf{x}) = \frac{1}{Z^{\beta}_{\mathcal{P}}} \mathbb{I}[\mathbf{x} \in \mathcal{F}(\mathcal{P})] \exp[-\beta \tilde{f}_{\mathcal{P}}(\mathbf{x})] \nu_{\mathcal{P}}(\mathbf{x}) \tag{2}$$

where $\nu_{\mathcal{P}}$ is a base sampling distribution, $Z^{\beta}_{\mathcal{P}}$ is the normalizing constant, and $\tilde{f}_{\mathcal{P}}$ is a normalized objective: $\tilde{f}_{\mathcal{P}}(\mathbf{x}) = \frac{\mathbf{c}^\top \mathbf{x} - L_{\mathcal{P}}}{U_{\mathcal{P}} - L_{\mathcal{P}} + \varepsilon}$. Here $L_{\mathcal{P}}$ and $U_{\mathcal{P}}$ denote lower and upper estimates of the objective value, respectively. Different values of β encode different task preferences. When $\beta = 0$, the distribution emphasizes feasible-region coverage. When β is large, the distribution concentrates on near-optimal regions. To preserve both broad feasibility and high-quality near-optimal solutions, we use a multi-temperature mixture $\pi_{\mathcal{P}} = \sum_{r=1}^{R} \alpha_r \pi^{\beta_r}_{\mathcal{P}}$,$\sum_{r=1}^{R} \alpha_r = 1$,where $\alpha_r \ge 0$ is the weight of

the r-th temperature component. For each source instance, we approximate $\pi_{\mathcal{P}}$ by a finite source solution pool and separate the pool into disjoint training, calibration, and test subsets. The implementation details including pool construction, induced empirical distribution, and the data-splitting protocol are described in Appendix C.2.

3.3 Solution-Landscape Preservation

The core question addressed in this paper is whether the solution distribution of generated instances can preserve the main distribution of the source instances solutions. Let $\pi_{\mathcal{P}'}$ denote the conditioned solution distribution on the generated instance. But in practice, exact distributional equality is difficult to verify directly. We therefore decompose preservation into several observable indicators shown as metrics in Appendix C.4.

A key point is that the existence of one feasible solution does not mean $\pi_{\mathcal{P}'} \approx \pi_{\mathcal{P}}$. For example, the zero vector is naturally feasible in some Maximum Independent Set formulations, while the all-one vector is naturally feasible in some Set Covering formulations. A generator may preserve only such default solutions while destroying the majority of meaningful solutions. Therefore, single-witness feasibility is only a minimal requirement and cannot replace source-solution distribution preservation. The method and evaluation protocol in this paper are both designed around this observation. As shown in Figure 1, we can clearly see that in structure preserving generation tasks, although the distribution of instances is similar, there is a significant difference in the distribution of their solution spaces. This is also an important motivation for us to carry out solution-landscape preservation generation.

4 Methodology

We present SLP-MILP to generate new MILP instances while preserving the solution-landscape characteristics of source instances and maintaining structural similarity and controllable novelty. As illustrated in Figure 2, constraints and variables are shown in blue and orange, respectively. SLP-MILP constructs a diverse source solution pool by solver-assisted sampling, derives solution-aware constraint groups, reconstructs selected groups with a solution-conditioned latent generator, and finally aligns the regenerated response profiles with those of the source constraints while enforcing a conformal source-mass safeguard.

4.1 Diverse Source Solution Pool

The conditioned solution distribution $\pi_{\mathcal{P}}$ defined in Section 3.2 specifies the solution landscape that we aim to preserve. Since this distribution is not available in closed form for a general MILP, we approximate it with a finite source solution pool

$$\mathcal{S}_{\mathcal{P}} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}\} \subset \mathcal{F}(\mathcal{P}). \tag{3}$$

The temperature-dependent weights of the induced empirical distribution over $\mathcal{S}_{\mathcal{P}}$ are provided in Appendix C.2. Rather than assuming exact samples from $\pi_{\mathcal{P}}$, we construct $\mathcal{S}_{\mathcal{P}}$ to cover both diverse feasible and near-optimal regions.

Starting from a feasible solution $\mathbf{x}^{(t)}$, we use large neighborhood search (LNS) and solver-assisted sampling to select a variable subset $D_t \sim \pi_{\varphi}(D \mid \mathcal{P}, \mathbf{x}^{(t)})$, where π_{φ} is the variable-selection policy. All variables outside D_t are fixed, yielding

$$\mathcal{N}(\mathbf{x}^{(t)}, D_t) = \left\{ \mathbf{x} \in \mathcal{F}(\mathcal{P}) : x_j = x_j^{(t)},\ j \notin D_t \right\}. \tag{4}$$

Because $\mathbf{x}^{(t)} \in \mathcal{N}(\mathbf{x}^{(t)}, D_t)$, each neighborhood is nonempty. To improve solution diversity, we perturb the objective when solving each neighborhood:

$$\begin{aligned} \min_{\mathbf{x}} \quad & (\mathbf{c} + \boldsymbol{\xi}_t)^{\top} \mathbf{x}, \\ & \boldsymbol{\xi}_t \sim \mathcal{N}(0, \sigma^2 I). \end{aligned} \tag{5}$$

For integer and continuous variables, candidate solutions are compared using the L_1 distance, for binary variables, we use the Hamming distance. A candidate is rejected or resampled if its minimum

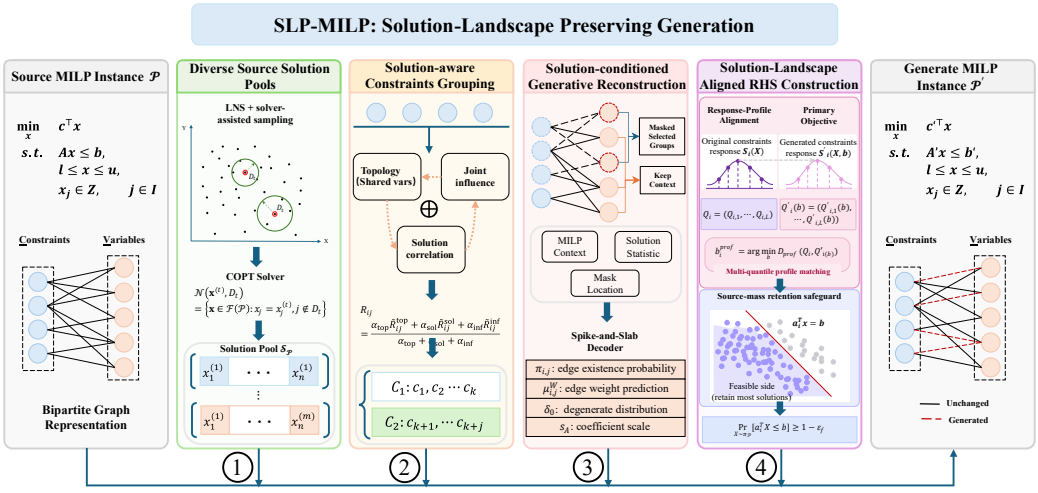


Figure 2: Framework of our SLP-MILP.

distance to the existing pool is below a threshold. After that, the solution pools will be divided into disjoint training, calibration, and test subsets. The details are summarized in Appendix C.2.

4.2 Solution-Aware Constraint Grouping

Conventional constraint grouping usually relies solely on the topological structure of the constraint variable bipartite graph, but it is also possible to group constraints that are topologically close but behavior independent based solely on this structural information, while ignoring constraints with stable functional relationships. Therefore, we simultaneously use topological relationships and source solution behavior to construct constraint groups.

For the i-th inequality constraint $\mathbf{a}_i^\top \mathbf{x} \le b_i$, define its normalized slack and soft activity as

$$s_i(\mathbf{x}) = \frac{b_i - \mathbf{a}_i^\top \mathbf{x}}{|b_i| + \|\mathbf{a}_i\|_1 + \varepsilon}, \qquad u_i(\mathbf{x}) = \exp\left(-\frac{|s_i(\mathbf{x})|}{\tau_s}\right). \tag{6}$$

Hence, $u_i(\mathbf{X})$ with $\mathbf{X} \sim \pi_{\mathcal{P}}$ describes how frequently constraint i is close to its boundary over the source solution landscape. In practice, these quantities are estimated from $\mathcal{S}_{\mathcal{P}}^{\text{train}}$.

