# UNIFY ML4MILP: DISENTANGLE, RECOMBINE, AND REEVALUATE TO BENCHMARK NEURAL PRIMAL HEURISTICS WITH SOLVER-BACKED SEARCH

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Machine learning for (mixed) integer linear programming (ML4MILP) is rapidly expanding from solver-internal assistance toward end-to-end primal solution generation. Yet progress remains difficult to compare and interpret: neural prediction, solver-backed completion, and incumbent refinement are often entangled within monolithic pipelines, while datasets, training styles, solver backends, and evaluation protocols vary substantially across studies. We introduce ML4MILP-Bench, a modular benchmark and systematic analysis framework that decomposes learning-based MILP primal heuristics into three reusable stages. A *Learner* maps a standardized bipartite graph to variable-wise soft scores; an *Enforcer* converts these predictions into a first complete solution; and a *Refiner* further improves a feasible incumbent under fixed budgets. Within the unified taxonomy and re-implementation, we streamline, recombine, and systematically probe 8 learners, 6 enforcers, and 3 refiners via re-modularized learning components and re-encapsulated solver backends. Across experiments on diverse settings, we separately assess raw neural predictions and their interactions with solver-backed components to discern where performance gains originate and which design principles remain effective under controlled ablation. We further envision ML4MILP-Bench as a living benchmark to maintain up-to-date reproductions and empirically grounded insights.

## 1 BACKGROUND AND INTRODUCTION

Neural combinatorial optimization (NCO) (Bengio et al., 2021; Kwon et al., 2020; Qiu et al., 2022; Luo et al., 2023) has achieved remarkable progress on canonical graph-structured problems (Li et al., 2024; Drakulic et al., 2025; Pan et al., 2025; Li et al., 2026b). However, these advances have not yet fully transferred to mixed-integer linear programming (MILP), a more general and practically dominant modeling form for real-world optimization. In practice, expert-engineered solvers like Gurobi, SCIP, CPLEX, and COPT, etc., remain the credible tools. Accordingly, early ML4MILP research mainly sought to augment individual solver components, e.g., learning to branch (Gasse et al., 2019; Zhang et al., 2024), presolve (Liu et al., 2024), select the cutting planes (Huang et al., 2022), or initialize the simplex basis (Lu & Yan, 2025). Though effective, they stay confined to assisting or accelerating solver-internal procedures rather than directly completing the primal solutions.

More recently, especially inspired by one-shot or generative solvers for graph-based NCO (Joshi et al., 2019; Fu et al., 2021; Sun & Yang, 2023; Li et al., 2023; Sanokowski et al., 2024; Ma et al., 2025a; Pan et al., 2026), increasing attention has shifted toward (near) end-to-end neural primal heuristics for ILP and MILP (Zeng et al., 2024; Geng et al., 2025; Li et al., 2026a; Lee & Kim, 2026). They commonly employ a neural module to predict variable-wise soft scores (logits or marginal probabilities, a.k.a. heatmaps). Even when end-to-end feasibility is not explicitly targeted (Han et al., 2023; Yuan et al., 2025a; Pu et al., 2026; Huang et al., 2026a), the predicted scores are typically converted into complete assignments through simple rounding, sampling, or solver-supported post-inference search.

Despite growing methodological diversity, current ML4MILP methods remain difficult to organize, reproduce, and fairly evaluate. Existing studies often vary simultaneously in data generation, supervision, model architecture, decoding, solver backends, computational budgets, and evaluation protocols. As a result, nominally method-level comparisons rarely isolate the contributions of individual components. The principal limitations in this regard can be summarized as follows.



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.

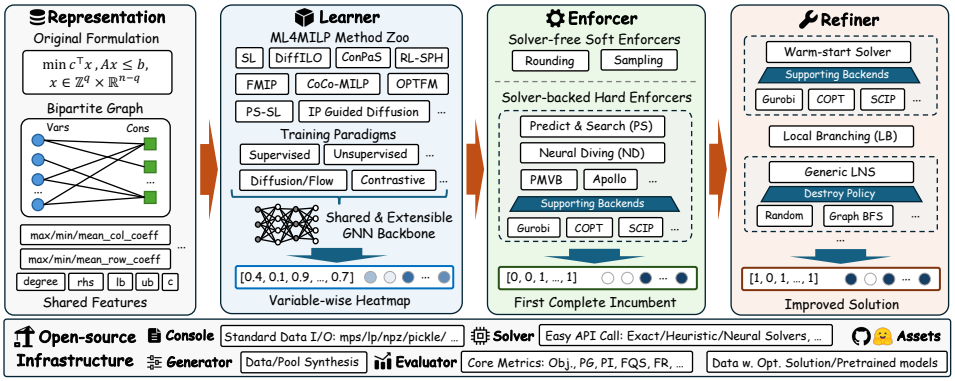



Figure 1: Overview of the ML4MILP-Bench framework. We offer minimal code examples in Appendix J.

**1) Ambiguous Method Taxonomy and Entangled Pipeline Semantics.** The literature lacks a consistent definition of a learning-based MILP "method". For example, Predict-and-Search (Han et al., 2023) and Neural Diving (Nair et al., 2021) are sometimes treated as standalone learning-based baselines (Zeng et al., 2024), and sometimes as neural predictors coupled with an exact solver, e.g., `PS+Gurobi` (Geng et al., 2025), whereas other works describe them as "downstream solvers" that consume learned variable scores and invoke exact optimization (Li et al., 2026a). Composite labels like `WS-Ours+Gurobi` (Liu et al., 2025) and `CoCo-MILP+Apollo-MILP+Gurobi` (Pu et al., 2026) further conflate orthogonal elements, making the nomenclature inconsistent across papers. **2) Fragmented Implementations and Non-standardized Data Resources.** Most studies generate task-specific synthetic instances with different scales and problem parameters, even for nominally identical problem classes (Gasse et al., 2019; Ye et al., 2023; Zeng et al., 2024; Geng et al., 2025). Further, their solver-generated reference solutions are often unavailable or only partially released (Han et al., 2023; Huang et al., 2024a; Yuan et al., 2025a; Li et al., 2026a). Reproducing a baseline requires adapting to its data format, re-solving its instances, and rebuilding its preprocessing pipeline, thereby hindering the accumulation of reusable components and standardized experimental assets. **3) Irregular Evaluation Protocols. i) Obscured Raw Learning Capability.** Neural primal heuristics seek to learn informative solution distributions or variable-wise scores, yet raw outputs are rarely evaluated before solver intervention, except in a few recent studies (Zeng et al., 2024; Geng et al., 2025; Huang et al., 2026b). Accordingly, excessively emphasizing marginal gains or speedups relative to exact solvers alone provides limited evidence about the contribution of the learning component. **ii) Inconsistent Metrics.** Existing works report metrics, e.g., primal gaps, under incompatible definitions (further demonstrated Table 5). Infeasible outputs are also discarded or penalized differently, biasing the comparison. **iii) Outdated Solver Backends.** Existing studies often rely on stale solver versions, which may overstate the incremental value of learned guidance. E.g., Yuan et al. (2025b); Huang et al. (2024a) still resort to Gurobi 9.x/10.x for the best known solutions (BKS), despite substantial improvements in later releases (further discussed around Table 6).

Notably, related efforts (detailed in Appendix A) have addressed analogous issues in relevant learning-to-optimize domains through unified benchmarking: RL4CO (Berto et al., 2023) standardizes RL-based NCO implementations and evaluation, ML4TSPBench (Li et al., 2025) and ML4CO-Bench-101 (Ma et al., 2025b) further disentangle methodological components to identify their genuine contributions and derive design principles for TSP and broader graph-based CO problems, while facilitating reproducible research through open-source toolkits. Building on this line of work, we adpot the same philosophy and develop ML4MILP-Bench, a modular benchmark decomposing learning-based MILP solving into unified stages: **1) Learner**, which maps a unified bipartite representation to variable-wise soft scores while encapsulating method-specific objectives and training paradigms; **2) Enforcer**, which converts these scores into a complete candidate through solver-free rounding or sampling, or into a guaranteed feasible incumbent through solver-backed fixing or completion; and **3) Refiner**, which improves a feasible incumbent under a controlled budget. Our main contributions are:

❶ **Streamlined Framework under a Unified Taxonomy.** ML4MILP-Bench formalizes and re-implements the learner-enforcer-refiner decomposition. The separation of variable-level solution learning, feasibility enforcement, and incumbent refinement enables monolithic pipelines to be consistently categorized, independently configured, systematically recombined, and fairly ablated.

❷ **Solver-free Feasibility-first Quality-aware Evaluation.** We report multiple complementary metrics, including the proposed Feasible Quality Score (FQS), that prioritize primal feasibility

while accounting for solution quality and efficiency. This enables direct assessment of how well existing learners produce feasible and high-quality solutions without relying on external solvers.

❸ **Systematic Investigation with Insights via Stage-wise Recombination.** Under shared data contracts, solver settings, and, where applicable, unified neural backbones, we conduct stage-wise comparisons to determine when learned variable-wise scores genuinely improve solver-backed enforcement and refinement, and which design choices remain effective under controlled ablations.

❹ **Open Toolkit and Modular Infrastructure.** Beyond method replication, ML4MILP-Bench standardizes lower-level engineering components, including MILP data I/O, customized instance generation, label and solution-pool construction, one-command interfaces to major exact solvers, and shared base classes for learning, enforcement, refinement, and evaluation. We will release all training and test instances, solver-generated reference solutions, and unified pretrained checkpoints.

## 2 A Modular Design Space for Learning-Based Solving Pipelines

**Overview.** As illustrated in Fig. 1, ML4MILP-Bench consists of a unified representation followed by three stages: Learner, Enforcer, and Refiner, which is organized around a central research question: *how should learning-based MILP solvers be categorized, decomposed, and evaluated so that the contribution of each component can be transparently understood and compared?*

### 2.1 Unified Representation and Data Contract

**Mixed Integer Linear Programming.** A mixed-integer linear program (MILP) optimizes a linear objective over linear constraints with a subset of variables constrained to be integral:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \ \mathbf{c}^\top \mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \le \mathbf{b}, \quad \mathbf{x} \in \mathbb{Z}^q \times \mathbb{R}^{n-q}, \quad \ell \le \mathbf{x} \le u, \tag{1}$$

where $n$ and $m$ are the numbers of decision variables and constraints, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$, $\ell, u \in (\mathbb{R} \cup \{\pm\infty\})^n$, and the first $q$ variables are discrete. Equality and $\ge$ constraints can be converted to this form by standard linear transformations, and maximization problems can be handled by negating the objective. **Integer linear programming (ILP)** is the pure-integer case where all decision variables are integers. A particularly widely studied subclass is **binary ILP**, where $\mathbf{x}_i \in \{0, 1\}$ for all integer variables. We denote the feasible set of an MILP instance $\mathcal{I}$ by $\mathcal{X}(\mathcal{I})$ and its objective function by $f_{\mathcal{I}}(\mathbf{x}) = \mathbf{c}^\top \mathbf{x}$. This work focuses primarily on the *binary ILP* instances.

**Variable-Constraint Bipartite Representation.** By convention (Ding et al., 2020), we map each MILP instance to a variable-constraint bipartite graph $G_{\mathcal{I}}$. We denote this mapping as a deterministic representation operator $\Phi$, such that $G_{\mathcal{I}} = \Phi(\mathcal{I}) = (\mathcal{V}, \mathcal{C}, \mathcal{E}, \mathbf{X}^v, \mathbf{X}^c, \mathbf{X}^e)$. Here, $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{C} = \{c_1, \ldots, c_m\}$ are the variable and constraint node sets, respectively, and $(c_i, v_j) \in \mathcal{E}$ if and only if $\mathbf{A}_{ij} \neq 0$. The matrices $\mathbf{X}^v \in \mathbb{R}^{n\times d_v}$, $\mathbf{X}^c \in \mathbb{R}^{m\times d_c}$, and $\mathbf{X}^e \in \mathbb{R}^{|\mathcal{E}|\times d_e}$ contain variable, constraint, and edge features, where $d_v$, $d_c$, and $d_e$ denote their feature dimensions (detailed in Appendix B). We align this representation across learners to reduce feature-induced variation.

**Unified Supervision Assets.** Different learning paradigms require different forms of training information. For instance $\mathcal{I}_i$, we denote the available supervision by $\mathcal{Y}_i$. It may be empty for unsupervised learning, a single reference solution $\mathbf{x}_i^{\text{ref}}$, a feasible solution pool $\mathcal{P}_i^+$, or paired positive and negative pools $(\mathcal{P}_i^+, \mathcal{P}_i^-)$. A benchmark dataset can therefore be represented as $\mathcal{D} = (\mathcal{I}_i, \mathcal{Y}_i)_{i=1}^N$, where $N$ is the number of instances. This contract accommodates single-solution supervision, solution-pool supervision (Han et al., 2023), contrastive learning (Huang et al., 2024a), and unsupervised learning (Geng et al., 2025; Huang et al., 2026a) without method-specific data formats. At the infrastructure level, we further standardize LP/MPS file parsing, instance generation, reference solving, solution-pool construction, graph conversion, and serialization under a common coding interface.

### 2.2 Learner-Enforcer-Refiner Decomposition

For an instance $\mathcal{I}$, we represent the complete learning-based solving workflow as

$$G_{\mathcal{I}} \xrightarrow{\mathsf{L}_\theta} \mathbf{s} \xrightarrow{\mathsf{E}_\alpha} \left(\widetilde{\mathbf{x}}^0, \delta^0\right) \xrightarrow{\mathsf{R}_\beta} \widetilde{\mathbf{x}}. \tag{2}$$

Here, $\mathsf{L}_\theta$, $\mathsf{E}_\alpha$, and $\mathsf{R}_\beta$ denote the Learner, Enforcer, and Refiner with method-specific (learnable or hyper) parameters $\theta$, $\alpha$, and $\beta$, respectively. The vector $\mathbf{s} \in \mathbb{R}^q$ represents the (integer) variable-wise heatmap as neural guidance, $\widetilde{\mathbf{x}}^0 \in \{0, 1\}^q \times \mathbb{R}^{n-q}$ is the initial complete assignment[1] returned by

[1] In the following, we focus on binary ILPs, for which $n = q$, as with most existing Learners that predict only binary variables, while for MILPs, the $n - q$ continuous variables are completed by the backend solver.

Table 1: General categorization and core features of recent ML4MILP methods. ✓: officially available; ×: unavailable; –: not applicable. *Partially released. †Only auxiliary learner-level ranking or score diagnostics are reported. Raw L denotes metrics on learner outputs without solver-backed enforcement; L+X denotes results after any post-processing, e.g., a solver-backed enforcer or refiner.

| Method | Major Role | Training Type (Feature) | Task | Availability | | | Report Level | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Code | Data | Weight | Raw L | L+X | Metrics |
| Gurobi (Gurobi Optimization, 2023) | Solver | – | LP/QP/MILP | × | – | – | – | – | Any |
| SCIP (Maher et al., 2016) | Solver | – | LP/MILP/MINLP | ✓ | – | – | – | – | Any |
| COPT (Ge et al., 2026) | Solver | – | LP/QP/MILP | × | – | – | – | – | Any |
| CPLEX (Studio, 2020) | Solver | – | LP/QP/MILP | × | – | – | – | – | Any |
| Base-SL | Learner | SL (BCE/MSE) | Binary ILP | – | – | – | ✓ | ✓ | Obj./Gap/Feas./Viol./Time |
| PS-SL (Han et al., 2023) | Learner | SL (pool-weighted BCE) | MILP | ✓ | × | ✓ | × | ✓ | Obj./Gap/Time |
| ConPaS (Huang et al., 2024a) | Learner | SL (contrastive) | Binary MILP | × | × | × | × | ✓ | Obj./Gap/Time |
| Guided DDIM (Zeng et al., 2024) | Learner | SL (diffusion+guidance) | Binary IP | ✓ | × | × | ✓ | ✓ | Obj./Gap/Feas./Time |
| OPTFM (Yuan et al., 2025a) | Learner | SSL (contrastive)+SL (GBDT) | MILP | ✓ | ✓ | × | × | ✓ | Obj./Gap/PI/Improve. |
| DiffILO (Geng et al., 2025) | Learner | UL (obj+violation guidance) | Binary ILP | ✓ | ✓ | × | ✓ | × | Obj./Gap/Feas./Viol./Time |
| FMIP (Li et al., 2026a) | Learner | SL (flow matching) | MILP | ✓ | × | ✓ | × | ✓ | Obj./Gap/Feas./Time |
| CoCo-MILP (Pu et al., 2026) | Learner | SL (contrast+ranking) | MILP | ✓ | ✓ | × | ✓† | ✓ | Rank/Obj./Gap/Time |
| ND (Nair et al., 2021) | Enforcer | (energy+selective) | MILP | × | × | × | × | ✓ | Gap/PDGap/Time |
| PS (Han et al., 2023) | Enforcer | (trust region) | MILP | ✓ | × | ✓ | × | ✓ | Obj./Gap/Time |
| Apollo-MILP (Liu et al., 2025) | Enforcer | (predict-correct) | MILP | ✓ | × | ✓* | × | ✓ | Obj./Gap/Time |
| PMVB (Chen et al., 2026) | Enforcer | (multi-var branching) | Binary MIP | ✓ | ✓ | – | × | ✓ | Obj./Gap/Time/Nodes |
| Warm-start Solver | Refiner | – | MILP | – | – | – | – | ✓ | Obj./Gap/PI/Time |
| LB (Fischetti & Lodi, 2003) | Refiner | (Hamming ball) | Binary MIP | ✓ | – | – | – | ✓ | Obj./Gap/PI/Time |
| Generic LNS (Song et al., 2020) | Refiner | (adaptable to destroy policies) | Binary ILP | ✓ | ✓* | ✓* | × | ✓ | Gap/PI/Surv./Time |

the Enforcer, and $\delta^0 \in \{0,1\}$ indicates whether this assignment is feasible. The Refiner is invoked only when $\delta^0 = 1$, and typically it iteratively improves $\widetilde{\mathbf{x}}^0$ to the final solution $\widetilde{\mathbf{x}}$. Exact solvers are treated as explicit backends rather than inseparable parts of method names. We denote the optional solver backends of the Enforcer and Refiner by $S_E$ and $S_R$ (an empty backend $S = \texttt{None}$ indicates a solver-free component), and their computational budgets by $B_E$ and $B_R$, respectively. A complete pipeline configuration $\Pi$ can thus be formulated as $\Pi = \langle \Phi, \mathsf{L}_\theta, \mathsf{E}_\alpha[S_E, B_E], \mathsf{R}_\beta[S_R, B_R] \rangle$ .