For constraints i and j, we characterize their relation from both MILP topology and solution behavior:

$$\begin{aligned} R_{ij}^{\text{top}} &= \frac{|\mathcal{N}(i) \cap \mathcal{N}(j)|}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(j)|}}, \\ R_{ij}^{\text{sol}} &= \left|\text{Corr}_{\mathbf{X}\sim\pi_{\mathcal{P}}}\left(u_i(\mathbf{X}), u_j(\mathbf{X})\right)\right|, \qquad R_{ij}^{\text{inf}} = \mathbb{E}_{\mathbf{X}\sim\pi_{\mathcal{P}}}\left[u_i(\mathbf{X})u_j(\mathbf{X})\right]. \end{aligned} \tag{7}$$

The three terms measure shared-variable structure, similarity of activity patterns, and simultaneous boundary influence, respectively. We combine them as

$$R_{ij} = \lambda_{\text{top}} R_{ij}^{\text{top}} + \lambda_{\text{sol}} R_{ij}^{\text{sol}} + \lambda_{\text{inf}} R_{ij}^{\text{inf}}, \qquad \sum_{\star\in\{\text{top,sol,inf}\}} \lambda_\star = 1, \tag{8}$$

and partition the resulting relation graph into constraint groups $\{\mathcal{C}_1, \ldots, \mathcal{C}_G\}$ (Blondel et al., 2008).

Given a replacement ratio η, groups are sampled until the selected rows $\mathcal{R}_\eta$ reach the target modification budget. The resulting mask therefore respects not only graph topology but also constraint interactions under $\pi_{\mathcal{P}}$. Candidate-pair screening and grouping details are provided in Appendix A.2.

4.3 Solution-Conditioned Masked Generative Reconstruction

After determining where to modify the instance, we learn *what* constraint directions should replace the masked rows. Let $Y_{\mathcal{R}}$ denote the original parameters of the selected rows, which are used only

Table 1: Statistical distributional similarity scores. SLP-MILP is shaded, and bold indicates the best result in each dataset column.

Replacement Ratio	Model	MIK	CA	IP	IS
$\eta = 0.1$	G2MILP	0.967	**0.991**	0.958	0.945
	ACM-MILP	0.972	0.989	0.878	0.961
	MILP-FBGen	**0.991**	0.736	0.779	0.671
	SLP-MILP	0.887	0.982	**0.958**	**0.970**
$\eta = 0.2$	G2MILP	0.953	**0.982**	0.927	0.931
	ACM-MILP	0.957	0.979	0.839	0.954
	MILP-FBGen	**0.990**	0.672	0.654	0.601
	SLP-MILP	0.857	0.933	**0.930**	**0.984**
$\eta = 0.5$	G2MILP	0.883	0.955	0.870	0.920
	ACM-MILP	0.908	**0.957**	0.697	0.917
	MILP-FBGen	**0.977**	0.583	0.547	0.517
	SLP-MILP	0.846	0.889	**0.898**	**0.922**

as reconstruction targets during training. From $\mathcal{S}_{\mathcal{P}}^{\text{train}}$, we precompute solution-aware statistics for constraints and variables and denote them collectively by H_{sol}.

For a masked row set $\mathcal{R}$, the observable context is encoded as

$$C_{\mathcal{R}} = \text{Enc}\left(\mathcal{P}_{\setminus\mathcal{R}}, H_{\text{sol}}^{\text{obs}}, M_{\mathcal{R}}\right), \tag{9}$$

where $\mathcal{P}_{\setminus\mathcal{R}}$ denotes the masked MILP, $M_{\mathcal{R}}$ specifies the masked locations, and $H_{\text{sol}}^{\text{obs}}$ contains solution statistics observable after masking. Row-specific solution features derived from the original parameters of a masked constraint are masked together with that row to avoid target leakage. Therefore, the empirical source solution landscape affects generation through the conditioning context rather than through a solver-dependent objective.

Following the variational autoencoding framework (Kingma & Welling, 2014), we model the conditional distribution of the masked rows using a latent variable $\mathbf{z}_{\mathcal{R}}$. During training, the posterior and conditional prior are

$$q_\phi(\mathbf{z}_{\mathcal{R}} \mid Y_{\mathcal{R}}, C_{\mathcal{R}}), \qquad p_\psi(\mathbf{z}_{\mathcal{R}} \mid C_{\mathcal{R}}), \tag{10}$$

respectively. Given $\mathbf{z}_{\mathcal{R}}$ and $C_{\mathcal{R}}$, the decoder reconstructs both coefficient support and nonzero values using a Spike-and-Slab distribution (Mitchell & Beauchamp, 1988; George & McCulloch, 1993),

$$p_\theta(A_{ij} \mid \mathbf{z}_{\mathcal{R}}, C_{\mathcal{R}}) = (1 - \pi_{ij})\delta_0 + \pi_{ij}\mathcal{N}(\mu_{ij}^W, \sigma_W^2 s_A^2). \tag{11}$$

where π_{ij} is the edge-existence probability, μ_{ij}^W is the normalized edge-weight prediction, and δ_0 is the degenerate distribution representing edge absence. The scale s_A is defined as

$$s_A = \max\left(\sqrt{\frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} A_e^2}, \varepsilon\right). \tag{12}$$

The generator is trained by maximizing the conditional evidence lower bound (ELBO)

$$\begin{aligned}\log p_{\theta,\psi}(Y_{\mathcal{R}} \mid C_{\mathcal{R}}) \geq\ & \mathbb{E}_{q_\phi}\left[\log p_\theta(Y_{\mathcal{R}} \mid \mathbf{z}_{\mathcal{R}}, C_{\mathcal{R}})\right] \\ & - D_{\text{KL}}\left(q_\phi(\mathbf{z}_{\mathcal{R}} \mid Y_{\mathcal{R}}, C_{\mathcal{R}}) \| p_\psi(\mathbf{z}_{\mathcal{R}} \mid C_{\mathcal{R}})\right).\end{aligned} \tag{13}$$

At generation time, the original masked rows are unavailable, we instead sample $\mathbf{z}_{\mathcal{R}} \sim p_\psi(\mathbf{z}_{\mathcal{R}} \mid C_{\mathcal{R}})$ and decode new coefficient rows $A'_{\mathcal{R}}$. The solution-aware node features, network architecture, reconstruction losses, and latent parameterization are detailed in Appendices A.1, A.3 and B.1.

4.4 Solution-Landscape-Aligned RHS Construction

Section 4.3 generates new coefficient directions $\mathbf{a}'_i$ for the selected rows. A direction alone, however, does not determine how the resulting constraint interacts with the source solution landscape. We therefore construct the RHS through response-profile alignment, while imposing source-mass retention as a safeguard against overly restrictive generated constraints.

For an original constraint $\mathbf{a}_i^\top \mathbf{x} \leq b_i$, let $S_i(\mathbf{X})$ denote its normalized response under $\mathbf{X} \sim \pi_{\mathcal{P}}$. For a generated direction $\mathbf{a}'_i$ and candidate RHS b, let $S'_i(\mathbf{X}; b)$ denote the corresponding generated response. For a fixed quantile grid $\mathcal{T} \subset (0, 1)$, let $\mathbf{Q}_i$ and $\mathbf{Q}'_i(b)$ be the source and generated response-profile vectors, and let D_{prof} be their weighted L_1 discrepancy. We determine the profile-aligned RHS by minimizing the response-profile discrepancy while retaining at least a $1-\epsilon_f$ fraction of the source-solution probability mass:

$$\begin{aligned} b_i^{\text{prof}} = \arg\min_{b} \quad & D_{\text{prof}}\left(\mathbf{Q}_i, \mathbf{Q}'_i(b)\right) \\ \text{s.t.} \quad & \Pr_{\mathbf{X}\sim\pi_{\mathcal{P}}}\left[\mathbf{a}_i'^\top \mathbf{X} \leq b\right] \geq 1 - \epsilon_f. \end{aligned} \tag{14}$$

Since the population quantities are unavailable, we estimate b_i^{prof} from the fixed solution-training pool and apply a one-sided conformal correction on the independent calibration subset. The empirical estimator, correction rule, and proof are provided in Appendices B.2 and B.3.