### 2.3 Learners: Learning Variable-wise Primal Guidance

The Learner maps the standardized graph representation to variable-wise soft guidance: $\mathsf{L}_\theta : G_{\mathcal{I}} \mapsto \mathbf{s} \in \mathbb{R}^n$. Depending on the method, its entries may represent logits, marginal probabilities, variable priorities, or other prediction scores. For binary variables, scores are commonly transformed into probabilities $\mathbf{p} = (p_1, \ldots, p_n) \in [0,1]^n$. Methods may internally maintain richer representations, latent trajectories, or mixed-variable predictions. Our benchmark only requires a standardized variable-wise interface that can be consumed by downstream stages. Formally, given a training set $\mathcal{D}$, a generic learner can be expressed as $\theta^\star = \arg\min_\theta \sum_{(\mathcal{I}_i, \mathcal{Y}_i) \in \mathcal{D}} \mathcal{L}_m \left( \mathsf{L}_\theta(G_{\mathcal{I}_i}), \mathcal{Y}_i; \mathcal{I}_i \right)$, where $\mathcal{L}_m$ denotes the method-specific learning objective. This abstraction keeps supervision, loss design, and training paradigm internal to the Learner while exposing a common prediction interface.

**Supervised Reference Solution Learning. i) Individual Solution Proximity.** Let $\mathbf{p}_\theta \in [0,1]^q$ denote the predicted probabilities of the $q$ integer variables. In the simplest paradigm, vanilla supervised learning, the target is a reference assignment $\mathbf{x}^{\mathrm{ref}}$. **ii) Pooled Solution Proximity.** PS (Han et al., 2023) instead approximates a solution distribution from a pool $\mathcal{P}^+ = \{\mathbf{x}^{(r)}\}_{r=1}^R$ of high-quality feasible solutions and derives objective-weighted marginal targets $\mathbf{x}^{\mathrm{ref}} := \sum_{r=1}^R w_r \mathbf{x}^{(r)}$ with $w_r = \dfrac{\exp(-\mathbf{c}^\top \mathbf{x}^{(r)})}{\sum_{s=1}^R \exp(-\mathbf{c}^\top \mathbf{x}^{(s)})}$. Both can be trained through variable-wise binary cross-entropy (BCE), i.e., $\mathcal{L}_{\mathrm{SL}}(\theta) = -\frac{1}{q} \sum_{j=1}^q \left[ \mathbf{x}_j^{\mathrm{ref}} \log \mathbf{p}_{\theta,j} + (1 - \mathbf{x}_j^{\mathrm{ref}}) \log(1 - \mathbf{p}_{\theta,j}) \right]$. Thus, this family differs mainly in how reference solutions are converted into marginal supervision.

**Supervised Contrastive Solution Learning. i) Sample-level Contrast.** Rather than fitting each variable independently, ConPaS (Huang et al., 2024a) contrasts high-quality feasible solutions $\mathcal{P}^+$ against low-quality or infeasible solutions $\mathcal{P}^-$. With $\mathbf{p}_\theta$ denoting the predicted variable marginals, its quality-aware InfoNCE objective (van den Oord et al., 2019) takes the form

$$\mathcal{L}_{\mathrm{ConPaS}}(\theta) = -\frac{1}{|\mathcal{P}^+|} \sum_{\mathbf{x}^+ \in \mathcal{P}^+} \log \frac{\exp\left( (\mathbf{x}^+)^\top \mathbf{p}_\theta / \tau(\mathbf{x}^+) \right)}{\sum_{\mathbf{x}' \in \mathcal{P}^- \cup \{\mathbf{x}^+\}} \exp\left( (\mathbf{x}')^\top \mathbf{p}_\theta / \tau(\mathbf{x}') \right)}, \tag{3}$$

where $\tau(\mathbf{x}) > 0$ is a quality-dependent temperature, and $\tau^+ = \tau^- = 1$ are adopted in the benchmark runs following the original paper. **ii) Variable-level Contrast.** CoCo-MILP (Pu et al., 2026) instead contrasts variables within each reference solution. Let $\mathcal{V}^+ = \{j : \mathbf{x}_j^{\mathrm{ref}} = 1\}$ and $\mathcal{V}^- = \{j : \mathbf{x}_j^{\mathrm{ref}} = 0\}$, and let $\mathbf{s}_\theta$ be the vector of model-predicted variable logits. Its inter-variable contrastive loss

(VCL) combines global multi-sample contrast and pairwise ranking:

$$\mathcal{L}_{\mathrm{CoCo}}(\theta) = -\log \frac{\sum_{i\in\mathcal{V}^+} \exp(\mathbf{s}_{\theta,i}/\tau)}{\sum_{k\in\mathcal{V}^+\cup\mathcal{V}^-} \exp(\mathbf{s}_{\theta,k}/\tau)} + \frac{\lambda_{\mathrm{rank}}}{|\mathcal{V}^+||\mathcal{V}^-|} \sum_{i\in\mathcal{V}^+} \sum_{j\in\mathcal{V}^-} \max\left(0, \gamma - (\mathbf{s}_{\theta,i} - \mathbf{s}_{\theta,j})\right), \quad (4)$$

where $\tau > 0$ is the temperature, $\gamma > 0$ the ranking margin, and $\lambda_{\mathrm{rank}} \geq 0$ balances the two terms. Thus, both methods move beyond pointwise accuracy toward relative solution or variable preference.

**Supervised Generative Distribution Learning. i) Diffusion-based Generation.** Generative learners model distributions over complete candidate solutions instead of independent point estimates. IP-Guided Diffusion (Zeng et al., 2024) first learns latent instance and solution representations and trains a conditional diffusion model $\mathbf{f}_\theta$. Let $\mathbf{z}_{\mathbf{x}}$ be the clean solution embedding, $\mathbf{z}_{\mathbf{x}}^{(t)}$ its noisy state at diffusion step $t$, and $\mathbf{z}_{\mathcal{I}}$ the instance embedding. With reconstructed solution $\widetilde{\mathbf{x}}$, its objective is

$$\mathcal{L}_{\mathrm{IPDiff}}(\theta) = \mathbb{E}\left[\left\|\mathbf{f}_\theta\left(\mathbf{z}_{\mathbf{x}}^{(t)}, \mathbf{z}_{\mathcal{I}}, t\right) - \mathbf{z}_{\mathbf{x}}\right\|_2^2\right] + \mathcal{L}_{\mathrm{CE}} + \frac{\lambda}{m}\sum_{i=1}^{m} \max\left(\mathbf{a}_i^\top \widetilde{\mathbf{x}} - b_i, 0\right), \quad (5)$$

where $\mathcal{L}_{\mathrm{CE}}$ reconstructs the discrete solution and $\lambda \geq 0$ weights constraint violations. At inference, objective- and constraint-aware gradients further guide DDPM/DDIM sampling toward promising regions. **ii) Flow-based Generation.** FMIP (Li et al., 2026a) extends generative learning to genuine mixed-variable MILPs. Let $\mathbf{x}^{\mathrm{I}}$ and $\mathbf{x}^{\mathrm{C}}$ denote the integer and continuous parts of a target solution, respectively, and let $G_t$ denote its noisy time-dependent solution graph at time step $t$. FMIP jointly learns a categorical distribution for $\mathbf{x}^{\mathrm{I}}$ and a regression flow for $\mathbf{x}^{\mathrm{C}}$ through

$$\mathcal{L}_{\mathrm{FMIP}}(\theta) = \mathbb{E}_{t,G_1}\left[\frac{\|\widetilde{\mathbf{x}}_\theta^{\mathrm{C}}(G_t) - \mathbf{x}^{\mathrm{C}}\|_2^2}{1-t} - \omega \log \widetilde{\mathbf{p}}_\theta\left(\mathbf{x}^{\mathrm{I}} \mid G_t\right)\right], \quad (6)$$

where $\omega > 0$ balances continuous regression and integer categorical prediction.

**Unsupervised Objective-driven Learning.** Unsupervised learners dispense with solver-generated labels and optimize directly from the MILP formulation. For the binary ILPs considered by DiffILO (Geng et al., 2025), let $\mathbf{p}_\theta \in [0,1]^n$ parameterize an independent Bernoulli distribution $P_{\mathbf{p}_\theta}$ over $\mathbf{x}$, and $\mathbf{a}_i^\top$ denotes the $i$-th row of $\mathbf{A}$, the expected violation of constraint $i$ is $\widehat{\phi}_i(\mathbf{p}_\theta) = \mathbb{E}_{\mathbf{x}\sim P_{\mathbf{p}_\theta}}\left[\max\left(\mathbf{a}_i^\top \mathbf{x} - b_i, 0\right)\right]$. It minimizes the objective-violation merit function $\mathcal{L}_{\mathrm{DiffILO}}(\theta) = \mathbf{c}^\top \mathbf{p}_\theta + \mu \sum_{i=1}^m \widehat{\phi}_i(\mathbf{p}_\theta)$, where $\mu > 0$ controls the feasibility penalty. More recently, Huang et al. (2026a) follows the same label-free principle but learns a Langevin sampling process driven by objective and feasibility information with an LP oracle guiding the search distribution.

### 2.4 Enforcers: From Neural Guidance to Feasibility

Identical learner outputs can lead to substantially different primal solutions depending on how they are consumed. We therefore treat enforcement as an independent design axis and distinguish solver-free *soft* Enforcers from solver-backed *hard* Enforcers.

**Solver-free Soft Enforcers. i) Rounding.** The simplest deterministic strategy is threshold rounding. For a binary variable with predicted probability $\mathbf{p}_j$, the candidate value is $\widetilde{x}_j = \mathbb{I}[\mathbf{p}_j \geq \tau]$, where $\tau \in [0,1]$ is the rounding threshold. **ii) Sampling.** Sampling instead generates $K$ candidate assignments from the predicted distribution, e.g., under Bernoulli or Gumbel sampling, $\widetilde{x}_j^{(k)} \sim \mathrm{Sampler}(\mathbf{p}_j),\ k = 1, \ldots, K$ for binary variables. Then, the sampling Enforcer selects the best candidate out of the $K$ samples according to a *feasibility-first* rule, preferring the best objective among feasible samples when any exist. If all samples are infeasible, their violations remain available for learner-level evaluation. **Note.** Solver-free soft Enforcers apply strictly to pure binary problems, whereas mixed-variable programs require a solver backend to assign continuous variables.

**Solver-backed Hard Enforcers.** Inspired by Li et al. (2026a); Huang et al. (2026a), we streamline the "downstream solvers" as hard Enforcers, which consume learned variable guidance and rely on an exact backend to recover and certify feasibility. **i) Predict-and-Search (PS)** (Han et al., 2023) constructs a prediction-centered restricted search region before exact optimization. **ii) Neural Diving (ND)** (Nair et al., 2021) selectively fixes variables according to prediction confidence and delegates the remaining problem to the solver. **iii) Probabilistic Multi-variable Branching (PMVB)** (Chen et al., 2026) uses predicted probabilities to guide multi-variable branching, whereas **iv) Alternating Prediction-correction (Apollo)** (Liu et al., 2025) alternates prediction and correction during the

solving process. **Note.** Crucially, the enforcement mechanism and exact solver are treated as separate factors, e.g., $\mathsf{E}_{\mathrm{PS}}[\texttt{Gurobi}, B_E]$ and $\mathsf{E}_{\mathrm{PS}}[\texttt{SCIP}, B_E]$ denote the same PS Enforcer instantiated with different backends under the same budget. This separation replaces ambiguous composite labels such as `PS+Gurobi` with explicit component semantics and enables backend-controlled comparison.

### 2.5 Refiners: Post-feasibility Incumbent Improvement

Once a feasible incumbent $\widetilde{\mathbf{x}}^0$ is available, its subsequent improvement is independent of how it was obtained. We therefore isolate post-feasibility optimization as the Refiner stage and evaluate representative strategies under controlled budgets and solver backends.

**Warm-start Solver (WS).** The most direct Refiner passes $\mathbf{x}^0$ to an exact solver as a MIP start and continues optimization for a fixed budget. This tests whether an incumbent generated by a learner-enforcer pair provides useful initialization relative to the same solver without neural guidance.

**Local Branching (LB).** Local Branching (Fischetti & Lodi, 2003) restricts exact optimization to a neighborhood around the current incumbent. For binary decision variables, a Hamming neighborhood around $\mathbf{x}^0$ can be defined by $\sum_{j:\mathbf{x}^0_j=0} \mathbf{x}_j + \sum_{j:\mathbf{x}^0_j=1}(1-\mathbf{x}_j) \le r$, where $r$ is the neighborhood radius. The restricted MILP is then optimized using the chosen backend.

**Large Neighborhood Search (LNS).** Generic LNS (Song et al., 2020) iteratively releases a subset of variables while fixing the remainder to the incumbent, and delegates the subproblem to a MILP solver for repair and improvement. We benchmark three learning-free neighborhood destruction policies: random selection (`Rand`), random partitioning (`Part`), and graph-based breadth-first search (`BFS`).

**Further Discussions.** More Learner paradigms are discussed in Appendix C. Detailed algorithmic procedures for the benchmarked Enforcers and Refiners are in Appendices D and E, respectively.

## 3 Modularized Empirical Investigation

**Datasets.** Following prior work (Gasse et al., 2019), we generate SC/MIS/CA/CFL instances for the main ILP experiments (problem formulations and data statistics are provided in Appendix F). We further use the more challenging WA/BIP tasks (Gasse et al., 2022) and selected MIPLIB instances for generalization tests on mixed-integer settings (results are provided in Appendix I.1).

**Settings.** We retrain all baselines within our unified framework using a shared GNN backbone (see Appendix G). Other detailed settings (hardware, hyper-parameters, etc.) are provided in Appendix H.

**Generic Metrics. 1)** *Objective Value (Obj.)*: The absolute objective value $\mathbf{c}^\top\mathbf{x}$. **2)** *Primal Gap (Gap)*: Percentage relative gap to the best known solution (BKS) obtained by Gurobi with a 3600s time limit, i.e., $\frac{\mathbf{c}^\top\mathbf{x}-\mathbf{c}^\top\mathbf{x}^{\mathrm{ref}}}{\mathbf{c}^\top\mathbf{x}^{\mathrm{ref}}+10^{-10}} \times 100\%$ for minimization tasks. **3)** *Time*: Average per-instance runtime, including neural inference and solver-based post-processing. **Note.** Obj. and Gap are averaged over feasible solutions only. We intentionally avoid normalizing the gap denominator by $\max\{|\mathbf{c}^\top\mathbf{x}|, |\mathbf{c}^\top\mathbf{x}^{\mathrm{ref}}|\}$, which compresses differences in solution quality, and instead follow the standards widely adopted in relevant NCO benchmarks (Berto et al., 2023; Li et al., 2025; Ma et al., 2025b; Yan et al., 2026).

### 3.1 Informativeness of Raw Neural Predictions

In this section, we investigate ***RQ1****: How informative are raw neural predictions for directly producing feasible and high-quality primal solutions before any solver-backed intervention?*

**Targeted Metrics.** To isolate the capability of the Learner itself, beyond the generic metrics, we evaluate the raw predictions using supplementary metrics: **i)** *Instance-wise Feasibility Rate (IFR)* measures the fraction of instances for which at least one feasible solution is obtained. For sampling, **ii)** *Sample-wise Feasibility Rate (SFR)* further measures the fraction of all sampled solutions that are feasible. **iii)** *Infeasible Violation (IViol)* reports the average constraint violation of infeasible predictions. Further, to jointly assess feasibility and solution quality, we propose **iv)** ***Feasible Quality Score (FQS)***. Let $\rho_{\mathrm{feas}} \in [0,1]$ denote IFR and $g \ge 0$ the mean (fractional) feasible-only primal gap. We define $\mathrm{FQS}(\rho_{\mathrm{feas}}, g) := \frac{\rho_{\mathrm{feas}}}{1+\log(1+g)} \times 100\%$. As

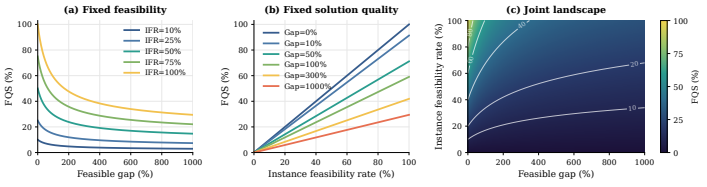



Figure 2: Behavioral landscape of the FQS metric.