5 Experiments

5.1 Experimental Setup

Datasets. We use four MILP families: Mixed Inter Knapsack (MIK) (Atamtürk, 2003), Combinatorial Auction (CA) (Leyton-Brown et al., 2000), Balanced Item Placement (IP) (Gasse et al., 2022), and Independent Set (IS) (Gasse et al., 2019). Each family follows fixed training, validation, and test splits, and the variable order is kept consistent between source and generated instances. Dataset statistics and split rules are provided in Appendix C.1.

Baselines. Since this paper focuses on learning-based MILP instance generation methods, the Bowly (Bowly, 2019) method is not considered when selecting baselines. We compare SLP-MILP with G2MILP, ACM-MILP, and MILP-FBGen, which represent graph-structure-preserving, hardness-preserving, and feasibility-oriented generation, respectively. Since MILP-FBGen has no publicly available official implementation, we reproduce it from the original paper. All methods use the same source instances, generation budget, and solver configuration, and the main comparison matches the realized edit ratio. Implementation details are provided in Appendix C.3. For each source instance and parameter setting, we generate and evaluate 20 instances and report their average metrics.

Generation and solution-pool protocol. For each source instance, the source solution pool is divided into disjoint solution-training, calibration, and test subsets. The solution-training subset provides solution-aware statistics, the calibration subset is used only for the conformal retention safeguard, and the test subset is reserved exclusively for final evaluation. We evaluate several levels of η_{mod} and ϵ_f. Here, η_{mod} controls the structural modification budget, whereas ϵ_f controls the allowed failure probability of source-solution retention and is not a direct target for the global distance between $\pi_{\mathcal{P}}$ and $\pi_{\mathcal{P}'}$. Because constraint groups are atomic, the realized modification ratio $\widehat{\eta}_{\text{mod}}$ may exceed its target. We therefore compare methods at matched realized edit ratios. For each generated instance, we independently construct its feasible solution pool using the same solver version, time limit, thread count, and random seed as for the source solution pool.

Evaluation metrics. The main evaluation uses eight metrics mentioned in section 3.3. All metrics compare the source solution pool with the generated solution pool. When multiple generated instances are available, the metric is computed for each instance and then averaged over instances sharing the same source and method. The specific meaning and calculation formula of the indicators are shown in Appendix C.4.

Downstream task. We further evaluate a downstream variable-fixing task, where high-confidence variable assignments inferred from high-quality solutions of generated instances are transferred to the corresponding source MILPs. The goal is to examine whether solution-landscape information preserved by generated instances can provide reliable variable preferences for reducing the effective search space and accelerating optimization. We report the average solving time and objective value, with implementation details provided in Appendix C.5.

Table 2: Solution-landscape distributional similarity on datasets($\eta = 0.1$). ↓ means the smaller the better, ↑ means the larger the better.

Indicators	**Chamfer↓**				**SymNN↓**				**Supp. Jaccard↑**				**Act. Jaccard↑**			
Model	**MIK**	**CA**	**IP**	**IS**	**MIK**	**CA**	**IP**	**IS**	**MIK**	**CA**	**IP**	**IS**	**MIK**	**CA**	**IP**	**IS**
G2MILP	0.191	0.096	0.138	0.244	0.208	0.097	0.145	0.289	0.858	0.263	0.007	0.572	0.439	0.717	0.019	0.632
ACM-MILP	0.268	0.093	0.138	0.221	0.299	0.094	0.144	0.249	0.842	**0.287**	0.013	0.573	0.513	0.724	0.018	0.626
MILP-FBGen	2.093	0.125	0.129	0.192	2.117	0.127	0.126	0.238	0.404	0.075	**0.315**	0.632	0.158	0.667	0.025	0.642
SLP-MILP	**0.127**	**0.072**	**0.103**	**0.113**	**0.203**	**0.078**	**0.114**	**0.194**	**0.934**	0.283	0.127	**0.872**	**0.747**	**0.858**	**0.089**	**0.776**
Indicators	**MMD↓**				**SWD↓**				**Corr. shift↓**				**Obj. shift↓**			
Model	**MIK**	**CA**	**IP**	**IS**	**MIK**	**CA**	**IP**	**IS**	**MIK**	**CA**	**IP**	**IS**	**MIK**	**CA**	**IP**	**IS**
G2MILP	1.096	0.557	0.495	0.858	0.153	0.046	0.096	0.124	0.138	0.022	0.116	0.180	22.94	11.76	19.16	16.38
ACM-MILP	1.201	0.528	**0.050**	0.643	0.126	0.052	0.098	**0.109**	0.121	0.023	0.116	0.171	22.51	10.41	19.62	**2.082**
MILP-FBGen	1.126	0.523	0.198	0.994	0.150	**0.038**	0.092	0.123	0.207	0.053	0.132	0.147	36.18	23.16	**9.868**	28.79
SLP-MILP	**0.923**	**0.252**	0.110	**0.452**	**0.121**	0.063	**0.076**	0.119	**0.097**	**0.017**	**0.105**	**0.054**	**17.63**	**8.437**	11.27	7.263

Table 3: Downstream variable-fixing results at $\eta = 0.1$. The Source row reports the average solving result of the original source MILPs without variable fixing, while the remaining rows report results after applying variable fixing based on solutions from generated instances.

Model	**Fixed_vars**	**Time**	**Objective**
Source	0	171.79s	674.76
G2MILP	48	–	–
ACM-MILP	114	144.69s(15.8%)	617.22(8.53%)
MILP-FBGen	172	112.35s(34.6%)	612.46(9.23%)
SLP-MILP	**327**	87.04s(**49.3%**)	650.83(**3.55%**)

5.2 Results and discussion

Instance similarity. This article uses the instance similarity evaluation metric proposed by Geng et al. (2023) to calculate the similarity between training and generated instances generated by a total of four methods on four different datasets. The statistical distributional similarity is shown in Table 1. From the table, we can see that our proposed SLP-MILP performs comparably to G2MILP and ACM-MILP (achieving the best results on the IP/IS datasets, while on the other two datasets the gap to the best is very small). In addition, together with the subsequent Table 2, this further demonstrates that the method proposed in this paper, while guaranteeing the solution-landscape, also achieves a good balance in terms of instance structure similarity.

Solution-Landscape distributional similarity. We proposes 8 indicators for evaluating solution-landscape similarity, as detailed inC.4. During the experiment, four methods were used to generate new instances on four corresponding datasets and then perform metric calculations. Here, we only present the results with $\eta = 0.1$ in Table 2. From the results in the table, it can be seen that our proposed method achieves the best value on the vast majority of metrics, and for the very few metrics where it is not optimal, the gap to the best value is also very small.

Variable-fixing task. Using the Independent Set (IS) problem as an example, Table 3 shows that better preservation of the source solution structure leads to more variables exceeding the fixing threshold. In the current experimental setting, transferring the fixed variables obtained from generated instances back to the original source problem makes the resulting reduced problem infeasible for G2MILP. Therefore, its solving time and optimal objective are not reported. For the remaining methods, variable fixing reduces the solving time by reducing branching in the search process, and SLP-MILP achieves the largest reduction, consistent with its largest number of fixed variables. Although none of the reported methods reaches the source optimal value, SLP-MILP achieves the best objective.

Table 4: Ablation study of solution-pool construction and solution-aware constraint grouping. Dataset: CA, $\eta = 0.1$.

Method	Chamfer ↓	Act. Jaccard ↑	Obj. shift ↓
SLP-MILP	**0.072**	**0.858**	8.437
w/ Fixed perturbation	0.533	0.352	**8.133**
w/ Uniform perturbation	0.287	0.649	11.25
w/o Solution-aware grouping	0.304	0.427	9.218

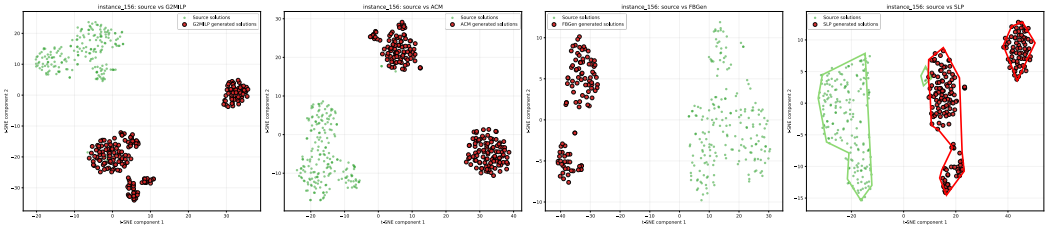


Figure 3: The t-SNE visualization of solution pool representations for CA

5.3 Ablation Study

We conduct ablation studies to investigate the contribution of two key components in SLP-MILP: diverse source solution pool construction and solution-aware constraint grouping. Specifically, we consider two variants: (1) replacing the Gaussian perturbation-based solution sampling strategy with alternative perturbation schemes, and (2) removing solution-aware information from constraint grouping and only retaining topology-based grouping. To evaluate the impact of each component on solution-landscape preservation, we report three representative metrics: Chamfer distance, Activation Jaccard, and Objective shift.