Table 2: Results for raw *learner* capability. Rounding (R) threshold: 0.5. Number of samples (S): 1000. (Obj.): mean Obj. when no feasible solution is found. Gurobi/SCIP (H) denote solver heuristics as in Zeng et al. (2024).

| Solver / Learner | Type | SC (100 instances) Obj.↓ | Gap↓ | IFR↑ | SFR↑ | IViol↓ | FQS↑ | Time↓ | MIS (100 instances) Obj.↑ | Gap↓ | IFR↑ | SFR↑ | IViol↓ | FQS↑ | Time↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BKS | Exact | 86.43* | 0.00 | 100 | – | 0.00 | 100.00 | | 684.14* | 0.00 | 100 | – | 0.00 | 100.00 | |
| Gurobi (H) | Heuristic | 2404.16 | 2711.83 | 100 | – | 0.00 | 23.06 | 4.57s | 416.16 | 39.17 | 100 | – | 0.00 | 75.16 | 0.23s |
| SCIP (H) | Heuristic | 1869.58 | 2081.42 | 100 | – | 0.00 | 24.49 | 8.93s | 563.69 | 17.60 | 100 | – | 0.00 | 86.05 | 1.00s |
| Base-SL | SL+R | 94.67 | 8.83 | 3 | – | 9.14 | 2.77 | 0.40s | (680.71) | N/A | 0 | – | 21.78 | 0.00 | 0.06s |
| PS-SL | SL+R | (78.80) | N/A | 0 | – | 12.16 | 0.00 | 0.31s | (678.13) | N/A | 0 | – | 17.19 | 0.00 | 0.06s |
| ConPaS | SL+R | (17.08) | N/A | 0 | – | 2928.52 | 0.00 | 0.35s | 238.80 | 65.64 | 95 | – | 2.00 | 63.14 | 0.06s |
| CoCo | SL+R | 341.90 | 296.49 | **99** | – | **1.00** | 41.64 | 0.44s | (771.22) | N/A | 0 | – | 402.56 | 0.00 | 0.08s |
| OPTFM | UL+SL+R | (38.80) | N/A | 0 | – | 382.01 | 0.00 | 0.23s | (659.56) | N/A | 0 | – | 94.83 | 0.00 | 0.24s |
| DiffILO | UL+R | 123.13 | 46.16 | 67 | – | 1.21 | 48.57 | 0.42s | 636.51 | 6.96 | 100 | – | 0.00 | 93.70 | 0.07s |
| DDIM | SL(Gen)+R | 97.85 | 13.47 | 84 | – | 1.31 | **74.57** | 7.54s | 681.92 | 0.33 | 100 | – | 0.00 | **99.68** | 2.07s |
| FMIP | SL(Gen)+R | 90.96 | **5.95** | 74 | – | 1.31 | 69.96 | 7.64s | 682.02 | 0.30 | 98 | – | 1.00 | 97.71 | 2.15s |
| Base-SL | SL+S | 93.50 | 8.40 | 12 | 2.24 | 4.01 | 11.10 | 1.33s | (673.92) | N/A | 0 | 0.00 | 13.90 | 0.00 | 0.30s |
| PS-SL | SL+S | 109.79 | 30.82 | 52 | 0.11 | 2.67 | 40.99 | 1.16s | (664.85) | N/A | 0 | 0.00 | 19.14 | 0.00 | 0.30s |
| ConPaS | SL+S | 38801.67 | 45279.10 | 100 | **100.00** | 0.00 | 14.05 | 1.12s | 245.43 | 64.12 | 100 | 76.83 | 0.00 | 66.87 | 0.37s |
| CoCo | SL+S | 279.18 | 223.83 | 100 | 97.62 | 0.00 | 45.98 | 1.13s | (743.69) | N/A | 0 | 0.00 | 282.47 | 0.00 | 0.32s |
| OPTFM | UL+SL+S | 797.80 | 753.22 | 7 | 0.02 | 5.83 | 2.23 | 0.97s | (652.57) | N/A | 0 | 0.00 | 247.70 | 0.00 | 0.44s |
| DiffILO | UL+S | 121.37 | 40.50 | 100 | 98.53 | 0.00 | 74.62 | 2.07s | 638.02 | 6.74 | 100 | 99.98 | 0.00 | 93.88 | 1.32s |
| DDIM | SL(Gen)+S | 97.76 | 13.15 | 100 | 86.50 | 0.00 | 89.00 | 8.40s | 681.94 | 0.32 | 100 | 99.31 | 0.00 | 99.68 | 2.55s |
| FMIP | SL(Gen)+S | 91.43 | 5.77 | 100 | 76.74 | 0.00 | **94.69** | 8.44s | 682.12 | 0.30 | 100 | 91.20 | 0.00 | **99.71** | 2.62s |

| Solver / Learner | Type | CA (100 instances) Obj.↑ | Gap↓ | IFR↑ | SFR↑ | IViol↓ | FQS↑ | Time↓ | CFL (100 instances) Obj.↓ | Gap↓ | IFR↑ | SFR↑ | IViol↓ | FQS↑ | Time↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BKS | Exact | 22272.55* | 0.00 | 100 | – | 0.00 | 100.00 | | 11388.79* | 0.00 | 100 | – | 0.00 | 100.00 | |
| Gurobi (H) | Heuristic | 16420.59 | 26.28 | 100 | – | 0.00 | 81.08 | 0.16s | 55043.58 | 383.51 | 100 | – | 0.00 | 38.82 | 0.23s |
| SCIP (H) | Heuristic | 9173.31 | 58.82 | 100 | – | 0.00 | 68.37 | 0.07s | 85311.31 | 655.45 | 100 | – | 0.00 | 33.09 | 0.29s |
| Base-SL | SL+R | 18805.10 | 13.24 | 1 | – | 6.39 | 0.89 | 0.06s | (11151.40) | N/A | 0 | – | 227.01 | 0.00 | 0.08s |
| PS-SL | SL+R | 15719.49 | 28.31 | 2 | – | 5.50 | 1.60 | 0.05s | (10457.62) | N/A | 0 | – | 230.77 | 0.00 | 0.08s |
| ConPaS | SL+R | 0.00 | 100.00 | 100 | – | 0.00 | 59.06 | 0.06s | (0.00) | N/A | 0 | – | 2092.71 | 0.00 | 0.70s |
| CoCo | SL+R | (61903.77) | N/A | 0 | – | 538.73 | 0.00 | 0.11s | (53533.21) | N/A | 0 | – | 55.80 | 0.00 | 0.08s |
| OPTFM | UL+SL+R | 2033.72 | 90.82 | 12 | – | 3.25 | 7.29 | 0.10s | (3514.39) | N/A | 0 | – | 1403.33 | 0.00 | 0.15s |
| DiffILO | UL+R | 15306.40 | 31.26 | 98 | – | 1.50 | 77.04 | 0.06s | 18527.99 | 62.25 | 85 | – | 40.73 | 57.28 | 0.11s |
| DDIM | SL(Gen)+R | 21593.40 | 2.98 | 90 | – | 1.60 | **87.43** | 2.05s | 13264.46 | 17.41 | 26 | – | 46.84 | 22.40 | 2.22s |
| FMIP | SL(Gen)+R | 21658.81 | 2.58 | 89 | – | 1.55 | 86.79 | 2.11s | 13104.81 | 14.85 | 24 | – | 13.41 | 21.08 | 2.24s |
| Base-SL | SL+S | 17946.34 | 19.56 | 16 | 0.28 | 14.01 | 13.57 | 0.12s | (10771.55) | N/A | 0 | 0.00 | 165.69 | 0.00 | 0.43s |
| PS-SL | SL+S | (24611.59) | N/A | 0 | 0.00 | 69.57 | 0.00 | 0.14s | (10284.51) | N/A | 0 | 0.00 | 147.57 | 0.00 | 0.42s |
| ConPaS | SL+S | 1648.24 | 92.60 | 100 | **99.57** | 0.00 | 60.41 | 0.16s | (0.00) | N/A | 0 | 0.00 | 2092.71 | 0.00 | 0.45s |
| CoCo | SL+S | (64242.77) | N/A | 0 | 0.00 | 567.89 | 0.00 | 0.31s | (53165.91) | N/A | 0 | 0.00 | 56.74 | 0.00 | 0.37s |
| OPTFM | UL+SL+S | (16491.45) | N/A | 0 | 0.00 | 66.14 | 0.00 | 0.23s | (20341.20) | N/A | 0 | 0.00 | 807.74 | 0.00 | 0.50s |
| DiffILO | UL+S | 15659.77 | 29.68 | **100** | 98.48 | 0.00 | 79.37 | 0.23s | 18212.19 | 59.55 | **93** | **81.42** | 47.71 | **63.39** | 2.88s |
| DDIM | SL(Gen)+S | 21657.35 | 2.79 | 99 | 87.13 | 2.00 | **96.35** | 2.22s | 13226.95 | 16.66 | 36 | 24.40 | 121.08 | 31.19 | 3.07s |
| FMIP | SL(Gen)+S | 21705.02 | **2.57** | 98 | 82.15 | 11.50 | 95.58 | 2.26s | 12734.76 | **12.34** | 50 | 22.71 | 26.70 | 44.79 | 2.85s |

shown in Fig. 2, it is bounded by the IFR, decreases logarithmically with solution suboptimality, and reaches 100% only when all instances are solved feasibly with zero gap.

**Results and Analysis.** Table 2 reports raw learner performance (full training dynamics are visualized in Appendix I.2), revealing patterns largely obscured by solver intervention. **1) Generative learners achieve strong overall feasibility-quality trade-offs.** In particular, DDIM and FMIP dominate on SC, MIS, and CA, reaching FQS values up to 99.68 and 99.71 under sampling. Their feasible solutions are also substantially closer to the BKS, e.g., on SC, FMIP attains a 5.77% Gap versus 40.50% and 223.83% for DiffILO and CoCo-MILP. These results are consistent with distributional modeling better capturing variable dependencies relevant to solution quality. **2) Formulation-aware unsupervised learning is particularly robust in feasibility.** DiffILO maintains consistently high feasibility across all four tasks, likely because directly optimizing objective and constraint violations keeps predictions closer to the feasible region, whereas reference-based supervised learners may fit high-quality modes without sufficiently covering the feasible manifold. **3) Feasibility and solution quality are only partially coupled.** ConPaS achieves 100% IFR/SFR on SC under sampling but a 45279.10% Gap, while DDIM and FMIP obtain much smaller gaps on CFL but only 36% and 50% IFR. Thus, strong performance on either feasibility or feasible-only quality alone can be misleading, motivating their joint evaluation through FQS. **4) Sampling exposes the quality of the learned distribution beyond its mode.** It substantially improves strong learners when probability mass covers multiple feasible assignments, but cannot rescue poorly aligned score distributions. Correspondingly, high IFR with low SFR indicates feasibility obtained mainly through repeated trials, whereas DiffILO's near-100% SFR on SC, MIS, and CA reflects probability mass stably concentrated around feasible regions.

### 3.2 Synergy with Solver-Supported Enforcers

In this section, we probe ***RQ2****: To what extent does learned variable-wise guidance improve solver-supported feasibility enforcement, and how does this depend on the Learner-Enforcer pairing?*

**Targeted Metrics.** In Sec. 3.2 & 3.3, we report the mean (feasible) gap $g(t)$ at 3 timepoints (G@$t$) and the primal integral (PI) over the entire budget. For each instance $i$, $\mathrm{PI}_i(T) = \int_0^T \min(g_i(t), 100)\,\mathrm{d}t$ is the area under the best-so-far gap curve, with a gap of 100% before the first feasible incumbent.

**Results and Analysis.** Table 3 reveals main results for the Learner-Enforcer interaction. **1) Raw Learner quality does not directly translate into downstream solver gains.** Although DDIM and FMIP dominate the evaluation in Sec. 3.1, the best solver-backed combinations are often different: PS-SL+`PMVB` achieves the lowest PI on SC and MIS, while DiffILO+`PS` performs best on CA.

Table 3: Results for learners with different feasibility *enforcers* at multiple solving timepoints. $S_E$: `Gurobi`. The blue number denotes the instance-wise feasibility rate (IFR) when any infeasible solutions remain.

| Learner | Enforcer | SC (100 inst.) G@10 | SC G@10² | SC G@10³ | SC PI↓ | MIS (100 inst.) G@1 | MIS G@10 | MIS G@10² | MIS PI↓ | CA (100 inst.) G@10 | CA G@10² | CA G@10³ | CA PI↓ | CFL (100 inst.) G@1 | CFL G@10 | CFL G@10² | CFL PI↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gurobi | | 7.52 | 0.93 | 0.14 | 973.68 | 1.31 | 0.02 | 0.00 | 15.99 | 0.68 | 0.09 | 0.00 | 42.99 | 1.07 | 0.10 | 0.00 | 20.24 |
| Base-SL | PS | 15.81 | 0.28 | 0.03 | 884.66 | 0.82 | 0.10 | 0.00 | 5.83 | 0.45 | 0.08 | 0.00 | 44.82 | 45.78(96) | 45.65(96) | 45.65(95) | 4790.16 |
| PS-SL | PS | 14.55 | 0.23 | 0.03 | 870.77 | 1.64 | 0.10 | 0.00 | 7.76 | 0.41 | 0.07 | 0.00 | 35.51 | 42.01(90) | 41.74(90) | 41.74(90) | 4764.77 |
| ConPaS | PS | 9.98 | 0.25 | 0.04 | 922.43 | 0.72 | 0.14 | 0.00 | 8.34 | 0.48 | 0.11 | 0.04 | 84.20 | N/A(0) | N/A(0) | N/A(0) | 10000.00 |
| CoCo | PS | 9.66 | 0.28 | 0.03 | 853.56 | 1.10 | 0.11 | 0.00 | 8.12 | 0.44 | 0.09 | 0.00 | 44.79 | 159.18 | 159.18 | 159.09(98) | 10000.00 |
| OPTFM | PS | 44.85 | 0.24 | 0.01 | 942.75 | 0.87 | 0.12 | 0.00 | 7.30 | 0.48 | 0.09 | 0.00 | 46.89 | 140.82 | 140.82 | 140.82 | 9983.99 |
| DiffILO | PS | 14.13 | 0.24 | 0.01 | 905.08 | 0.42 | 0.04 | 0.00 | 8.41 | 0.38 | 0.06 | 0.00 | 32.82 | 73.58(93) | 73.58(93) | 71.79(97) | 7522.38 |
| DDIM | PS | 30.98 | 0.30 | 0.05 | 1035.82 | 0.16 | 0.05 | 0.00 | 3.47 | 0.44 | 0.09 | 0.00 | 48.06 | 40.85(7) | 40.55(7) | 19.72(39) | 9584.73 |
| FMIP | PS | 9.75 | 0.28 | 0.01 | 842.77 | 0.07 | 0.04 | 0.00 | 4.98 | 0.41 | 0.08 | 0.00 | 46.83 | 54.11(84) | 53.34(84) | 49.10(93) | 6043.84 |
| Base-SL | ND | 3.27 | 0.82 | 0.12 | 725.99 | 0.24 | 0.16 | 0.16 | 19.53 | 1.27 | 0.74 | 0.51 | 585.49 | 2.39 | 1.14 | 0.66 | 90.43 |
| PS-SL | ND | 4.45 | 1.50 | 0.11 | 901.44 | 0.26 | 0.17 | 0.17 | 21.69 | 1.51 | 0.91 | 0.62 | 713.27 | 3.13 | 1.25 | 0.62 | 93.73 |
| ConPaS | ND | 3930.43 | 3765.62 | 3765.62 | 100000.00 | 3.69 | 3.09 | 3.02 | 308.08 | 2.68 | 1.58 | 0.96 | 1221.52 | 4.43 | 2.15 | 1.06 | 169.45 |
| CoCo | ND | 14.28 | 6.75 | 2.98 | 4677.16 | 1.77(79) | 1.43(99) | 1.43(99) | 297.49 | 3.86(17) | 3.85(17) | 3.85(17) | 83663.75 | 68.07 | 48.69 | 48.63 | 4902.66 |
| OPTFM | ND | 49.03 | 6.50 | 1.71 | 4733.22 | 4.50(49) | 3.48(88) | 3.48(88) | 1597.84 | 4.40 | 2.89 | 2.64 | 2753.08 | 20.55 | 10.85 | 10.21 | 1081.95 |
| DiffILO | ND | 5.92 | 1.71 | 0.25 | 1081.00 | 1.35 | 1.08 | 1.08 | 109.58 | 3.18 | 2.14 | 1.58 | 1790.62 | 12.62 | 5.24 | 4.40(98) | 488.25 |
| DDIM | ND | 4.53 | 1.70 | 0.20 | 1080.80 | 0.10 | 0.06 | 0.06 | 6.56 | 1.20 | 0.80 | 0.60 | 669.77 | 3.06 | 1.34 | 0.90 | 115.76 |
| FMIP | ND | 3.26 | 0.89 | 0.14 | 782.43 | 0.11 | 0.08 | 0.08 | 8.39 | 1.27 | 0.80 | 0.65 | 701.22 | 2.65 | 1.09 | 0.73 | 95.08 |
| Base-SL | PMVB | 7.01 | 1.75 | 1.74 | 2210.81 | 0.22 | 0.22 | 0.22 | 22.42 | 2.00 | 2.00 | 2.00(99) | 2003.66 | 5.43(99) | 5.38(99) | 5.38(99) | 639.07 |
| PS-SL | PMVB | 1.55 | 0.22 | 0.04 | 242.75 | 0.03 | 0.02 | 0.02 | 2.80 | 0.68 | 0.11 | 0.06 | 91.22 | 2.54 | 2.18 | 2.17 | 225.20 |
| ConPaS | PMVB | 209.69 | 137.65 | 122.40 | 95564.46 | 42.90 | 42.90 | 42.90 | 4291.71 | 33.47 | 31.60 | 29.89 | 30491.72 | N/A(0) | N/A(0) | N/A(0) | 10000.00 |
| CoCo | PMVB | 130.04 | 129.55 | 129.55 | 98338.85 | 1.91(88) | 1.91(88) | 1.91(88) | 1368.40 | N/A(0) | N/A(0) | N/A(0) | 100000.00 | 257.05(92) | 257.05(92) | 257.05(92) | 10000.00 |
| OPTFM | PMVB | 4.63 | 0.86 | 0.17 | 920.21 | 0.11 | 0.00 | 0.00 | 4.85 | 0.70 | 0.09 | 0.00 | 45.36 | 1.00 | 0.12 | 0.00(98) | 22.77 |
| DiffILO | PMVB | 20.82 | 20.60 | 20.60 | 20698.54 | 4.42 | 4.42 | 4.42 | 442.12 | 13.09 | 13.09 | 13.07(99) | 13089.24 | 44.79 | 44.79 | 44.79 | 4481.28 |
| DDIM | PMVB | 5.25 | 5.25 | 5.25 | 5302.38 | 0.08 | 0.07 | 0.07 | 6.80 | 1.89 | 1.89 | 1.89 | 1890.74 | 10.77 | 10.77 | 10.77 | 1078.74 |
| FMIP | PMVB | 2.11 | 2.11 | 2.11 | 2138.08 | 0.10 | 0.07 | 0.07 | 7.52 | 1.84 | 1.84 | 1.85(99) | 1837.30 | 7.80 | 7.79 | 7.79 | 780.78 |
| Base-SL | Apollo | 7.55 | 0.84 | 0.32 | 1187.94 | 0.44 | 0.06 | 0.00 | 26.78 | 0.72 | 0.08 | 0.00 | 50.60 | 64.95(99) | 64.82(99) | 64.82(99) | 5865.59 |
| PS-SL | Apollo | 4.40 | 0.86 | 0.37 | 1089.59 | 1.96 | 0.06 | 0.00 | 11.14 | 0.62 | 0.06 | 0.00 | 58.42 | 42.08(90) | 41.78(90) | 41.78(90) | 4771.36 |
| ConPaS | Apollo | 7.97 | 0.79 | 0.23 | 1119.46 | 1.10 | 0.11 | 0.02 | 12.34 | 0.84 | 0.13 | 0.04 | 87.51 | N/A(0) | N/A(0) | N/A(0) | 10000.00 |
| CoCo | Apollo | 8.85 | 0.72 | 0.25 | 1031.96 | 4.87 | 0.10 | 0.00 | 46.94 | 0.73 | 0.09 | 0.00 | 79.54 | 146.50 | 146.50 | 146.50 | 10000.00 |
| OPTFM | Apollo | 29.15 | 0.91 | 0.38 | 1329.29 | 1.07 | 0.07 | 0.00 | 7.04 | 0.68 | 0.07 | 0.00 | 40.39 | 140.83 | 140.82 | 140.82 | 9983.78 |
| DiffILO | Apollo | 4.66 | 0.89 | 0.29 | 1062.10 | 0.41 | 0.02 | 0.00 | 10.70 | 0.76 | 0.07 | 0.00 | 47.64 | 77.98(94) | 77.98(94) | 76.40(97) | 7854.11 |
| DDIM | Apollo | 12.02 | 0.75 | 0.25 | 1056.77 | 0.12 | 0.04 | 0.00 | 4.11 | 0.79 | 0.09 | 0.00 | 47.86 | 40.23(9) | 45.63(14) | 27.60(34) | 9239.78 |
| FMIP | Apollo | 4.94 | 0.77 | 0.24 | 986.52 | 0.10 | 0.02 | 0.00 | 3.94 | 0.77 | 0.09 | 0.00 | 44.91 | 53.27(82) | 53.50(93) | 51.70(94) | 5586.59 |