We compare different variants of SLP-MILP to evaluate the contribution of each component. As shown in Table 4, removing either component leads to degraded solution-landscape preservation. The sampling ablation results show that alternative perturbation strategies reduce the similarity of generated solution distributions, indicating that diverse solution pools provide important guidance for generation. The grouping ablation demonstrates that topology information alone is insufficient to capture solution-related constraint interactions. By incorporating solution-aware correlations, SLP-MILP better preserves activation patterns and objective distributions of the source instances.

5.4 Visualization

As shown in Figure 3, we separately apply t-SNE to the solution spaces of the different methods for intuitive visualization. The solution pool generated by SLP-MILP in the embedding space has a shape more similar to that of the source solution pool, and it effectively covers different clusters of the source solutions. These results indicate that SLP-MILP better preserves the solution-landscape distribution than the other baseline methods.

6 Conclusion

We proposed SLP-MILP for solution-landscape-preserving MILP instance generation. It constructs solution-aware constraint groups, reconstructs selected groups with a conditional VAE, and calibrates RHS values through response-profile alignment and a conformal retention safeguard. Experiments show competitive structural similarity, improved solution-landscape preservation, and the effectiveness of transferring generated solution information to downstream variable-fixing tasks. The main limitation is the cost of constructing source solution pools and extracting solution-aware features; surrogate models provide a promising direction for reducing this cost.

AI USE STATEMENT

In this work, we utilized AI tools in the following areas.

- In the section of related work, we use AI tools to review the current summary to prevent any omissions in the latest related work.
- When reproducing the MILP-FBGen (Zhang et al., 2024) method described in the paper, AI tools are used to guide how the diffusion model can be integrated into the entire reproduction process.
- The paper utilized AI tools to assist in grammar checking and language polishing.

All data, text, and processes generated with the assistance of AI tools have undergone manual verification and revision. We take full responsibility for the content of our work.

REPRODUCIBILITY STATEMENT

Details such as the architecture, training process, and experimental setup of SLP-MILP are provided in Section 4. Additional experimental details are reported in Appendix C. The source code with remaining implementation details will be released upon acceptance.

REFERENCES

Salwani Abdullah and Majid Abdolrazzagh-Nezhad. Fuzzy job-shop scheduling problems: A review. *Information Sciences*, 278:380–407, 2014.

Alper Atamtürk. On the facets of the mixed–integer knapsack polyhedron. *Mathematical Programming*, pp. 145–175, 2003.

Egon Balas and Andrew Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study. *Math. Programming Stud.*, pp. 37–60, 1980.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10):P10008, 2008.

Simon Bowly, Kate Smith-Miles, Davaatseren Baatar, and Hans Mittelmann. Generation techniques for linear programming instances with controllable properties. *Mathematical Programming Computation*, 12(5), 2019.

Simon Andrew Bowly. *Stress testing mixed integer programming solvers through new test instance generation methods*. PhD thesis, University of Melbourne, 2019.

Francesco Demelas, Joseph Le Roux, Mathieu Lacroix, and Axel Parmentier. Predicting lagrangian multipliers for mixed integer linear programs. In *41st International Conference on Machine Learning, ICML 2024*, 2024.

Arnaud Deza, Elias B. Khalil, Zhenan Fan, Zirui Zhou, and Yong Zhang. Learn2aggregate: Supervised generation of chvatal-gomory cuts using graph neural networks. In *39th AAAI Conference on Artificial Intelligence*, 2025.

Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, pp. 9–26, 2022.

Zhenan Fan, Xinglu Wang, Oleksandr Yakovenko, Abdullah Ali Sivas, Owen Ren, Yong Zhang, and Zirui Zhou. Smart initial basis selection for linear programs. In *40th International Conference on Machine Learning, ICML 2023*, 2023.

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

Maxime Gasse, Simon Bowly, Quentin Cappart, et al. The machine learning for combinatorial optimization competition (ml4co): Results and insights. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, pp. 220–231. PMLR, 2022.

Dongdong Ge, Qi Huangfu, Zizhuo Wang, Jian Wu, and Yinyu Ye. Cardinal Optimizer (COPT) user guide. https://guide.coap.online/copt/en-doc, 2023.

Zijie Geng, Xijun Li, Jie Wang, Xiao Li, Yongdong Zhang, and Feng Wu. A deep instance generative framework for milp solvers under limited data availability. In *37th Conference on Neural Information Processing Systems, NeurIPS 2023*, 2023.

Zijie Geng, Jie Wang, Xijun Li, Fangzhou Zhu, Jianye Hao, Bin Li, and Feng Wu. Differentiable integer linear programming. In *13th International Conference on Learning Representations, ICLR 2025*, 2025.

Edward I. George and Robert McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, pp. 881–889, 1993.

Ziao Guo, Yang Li, Chang Liu, Wenli Ouyang, and Junchi Yan. Acm-milp: Adaptive constraint modification via grouping and selection for hardness-preserving milp instance generation. In *41st International Conference on Machine Learning, ICML 2024*, 2024.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026.

Taoan Huang, Aaron Ferber, Arman Zharmagambetov, Yuandong Tian, and Bistra Dilkina. Contrastive predict-and-search for mixed integer linear programs. In *ICML'24: Proceedings of the 41st International Conference on Machine Learning*, 2024.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Yufei Kuang, Xijun Li, Jie Wang, Fangzhou Zhu, Meng Lu, Zhihai Wang, Jia Zeng, Houqiang Li, Yongdong Zhang, and Feng Wu. Accelerate presolve in large-scale linear programming via reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–13, 2025.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *2nd ACM(Association for Computing Machinery) Conference on Electronic Commerce*, 2000.

Hongpei Li, Hui Yuan, Han Zhang, Jianghao Lin, Dongdong Ge, Mengdi Wang, and Yinyu Ye. FMIP: Joint continuous-integer flow for mixed-integer linear programming. In *The Fourteenth International Conference on Learning Representations*, 2026.

Y. Li, Xinyan Chen, Wenxuan Guo, Xijun Li, Wanqian Luo, Jun Huang, Hui-Ling Zhen, Min jie Yuan, and Junchi Yan. Hardsatgen: Understanding the difficulty of hard sat formula generation and a strong structure-hardness-aware baseline. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

Haoyang Liu, Jie Wang, Wanbo Zhang, Zijie Geng, Yufei Kuang, Xijun Li, Yongdong Zhang, Bin Li, and Feng Wu. Milp-studio: Milp instance generation via block structure decomposition. In *NIPS '24: 38th International Conference on Neural Information Processing Systems*, 2024.

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 *13th International Conference on Learning Representations, ICLR 2025*, 2025.

Haoyang Liu, Jie Wang, Yufei Kuang, Xijun Li, Yongdong Zhang, and Feng Wu. Promoting generalization for exact combinatorial solvers via adversarial instance augmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2026.

T. J. Mitchell and J. J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988. doi: 10.1080/01621459.1988.10478694.

Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2), 2016.

Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

SCIP. The SCIP Optimization Suite 10.0. Technical report, Optimization Online, November 2025.

Oshina Vasishth and Yogita Gigras. Path planning problem. *International Journal of Computer Applications*, 104(2):17–19, 2014.

Haoyu Peter Wang, Jialin Liu, Xiaohan Chen, Xinshang Wang, Pan Li, and Wotao Yin. DIG-MILP: a deep instance generator for mixed-integer linear programming with feasibility guarantee. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Russ J. Vander Wiel and Nikolaos V. Sahinidis. Heuristic bounds and test problem generation for the time-dependent traveling salesman problem. *Transp. Sci.*, 29:167–183, 1995.