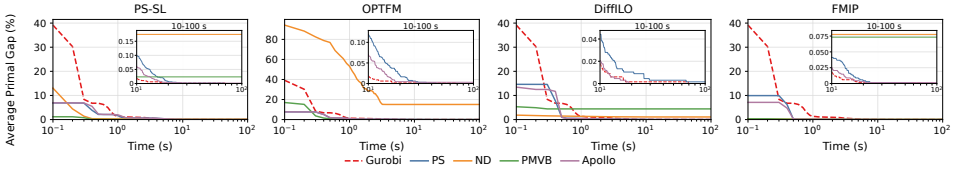



Figure 3: Enforcer comparison on MIS. Full *learner-enforcer* combinations and analysis are in Appendix I.3.

Conversely, strong raw predictions can become less competitive under an incompatible Enforcer. **2) PS provides mostly robust use of neural guidance.** On SC and MIS, most Learners coupled with PS improve the PI over Gurobi. Its trust-region formulation likely allows the backend to override individual prediction errors, making it comparatively tolerant to imperfect scores. However, this robustness does not extend to CFL, where restricted prediction-centered regions frequently yield poor incumbents or fail to recover feasibility. **3) More aggressive Enforcers require better-calibrated learned predictions.** PMVB produces the largest acceleration when its probability-derived restrictions are well aligned with the Learner, e.g., PS-SL on SC/MIS and OPTFM on CFL. However, it degrades sharply with ConPaS, CoCo-MILP, and DiffILO, as it directly converts predicted marginals into cardinality restrictions, where informative rankings alone are insufficient without calibrated probability magnitudes. ND and Apollo are generally more stable through repeated solver interaction, but their stronger correction also reduces the distinction among neural predictions. **4) Final gaps alone substantially understate stage-wise differences.** Strong exact search eventually drives many pipelines to similar small gaps: most PS combinations on SC reach 0.01-0.05% Gap by 1000s, and most PS/Apollo combinations on MIS reach zero by 100s. Yet their PIs remain markedly different, reflecting differences in when high-quality incumbents are found. Hence, the principal value of neural guidance is more of an *anytime acceleration* instead of a different long-horizon endpoint.

### 3.3 Synergy with Solver-Supported Refiners

In this section, we consider ***RQ3****: To what extent can learned incumbents enhance or accelerate solver-backed refinement, and how does this depend on the Learner-Refiner combination?*

**Protocol.** We initialize refinement with a fixed incumbent obtained by `ND(Gurobi-1s)` for CFL and `PS(Gurobi-1s)` for other tasks, ensuring fair comparison among refiners.

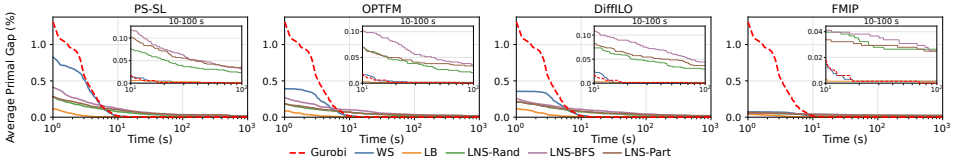



Figure 4: Refiner comparison on MIS. Full *learner-refiner* combinations and analysis are in Appendix I.4.

**Results and Analysis.** Table 4 presents the main results using Gurobi as the example backend. **1) Gains over the solver-only reference depend on the task.** `WS` reduces PI from 15.95 to 1.20-4.06 on MIS, but largely increases it on SC. Thus, learned incumbents do not uniformly improve anytime

Table 4: Results for learners with different incumbent *refiners* at multiple solving timepoints. $S_R$: `Gurobi`.