Sijia Zhang, Shuli Zeng, Shaoang Li, Feng Wu, and Xiang-Yang Li. Learning to select nodes in branch and bound with sufficient tree representation. In *13th International Conference on Learning Representations, ICLR 2025*, 2025.

Yahong Zhang, Chenchen Fan, Donghui Chen, Congrui Li, Wenli Ouyang, Mingda Zhu, and Junchi Yan. Milp-fbgen: Lp/milp instance generation with feasibility/boundedness. In *ICML'24: Proceedings of the 41st International Conference on Machine Learning*, 2024.

A Method Implementation Details

A.1 Bipartite Graph and Solution-Aware Features

An MILP instance is represented as a weighted bipartite graph $\mathcal{G} = (\mathcal{V}_c, \mathcal{V}_x, \mathcal{E})$, where $\mathcal{V}_c$ and $\mathcal{V}_x$ denote the constraint and variable vertices, respectively, and $\mathcal{E}$ refers to the edge set. The detailed features and descriptions can be found in Table 5.

In addition to the standard MILP features, we augment the graph with solution-aware statistics computed from the fixed source training pool $\mathcal{S}_{\mathcal{P}}^{\text{train}}$. These statistics provide an empirical representation of the conditioned source distribution $\pi_{\mathcal{P}}$ and collectively form H_{sol}.

For a variable j, we use

$$\mathbf{s}_j^x = \left[\mathbb{E}_{\widehat{\pi}_{\mathcal{P}}^{\text{train}}}[X_j], \text{Std}_{\widehat{\pi}_{\mathcal{P}}^{\text{train}}}(X_j), \Pr_{\widehat{\pi}_{\mathcal{P}}^{\text{train}}}(|X_j| > \epsilon_x) \right]. \tag{15}$$

For a constraint i, solution-aware statistics are computed from its normalized slack and activity,

$$s_i(\mathbf{x}) = \frac{b_i - \mathbf{a}_i^\top \mathbf{x}}{|b_i| + \|\mathbf{a}_i\|_1 + \varepsilon}, \qquad u_i(\mathbf{x}) = \exp\left(-\frac{|s_i(\mathbf{x})|}{\tau_s} \right), \tag{16}$$

using empirical moments and quantiles over $\mathcal{S}_{\mathcal{P}}^{\text{train}}$.

If a solution-aware constraint feature depends on the original parameters $(\mathbf{a}_i, b_i)$ of a masked row $i \in \mathcal{R}$, that feature is masked together with the row. Variable-level solution statistics and solution features of unmasked constraints remain observable. We denote the resulting observable solution information by $H_{\text{sol}}^{\text{obs}}$.

Table 5: Constraint, variable, and edge features used by the conditional generator.

Type	Feature	Description
C	normalized_rhs	Normalized RHS of the constraint.
	degree	Degree of the constraint vertex.
	coefficient_statistics	Statistics of incident coefficients.
	positional_encoding	Structural positional representation.
	solution_activity	Empirical statistics of $u_i(\mathbf{X})$ under the source training distribution.
	mask_indicator	Indicator of whether the row is masked.
V	objective_coefficient	Objective coefficient of the variable.
	variable_type	Binary, integer, or continuous type.
	bounds	Lower and upper variable bounds.
	degree	Degree of the variable vertex.
	solution_mean	Empirical mean of X_j.
	solution_variance	Empirical variance of X_j.
	support_frequency	Empirical probability that $\lvert X_j \rvert > \epsilon_x$.
E	coefficient	Original constraint–variable coefficient.
	normalized_coefficient	Normalized edge coefficient.
	edge_existence	Indicator of edge existence.

A.2 Solution-Aware Relation and Grouping

The relation scores in Section 4.2 are estimated from the source solution-training pool. For each candidate pair (i, j), we compute R_{ij}^{top}, R_{ij}^{sol}, and R_{ij}^{inf}, and form

$$R_{ij} = \lambda_{\text{top}} R_{ij}^{\text{top}} + \lambda_{\text{sol}} R_{ij}^{\text{sol}} + \lambda_{\text{inf}} R_{ij}^{\text{inf}}, \qquad \lambda_{\text{top}} + \lambda_{\text{sol}} + \lambda_{\text{inf}} = 1. \tag{17}$$

For scalability, the full $m \times m$ relation matrix is not materialized. Candidate constraint pairs are retained if they satisfy at least one of the following conditions: they share variables, are connected through short paths in the bipartite graph, or exhibit co-activity on the source training solutions. The complete relation score is evaluated only for retained pairs, producing a sparse weighted constraint graph.

We then apply Louvain community detection to this fixed relation graph and obtain

$$\{\mathcal{C}_1, \ldots, \mathcal{C}_G\} = \text{Louvain}(R). \tag{18}$$

The grouping stage is therefore an offline preprocessing operation. Its solution-dependent quantities are empirical statistics of $\pi_{\mathcal{P}}$ computed from $\mathcal{S}_{\mathcal{P}}^{\text{train}}$, and no gradient is propagated through the grouping procedure.

Given a target modification ratio η_{mod}, complete groups are sampled until the modification budget is reached. The resulting selected-row set is denoted by $\mathcal{R}_\eta$. Details of the resulting structural-budget error are given in Appendix A.4.

A.3 Solution-Conditioned Network Architecture

Context construction. For a selected row set $\mathcal{R}$, the generator receives

$$C_{\mathcal{R}} = \text{Enc}\left(\mathcal{P}_{\setminus\mathcal{R}}, H_{\text{sol}}^{\text{obs}}, M_{\mathcal{R}}\right). \tag{19}$$

Here, $\mathcal{P}_{\setminus\mathcal{R}}$ is the masked MILP graph, $M_{\mathcal{R}}$ identifies the selected rows, and $H_{\text{sol}}^{\text{obs}}$ contains the leakage-safe solution-aware features defined in Appendix A.1. Consequently, solution information affects the generated rows through the encoder context.

Conditional context encoder. The encoder performs bipartite message passing between constraint and variable vertices:

$$\mathbf{h}_i^{c,(l+1)} = \mathrm{MLP}_c\left(\mathbf{h}_i^{c,(l)}, \mathrm{AGG}_{j\in\mathcal{N}(i)}\,\phi_{x\to c}(\mathbf{h}_j^{x,(l)}, A_{ij})\right), \tag{20}$$

$$\mathbf{h}_j^{x,(l+1)} = \mathrm{MLP}_x\left(\mathbf{h}_j^{x,(l)}, \mathrm{AGG}_{i\in\mathcal{N}(j)}\,\phi_{c\to x}(\mathbf{h}_i^{c,(l)}, A_{ij})\right). \tag{21}$$

The initial node representations concatenate the observable structural features with the corresponding observable solution-aware features. For $i \in \mathcal{R}$, the original row parameters and any row-specific solution features derived from them are replaced by mask values before message passing.

Posterior and conditional prior. During training, the posterior encoder is

$$q_\phi(\mathbf{z}_\mathcal{R} \mid Y_\mathcal{R}, C_\mathcal{R}) = \mathcal{N}(\boldsymbol{\mu}_q, \mathrm{diag}(\boldsymbol{\sigma}_q^2)), \tag{22}$$

whereas the inference-time conditional prior is

$$p_\psi(\mathbf{z}_\mathcal{R} \mid C_\mathcal{R}) = \mathcal{N}(\boldsymbol{\mu}_p, \mathrm{diag}(\boldsymbol{\sigma}_p^2)). \tag{23}$$

Their KL divergence has the standard diagonal-Gaussian form

$$D_{\mathrm{KL}}(q\|p) = \frac{1}{2}\sum_d \left[\log\frac{\sigma_{p,d}^2}{\sigma_{q,d}^2} + \frac{\sigma_{q,d}^2 + (\mu_{q,d}-\mu_{p,d})^2}{\sigma_{p,d}^2} - 1\right]. \tag{24}$$

Sparse decoder. For each masked row i, the decoder evaluates candidate variables in

$$\mathcal{V}_i^{\mathrm{cand}} = \mathcal{N}_{\mathrm{context}}(i) \cup \mathcal{N}_{\mathrm{group}}(i) \cup \mathcal{V}_{\mathrm{sampled}}. \tag{25}$$

For candidate edge (i, j), the decoder predicts the edge probability π_{ij} and the parameters of its nonzero coefficient distribution. The resulting Spike-and-Slab likelihood is

$$p_\theta(A_{ij} \mid \mathbf{z}_\mathcal{R}, C_\mathcal{R}) = (1-\pi_{ij})\delta_0 + \pi_{ij}\mathcal{N}(\mu_{ij}^W, \sigma_W^2 s_A^2). \tag{26}$$

The practical generator objective corresponding to the ELBO is

$$\mathcal{L}_{\mathrm{gen}} = \mathcal{L}_{\mathrm{edge}} + \lambda_A\mathcal{L}_{\mathrm{coef}} + \beta_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}}, \tag{27}$$

where

$$\mathcal{L}_{\mathrm{edge}} = -\frac{1}{N_\mathcal{R}}\sum_{i\in\mathcal{R}}\sum_j \left[E_{ij}\log\pi_{ij} + (1-E_{ij})\log(1-\pi_{ij})\right], \tag{28}$$

and

$$\mathcal{L}_{\mathrm{coef}} = \frac{1}{|\mathcal{E}_\mathcal{R}|}\sum_{(i,j)\in\mathcal{E}_\mathcal{R}}\left(\frac{W_{ij}-\mu_{ij}^W}{s_A}\right)^2. \tag{29}$$

A.4 Replace ratio control

Given the constraint groups, we select the rows to modify according to an explicit structural budget. Let the source instance contain m constraints. For a target modification ratio $\eta_{\mathrm{mod}} \in [0, 1]$, the replace budget is $B_\eta = \lceil \eta_{\mathrm{mod}} m \rceil$. Because groups are atomic units, after selecting a set of complete groups $\mathcal{S}_\eta \subseteq \{1, \ldots, G\}$, the actual modified-row set is

$$\mathcal{R}_\eta = \bigcup_{g\in\mathcal{S}_\eta}\mathcal{C}_g, \qquad \widehat{\eta}_{\mathrm{mod}} = \frac{|\mathcal{R}_\eta|}{m}. \tag{30}$$

The realized modification ratio generally does not exactly equal the target ratio. We therefore define the overshoot $O_\eta = |\mathcal{R}_\eta| - B_\eta$. If the last selected group is g^*, then

$$\mathcal{R}_\eta = \mathcal{R}_\eta^- \cup \mathcal{C}_{g^*}, \qquad |\mathcal{R}_\eta^-| < B_\eta. \tag{31}$$

Hence, $0 \le O_\eta < |\mathcal{C}_{g^*}| \le C_{\max}, C_{\max} = \max_g |\mathcal{C}_g|$.

The realized modification ratio therefore satisfies

$$\frac{B_\eta}{m} \le \widehat{\eta}_{\mathrm{mod}} < \frac{B_\eta + C_{\max}}{m}. \tag{32}$$

This bound shows that the group-selection error is controlled entirely by the largest group size. To match the target modification ratio more precisely, experiments can use coverage-balanced sampling or split groups into finer sub-groups when the application permits.

The structural modification budget and the source-mass retention safeguard control different aspects of generation. The parameter η_{mod} determines how many constraint rows are regenerated, whereas ϵ_f specifies the maximum tolerated loss of source probability mass used by the safeguard in Section 4.4. The former controls structural novelty, while the latter prevents excessive destruction of the source landscape during response-profile alignment.

B Theoretical Details

B.1 Conditional ELBO and Generator Objective

For the selected rows $\mathcal{R}$, let $Y_{\mathcal{R}}$ denote the coefficient reconstruction target and let

$$C_{\mathcal{R}} = \mathrm{Enc}\left(\mathcal{P}_{\setminus\mathcal{R}}, H_{\mathrm{sol}}^{\mathrm{obs}}, M_{\mathcal{R}}\right) \tag{33}$$

be the observable context. The conditional marginal likelihood is

$$p_{\theta,\psi}(Y_{\mathcal{R}} \mid C_{\mathcal{R}}) = \int p_\theta(Y_{\mathcal{R}} \mid \mathbf{z}_{\mathcal{R}}, C_{\mathcal{R}}) p_\psi(\mathbf{z}_{\mathcal{R}} \mid C_{\mathcal{R}})\, d\mathbf{z}_{\mathcal{R}}. \tag{34}$$

For any posterior $q_\phi(\mathbf{z}_{\mathcal{R}} \mid Y_{\mathcal{R}}, C_{\mathcal{R}})$, the following exact decomposition holds:

$$\begin{aligned} \log p_{\theta,\psi}(Y_{\mathcal{R}} \mid C_{\mathcal{R}}) = \mathbb{E}_{q_\phi}\left[\log p_\theta(Y_{\mathcal{R}} \mid \mathbf{z}_{\mathcal{R}}, C_{\mathcal{R}})\right] \\ - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_{\mathcal{R}} \mid Y_{\mathcal{R}}, C_{\mathcal{R}}) \| p_\psi(\mathbf{z}_{\mathcal{R}} \mid C_{\mathcal{R}})\right) \\ + D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_{\mathcal{R}} \mid Y_{\mathcal{R}}, C_{\mathcal{R}}) \| p_{\theta,\psi}(\mathbf{z}_{\mathcal{R}} \mid Y_{\mathcal{R}}, C_{\mathcal{R}})\right). \end{aligned} \tag{35}$$

Because the last KL term is nonnegative, we obtain the conditional ELBO

$$\begin{aligned} \log p_{\theta,\psi}(Y_{\mathcal{R}} \mid C_{\mathcal{R}}) \ge \mathbb{E}_{q_\phi}\left[\log p_\theta(Y_{\mathcal{R}} \mid \mathbf{z}_{\mathcal{R}}, C_{\mathcal{R}})\right] \\ - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_{\mathcal{R}} \mid Y_{\mathcal{R}}, C_{\mathcal{R}}) \| p_\psi(\mathbf{z}_{\mathcal{R}} \mid C_{\mathcal{R}})\right). \end{aligned} \tag{36}$$

To connect this bound to the practical generator loss, consider a candidate edge (i, j) and let E_{ij} and W_{ij} denote its existence indicator and nonzero coefficient. The Spike-and-Slab likelihood factorizes as

$$p_\theta(E_{ij}, W_{ij} \mid \mathbf{z}_{\mathcal{R}}, C_{\mathcal{R}}) = (1 - \pi_{ij})^{1-E_{ij}} \pi_{ij}^{E_{ij}} \mathcal{N}\left(W_{ij}, \mu_{ij}^W, \sigma_W^2 s_A^2\right)^{E_{ij}}. \tag{37}$$

Taking the negative logarithm gives

$$\begin{aligned} -\log p_\theta(E_{ij}, W_{ij} \mid \mathbf{z}_{\mathcal{R}}, C_{\mathcal{R}}) = & -\left[E_{ij} \log \pi_{ij} + (1 - E_{ij}) \log(1 - \pi_{ij})\right] \\ & + E_{ij} \frac{(W_{ij} - \mu_{ij}^W)^2}{2\sigma_W^2 s_A^2} + \mathrm{const}. \end{aligned} \tag{38}$$

Here, const collects terms independent of the trainable model parameters. In particular, the Gaussian normalization term is

$$\frac{1}{2} E_{ij} \log(2\pi\sigma_W^2 s_A^2). \tag{39}$$

Since σ_W and s_A are treated as fixed hyperparameters, this term does not affect optimization and is absorbed into the constant.

The first term is the binary cross-entropy loss for edge existence, while the second is the normalized squared error for nonzero coefficients. Averaging these terms over the selected rows and eligible edges recovers $\mathcal{L}_{\mathrm{edge}}$ and $\mathcal{L}_{\mathrm{coef}}$ in Appendix A.3. The KL term is the diagonal-Gaussian divergence defined there. Therefore, up to the likelihood constants and optimization weights, the practical objective is

$$\mathcal{L}_{\mathrm{gen}} = \mathcal{L}_{\mathrm{edge}} + \lambda_A \mathcal{L}_{\mathrm{coef}} + \beta_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}}, \tag{40}$$

where λ_A balances coefficient reconstruction, and β_{KL} controls latent regularization.

B.2 Empirical Response-Profile Alignment

We now give the empirical construction of b_i^{prof} in Equation equation 14. For a modified row i, define the original and generated normalized responses as

$$S_i(\mathbf{x}) = \frac{b_i - \mathbf{a}_i^\top \mathbf{x}}{|b_i| + \|\mathbf{a}_i\|_1 + \varepsilon}, \qquad S_i'(\mathbf{x}; b) = \frac{b - \mathbf{a}_i'^\top \mathbf{x}}{|b| + \|\mathbf{a}_i'\|_1 + \varepsilon}. \tag{41}$$

Let

$$\widehat{\pi}_{\mathcal{P}}^{\mathrm{train}} = \sum_{k=1}^{K_{\mathrm{tr}}} w_k^{\mathrm{tr}} \delta_{\mathbf{x}^{(k)}} \tag{42}$$

be the weighted empirical source distribution on the solution-training pool, where

$$w_k^{\mathrm{tr}} = \frac{w_k}{\sum_{j:\mathbf{x}^{(j)} \in \mathcal{S}_{\mathcal{P}}^{\mathrm{train}}} w_j}. \tag{43}$$

For scalar observations $\{z^{(k)}\}_{k=1}^{K_{\mathrm{tr}}}$, define the weighted empirical quantile as

$$\widehat{Q}_\tau\left(\{z^{(k)}, w_k^{\mathrm{tr}}\}\right) = \inf\left\{ z : \sum_{k:z^{(k)} \le z} w_k^{\mathrm{tr}} \ge \tau \right\}. \tag{44}$$

For a fixed quantile grid $\mathcal{T} = \{\tau_1, \ldots, \tau_L\} \subset (0,1)$, the empirical response profiles are

$$\widehat{Q}_{i,\ell} = \widehat{Q}_{\tau_\ell}\left(\{S_i(\mathbf{x}^{(k)}), w_k^{\mathrm{tr}}\}_{k=1}^{K_{\mathrm{tr}}}\right), \tag{45}$$

and

$$\widehat{Q}'_{i,\ell}(b) = \widehat{Q}_{\tau_\ell}\left(\{S_i'(\mathbf{x}^{(k)}; b), w_k^{\mathrm{tr}}\}_{k=1}^{K_{\mathrm{tr}}}\right). \tag{46}$$

We estimate the response-profile discrepancy by

$$\widehat{D}_i(b) = \sum_{\ell=1}^{L} \omega_\ell \left| \widehat{Q}_{i,\ell} - \widehat{Q}'_{i,\ell}(b) \right|, \qquad \omega_\ell \ge 0, \qquad \sum_{\ell=1}^{L} \omega_\ell = 1. \tag{47}$$

The empirical retained source mass is

$$\widehat{M}_i(b) = \sum_{k=1}^{K_{\mathrm{tr}}} w_k^{\mathrm{tr}} \mathbb{I}\left[\mathbf{a}_i'^\top \mathbf{x}^{(k)} \le b\right]. \tag{48}$$

The provisional profile-aligned RHS is obtained from

$$\begin{aligned} b_i^{\mathrm{prof}} = \arg\min_{b \in \mathcal{B}_i} \quad & \widehat{D}_i(b) \\ \text{s.t.} \quad & \widehat{M}_i(b) \ge 1 - \epsilon_f, \end{aligned} \tag{49}$$

where $\mathcal{B}_i$ is a bounded one-dimensional candidate interval constructed from the empirical projections $\{\mathbf{a}_i'^\top \mathbf{x}^{(k)}\}_{k=1}^{K_{\mathrm{tr}}}$ and the numerical scale of the source row. This scalar problem can be solved using a finite candidate grid or a one-dimensional search procedure.

The empirical construction uses only $\mathcal{S}_{\mathcal{P}}^{\mathrm{train}}$. The resulting b_i^{prof} is passed to Appendix B.3 as the provisional boundary $\bar{b}_i'$.

B.3 Joint Retention Safeguard

Let

$$\bar{b}_i' = b_i^{\mathrm{prof}} \tag{50}$$

be the profile-aligned provisional RHS obtained in Appendix B.2 without using the calibration pool. Before accessing the calibration solutions, we fix A', $\bar{\mathbf{b}}'$, the selected rows $\mathcal{R}_\eta$, and all generation randomness.

Table 6: Dataset statistics. # of Data refers to the number of instances in training set, and # of Variable and Constraint means the average number of variables and constraints, respectively.

Dataset	#Data	#Variable	#Constraint	Variable Types
CA	400	1500	578	binary
IP	300	1083	195	binary/continuous
IS	300	1500	5942	binary
MIK	90	387	312	integer/continuous

For each modified row, define the scale using only the solution-training pool:

$$\tau_i = \mathrm{MAD}\left\{\mathbf{a}_i'^{\top}\mathbf{x} : \mathbf{x} \in \mathcal{S}_{\mathcal{P}}^{\mathrm{train}}\right\} + \tau_{\min}. \tag{51}$$

Given independent calibration solutions $\mathbf{X}_1, \ldots, \mathbf{X}_K$, define the joint nonconformity score

$$r_k = \max_{i \in \mathcal{R}_\eta} \frac{\mathbf{a}_i'^{\top}\mathbf{X}_k - \bar{b}_i'}{\tau_i}. \tag{52}$$

Let

$$k^* = \lceil (K+1)(1-\epsilon_f) \rceil, \tag{53}$$

and assume $k^* \le K$. Define

$$q_{\epsilon_f} = r_{(k^*)}, \tag{54}$$

where $r_{(k^*)}$ is the k^*-th ordered score. We use the one-sided correction

$$q_{\epsilon_f}^+ = \max\{0, q_{\epsilon_f}\}, \tag{55}$$

and set

$$b_i' = \begin{cases} \bar{b}_i' + q_{\epsilon_f}^+ \tau_i, & i \in \mathcal{R}_\eta, \\ b_i, & i \notin \mathcal{R}_\eta. \end{cases} \tag{56}$$

The nonnegative correction is deliberately one-sided. Conformal calibration may relax a profile-aligned boundary when required by the retention safeguard, but it does not tighten an already sufficiently conservative boundary.

Assume that $\mathbf{X}_1, \ldots, \mathbf{X}_K, \mathbf{X}_{K+1}$ are exchangeable under the conditioned source distribution $\pi_{\mathcal{P}}$, and that A', $\bar{\mathbf{b}}'$, $\mathcal{R}_\eta$, and all quantities used to construct them are independent of the calibration samples. Define

$$r_{K+1} = \max_{i \in \mathcal{R}_\eta} \frac{\mathbf{a}_i'^{\top}\mathbf{X}_{K+1} - \bar{b}_i'}{\tau_i}. \tag{57}$$

The standard split-conformal rank argument gives

$$\Pr\left[r_{K+1} \le q_{\epsilon_f}\right] \ge 1 - \epsilon_f. \tag{58}$$

If $q_{\epsilon_f} \ge 0$, the final boundary coincides with the standard conformal correction. If $q_{\epsilon_f} < 0$, replacing it by zero only relaxes the corresponding constraints relative to the conformal boundary. Therefore,

$$\Pr_{\mathbf{X}_{K+1} \sim \pi_{\mathcal{P}}}\left[A'\mathbf{X}_{K+1} \le \mathbf{b}'\right] \ge 1 - \epsilon_f. \tag{59}$$

C Experimental Reproduction Details

C.1 Dataset Statistics.

As described in the main paper, we conduct experiments on four datasets, including MIK, CA, IP, IS. We demonstrate the detailed information of these datasets in Table 6.

C.2 Solution Collection

For each source instance, we construct a finite source solution pool

$$\mathcal{S}_{\mathcal{P}} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(K)}\} \subseteq \mathcal{F}(\mathcal{P}) \tag{60}$$

using solver-assisted sampling and local neighborhood search. Since the LNS variable-selection and perturbed-objective procedures are presented in the main methodology, we focus here on the finite empirical representation and the associated data protocol.

To prevent near-duplicate solutions, let $\mathcal{B} \subseteq \{1, \ldots, n\}$ denote the set of binary-variable indices. For two solutions $\mathbf{x}$ and $\mathbf{y}$, we define the distance

$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) &= \sum_{j \notin \mathcal{B}} |x_j - y_j| + d_H(\mathbf{x}_{\mathcal{B}}, \mathbf{y}_{\mathcal{B}}), \\ d_H(\mathbf{x}_{\mathcal{B}}, \mathbf{y}_{\mathcal{B}}) &= \sum_{j \in \mathcal{B}:\, y_j = 0} x_j + \sum_{j \in \mathcal{B}:\, y_j = 1} (1 - x_j). \end{aligned} \tag{61}$$

Thus, binary variables are compared by the Hamming distance, whereas integer and continuous variables are compared by the absolute difference. For a candidate solution $\mathbf{x}$, the minimum distance to the current pool is

$$d_{\min}(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{S}_{\mathcal{P}}} d(\mathbf{x}, \mathbf{y}). \tag{62}$$

The candidate is inserted into $\mathcal{S}_{\mathcal{P}}$ only when $d_{\min}(\mathbf{x}) \geq \delta_d$, otherwise it is rejected or resampled up to a pre-specified number of attempts.

The induced weighted empirical distribution is

$$\widehat{\pi}_{\mathcal{P}} = \sum_{k=1}^{K} w_k \delta_{\mathbf{x}^{(k)}}, \qquad w_k = \frac{\sum_{r=1}^{R} \frac{\alpha_r}{Z_{\mathcal{P}}^{\beta_r}} \exp[-\beta_r \tilde{f}_{\mathcal{P}}(\mathbf{x}^{(k)})]}{\sum_{k'=1}^{K} \sum_{r=1}^{R} \frac{\alpha_r}{Z_{\mathcal{P}}^{\beta_r}} \exp[-\beta_r \tilde{f}_{\mathcal{P}}(\mathbf{x}^{(k')})]}. \tag{63}$$

Here, the factors $Z_{\mathcal{P}}^{\beta_r}$ are retained because the normalization constants of different temperature components are generally unequal and cannot be cancelled across the mixture. We treat the constructed pool as a finite support whose retained solutions are representative of $\nu_{\mathcal{P}}$, so common base-density factors cancel under normalization.

For each source instance, the resulting solution pool is partitioned into

$$\mathcal{S}_{\mathcal{P}} = \mathcal{S}_{\mathcal{P}}^{\text{train}} \dot{\cup} \mathcal{S}_{\mathcal{P}}^{\text{cal}} \dot{\cup} \mathcal{S}_{\mathcal{P}}^{\text{test}}. \tag{64}$$

The training subset is used to construct solution-aware constraint relations in Section 4.2, solution-conditioned features in Section 4.3, and empirical response profiles and provisional RHS values in Section 4.4. For each generated instance $\mathcal{P}'$, an independent test solution pool is constructed using the same solution-collection protocol, multi-temperature parameters, solver configuration, and evaluation budget as those used for the corresponding source instance.

C.3 Baseline Implementations

For G2MILP and ACM-MILP, we reproduce the methods using the publicly available code repositories provided by the corresponding studies. Data preprocessing and MILP solving are performed on a local machine equipped with an Intel Core Ultra 7 258V processor and an Intel Arc 140V GPU (16 GB), with COPT 8.0.5 used as the optimization solver. Neural network training is conducted on a remote server running Ubuntu 22.04.5 LTS and equipped with eight NVIDIA L40 GPUs. Our proposed SLP-MILP is evaluated under the same computational environment and software configuration, and its implementation will be made publicly available on GitHub. To the best of our knowledge, MILP-FBGen does not currently provide a publicly available official code repository. We therefore reimplement MILP-FBGen based on the methodological descriptions and pseudocode presented in the original paper, together with the two related studies referenced for its implementation. To facilitate reproducibility, our reimplementation of method MILP-FBGen will also be made publicly available.

C.4 METRIC CALCULATION DETAILS

The evaluation metrics of the paper mainly use the graph structure metrics mentioned by Geng et al. (2023), as well as the eight metrics proposed in this paper for evaluating solution structure similarity. Among them, the graph structure metrics are obtained by comparing source instances with generated instances, while the solution structure metrics are evaluated by comparing source solution sets with generated solution sets. When there are multiple generated instances, the metric values are calculated separately for each instance and then averaged over instances with the same source and method.

The detailed information of the proposed indicators is as follows:

Chamfer distance. For each source solution, we find the nearest generated solution, for each generated solution, we find the nearest source solution. Chamfer distance is the average of the two directed mean nearest-neighbor distances:

$$\text{Chamfer} = \frac{1}{2}\left(\frac{1}{|X|}\sum_{\mathbf{x}\in X}\min_{\mathbf{y}\in Y}\|\mathbf{x}-\mathbf{y}\|_1 + \frac{1}{|Y|}\sum_{\mathbf{y}\in Y}\min_{\mathbf{x}\in X}\|\mathbf{y}-\mathbf{x}\|_1\right). \tag{65}$$

Symmetrized nearest-neighbor distance (SymNN). This is the larger of the two directed mean nearest-neighbor distances and is stricter than Chamfer distance.

$$\text{SymNN} = \max\left(d_{X\to Y}, d_{Y\to X}\right). \tag{66}$$

Support Jaccard. For binary and integer variables, define the support set $S(\mathbf{x}) = \{j : |x_j| > \tau_{\text{supp}}\}$, and compute

$$J(S(\mathbf{x}), S(\mathbf{y})) = \frac{|S(\mathbf{x}) \cap S(\mathbf{y})|}{|S(\mathbf{x}) \cup S(\mathbf{y})|}. \tag{67}$$

We apply bidirectional best matching between source and generated solutions and average the resulting similarities.

Active constraint Jaccard. For each solution, define the active constraint set $\text{A}(\mathbf{x}) = \{i : m_i(\mathbf{x}) \leq \epsilon_{\text{active}}\}$, where $m_i(\mathbf{x})$ is the normalized boundary distance. We compute the Jaccard similarity between source and generated active sets, apply bidirectional best matching, and average the scores.

Maximum Mean Discrepancy (MMD). We use the RBF kernel

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right) \tag{68}$$

to compute the maximum mean discrepancy between the source and generated solution distributions.

Sliced Wasserstein Distance (SWD). For each solution, we compute a normalized slack vector and calculate the sliced Wasserstein distance between the source and generated slack distributions over random unit projections:

$$\text{SWD} = \frac{1}{P}\sum_{p=1}^{P} W_1\left(\mathbf{u}_p^\top S_{\text{src}}, \mathbf{u}_p^\top S_{\text{gen}}\right). \tag{69}$$

Correlation shift. We compute the variable correlation matrices of the source and generated solution pools and compare their off-diagonal entries:

$$\text{CorrShift} = \frac{1}{n(n-1)}\sum_{i\neq j}|C_{\text{src},ij} - C_{\text{gen},ij}|. \tag{70}$$

Objective shift. We evaluate source solutions with the source objective and generated solutions with the generated objective, compare the two objective-value distributions by one-dimensional Wasserstein distance, and normalize by the source objective standard deviation.

C.5 More Supplementary Experiments

Variable fixing task. To evaluate whether the solution-landscape information preserved by generated instances can benefit downstream optimization, we design a variable-fixing task. The objective is to exploit solution patterns from generated instances to infer high-confidence variable assignments and transfer these assignments back to the corresponding source MILPs.

For each source instance $\mathcal{P}$, we first solve the generated instances produced by different generation methods and collect their feasible solution pools. Since near-optimal solutions contain more informative variable patterns, we select solutions with objective values close to the best solution found in the generated solution pool. Specifically, a feasible solution x is selected when

$$\frac{|f(x) - f^\star|}{|f^\star| + \epsilon} \leq \delta, \tag{71}$$

where $f^\star$ denotes the best objective value among the collected solutions, and δ is the near-optimality tolerance.

For each binary variable x_j, we estimate its value preference from the selected near-optimal solutions. Let $\mathcal{S}$ denote the selected solution set. The probability of assigning $x_j = 1$ is calculated as

$$p_j = \frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} x_j. \tag{72}$$

A variable is fixed only when its dominant assignment probability exceeds a confidence threshold ρ:

$$\max(p_j, 1 - p_j) \geq \rho. \tag{73}$$

In all experiments, we set $\rho = 0.70$. The selected assignments are then transferred to the corresponding source MILP by adding equality constraints for the fixed variables.

After variable fixing, we solve the reduced source MILP using the same solver configuration as the original problem. We report the solving time and final objective value to characterize the trade-off between computational efficiency and solution quality after applying variable fixing.

All compared generation methods use the same source instances, generated instance budget, solution collection procedure, solver version, time limit, and thread configuration. The reported results are averaged over all evaluated source instances.