| Learner | Refiner | SC (100 inst.) G@10 | SC G@10² | SC G@10³ | SC PI↓ | MIS (100 inst.) G@10 | MIS G@10² | MIS G@10³ | MIS PI↓ | CA (100 inst.) G@10 | CA G@10² | CA G@10³ | CA PI↓ | CFL (100 inst.) G@10 | CFL G@10² | CFL G@10³ | CFL PI↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gurobi | | 6.259 | 0.916 | 0.140 | 973.68 | 0.018 | 0.000 | 0.000 | 15.95 | 0.676 | 0.086 | 0.000 | 42.99 | 0.097 | 0.000 | 0.000 | 20.24 |
| Base-SL | WS | 65.182 | 0.955 | 0.162 | 1255.95 | 0.025 | 0.000 | 0.000 | 2.18 | 0.595 | 0.071 | 0.000 | 33.77 | 0.125 | 0.000 | 0.000 | 21.06 |
| PS-SL | WS | 30.583 | 0.899 | 0.162 | 1191.25 | 0.016 | 0.002 | 0.000 | 4.06 | 0.705 | 0.082 | 0.000 | 40.72 | 0.113 | 0.000 | 0.000 | 21.52 |
| ConPaS | WS | 105.007 | 0.940 | 0.163 | 1209.33 | 0.012 | 0.000 | 0.000 | 2.47 | 0.662 | 0.088 | 0.000 | 41.55 | 0.109 | 0.000 | 0.000 | 21.13 |
| CoCo | WS | 51.943 | 0.943 | 0.153 | 1228.89 | 0.016 | 0.000 | 0.000 | 3.05 | 0.726 | 0.063 | 0.000 | 33.96 | 0.104 | 0.000 | 0.000 | 20.71 |
| OPTFM | WS | 44.906 | 1.016 | 0.142 | 1219.57 | 0.016 | 0.002 | 0.002 | 3.55 | 0.665 | 0.082 | 0.000 | 36.73 | 0.108 | 0.000 | 0.000 | 21.64 |
| DiffILO | WS | 83.326 | 0.999 | 0.195 | 1307.18 | 0.027 | 0.000 | 0.000 | 1.93 | 0.647 | 0.088 | 0.000 | 36.69 | 0.107 | 0.000 | 0.000 | 21.30 |
| DDIM | WS | 49.802 | 0.933 | 0.142 | 1207.40 | 0.015 | 0.002 | 0.000 | 1.20 | 0.675 | 0.089 | 0.000 | 40.75 | 0.111 | 0.000 | 0.000 | 21.53 |
| FMIP | WS | 19.814 | 0.955 | 0.121 | 1157.10 | 0.015 | 0.002 | 0.000 | 1.91 | 0.721 | 0.080 | 0.000 | 38.20 | 0.107 | 0.000 | 0.000 | 21.39 |
| Base-SL | LB | 45.953 | 0.879 | 0.182 | 1245.11 | 0.002 | 0.002 | 0.000 | 0.65 | 0.742 | 0.122 | 0.023 | 61.23 | 0.104 | 0.000 | 0.000 | 21.23 |
| PS-SL | LB | 30.898 | 0.993 | 0.210 | 1242.88 | 0.006 | 0.002 | 0.002 | 2.47 | 0.784 | 0.112 | 0.015 | 59.28 | 0.132 | 0.000 | 0.000 | 22.39 |
| ConPaS | LB | 57.030 | 1.044 | 0.230 | 1276.43 | 0.004 | 0.002 | 0.000 | 0.89 | 0.803 | 0.114 | 0.014 | 56.88 | 0.101 | 0.000 | 0.000 | 21.03 |
| CoCo | LB | 37.482 | 0.901 | 0.187 | 1222.50 | 0.003 | 0.000 | 0.000 | 0.83 | 0.871 | 0.108 | 0.006 | 51.42 | 0.112 | 0.000 | 0.000 | 22.45 |
| OPTFM | LB | 15.601 | 0.900 | 0.278 | 1088.15 | 0.003 | 0.000 | 0.000 | 0.59 | 0.801 | 0.087 | 0.007 | 43.66 | 0.109 | 0.000 | 0.000 | 22.39 |
| DiffILO | LB | 33.023 | 0.980 | 0.128 | 1203.57 | 0.004 | 0.002 | 0.000 | 0.83 | 0.755 | 0.136 | 0.013 | 52.85 | 0.078 | 0.000 | 0.000 | 21.36 |
| DDIM | LB | 77.633 | 1.020 | 0.181 | 1302.59 | 0.004 | 0.003 | 0.000 | 1.19 | 0.794 | 0.118 | 0.018 | 62.83 | 0.132 | 0.000 | 0.000 | 21.72 |
| FMIP | LB | 15.031 | 0.951 | 0.211 | 1198.56 | 0.003 | 0.002 | 0.000 | 1.40 | 0.811 | 0.091 | 0.015 | 49.67 | 0.080 | 0.000 | 0.000 | 21.53 |
| Base-SL | LNS-Rand | 5.142 | 2.126 | 1.157 | 1816.74 | 0.075 | 0.028 | 0.015 | 20.89 | 1.268 | 1.161 | 1.015 | 1075.37 | 15.155 | 8.502 | 7.284 | 8041.36 |
| PS-SL | LNS-Rand | 5.091 | 1.912 | 1.143 | 1732.34 | 0.076 | 0.023 | 0.013 | 18.82 | 1.881 | 1.648 | 1.434 | 1524.81 | 15.514 | 7.706 | 6.507 | 7395.04 |
| ConPaS | LNS-Rand | 24.768 | 2.308 | 1.057 | 1984.77 | 0.078 | 0.031 | 0.021 | 26.52 | 2.110 | 1.796 | 1.547 | 1640.15 | 15.803 | 10.332 | 8.898 | 9787.25 |
| CoCo | LNS-Rand | 4.281 | 2.091 | 1.346 | 1820.71 | 0.078 | 0.029 | 0.015 | 21.13 | 1.986 | 1.757 | 1.518 | 1634.55 | 15.600 | 8.488 | 6.973 | 7930.59 |
| OPTFM | LNS-Rand | 4.696 | 1.991 | 1.121 | 1702.71 | 0.069 | 0.020 | 0.016 | 18.98 | 1.459 | 1.319 | 1.162 | 1227.84 | 15.295 | 9.171 | 7.957 | 8664.89 |
| DiffILO | LNS-Rand | 4.730 | 2.147 | 1.075 | 1729.69 | 0.073 | 0.029 | 0.016 | 22.38 | 1.286 | 1.188 | 1.057 | 1104.78 | 15.512 | 8.375 | 7.212 | 8016.68 |
| DDIM | LNS-Rand | 4.713 | 2.131 | 1.037 | 1676.11 | 0.044 | 0.023 | 0.016 | 19.55 | 1.906 | 1.689 | 1.467 | 1574.61 | 14.421 | 7.648 | 6.624 | 7310.73 |
| FMIP | LNS-Rand | 4.390 | 1.922 | 1.071 | 1601.80 | 0.040 | 0.026 | 0.023 | 24.48 | 2.019 | 1.727 | 1.441 | 1568.28 | 13.921 | 8.156 | 7.000 | 7780.43 |

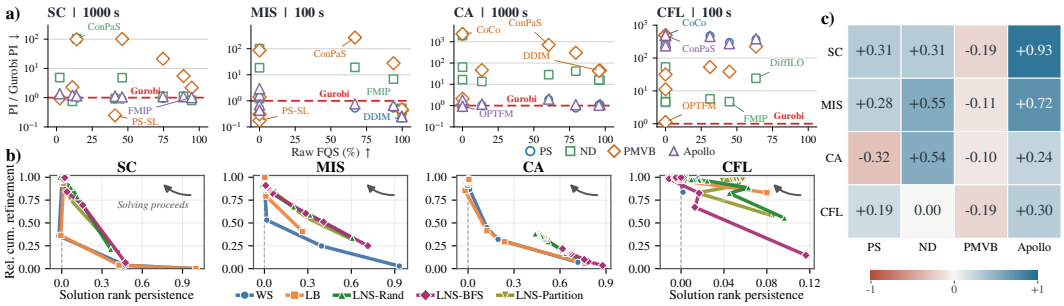



Figure 5: a) Raw learner quality (FQS)↑ vs. enforcer utility (relative PI w.r.t. Gurobi)↓. b) Relative cumulative refinement vs. solution rank persistence during refining. c) Spearman correlation $\rho$(FQS, $-$PI) of enforcers.

performance over Gurobi from scratch. **2) More elaborate refinement is not necessarily stronger.** Despite iterative destroy-and-repair, the tested generic `LNS` variants are generally less competitive than the simpler `WS` and `LB` (e.g., on CA), showing that additional refinement machinery does not guarantee better search than a simple warm start or local neighborhood. **3) The preferred Learner depends on the Refiner.** On MIS, DDIM achieves the lowest PI with `WS` (1.20), while OPTFM does so with `LB` (0.59). These results support evaluating Learners and Refiners as explicit combinations. **4) Refinement reorders Learner rankings**. The strongest Learners before refinement are not always the strongest afterwards, as early incumbent advantages can be largely washed out by exact search.

### 3.4 Correlation Analysis of the Solving Stages

**Learner-Enforcer Compatibility.** Fig. 5a) shows that similar raw FQS can translate into markedly different PIs depending on the Enforcer, while strong solver-backed completion compresses differences among Learners and obscures their predictive quality (Appendix I.5.1). The rank correlations in Fig. 5c) further reveal Enforcer-dependent information preservation: PMVB exhibits weakly negative correlations across tasks, whereas Apollo consistently yields positive correlations, due plausibly to its iterative search that better utilizes Learner scores. **Refiner Persistence and Leverage.** Fig. 5b) characterizes Refiners by the persistence of initial solution-quality rankings and cumulative gap reduction (Appendix I.5.2). Their relationship is strongly task and Refiner dependent: high persistence may merely reflect search inertia, whereas effective convergence may erase the initial ranking. Thus, neither correlation nor persistence alone establishes effective use of learned guidance. Enforcers and Refiners should possibly be co-designed with the semantics of learned outputs and early incumbent benefits to preserve and exploit, rather than substantially override, informative predictions.

## 4 Conclusion

We present ML4MILP-Bench, a modular framework that decomposes learning-based MILP solvers into Learner, Enforcer, and Refiner stages, enabling controlled evaluation and flexible recombination. Comprehensive analysis reveals that solver-backed post-processing can obscure meaningful differences in raw prediction quality, highlighting the need to jointly design learning targets and solver interventions. Through standardized components, evaluation protocols, and extensible interfaces, ML4MILP-Bench provides a reproducible foundation for developing effective and accountable ML4MILP workflows. Limitations and future directions are discussed in detail in Appendix K.

## AI USE STATEMENT

Generative AI tools were used only to improve the readability and linguistic presentation of author-written text. They were not used to develop the methodology, formulate mathematical claims, design experiments, generate or process data. All suggested edits were manually reviewed by the authors, who take full responsibility for the final content of this submission.

## ETHICS STATEMENT

This work does not involve human participants, personally identifiable information, or private and sensitive data. All datasets, software, and experimental procedures were used in accordance with their applicable licenses and research policies. To the best of the authors' knowledge, this work does not raise additional material ethical concerns.

## REPRODUCIBILITY STATEMENT

We provide the information required to reproduce our results throughout the main paper and supplementary materials. The benchmark datasets are described in Appendix F in detail. Complete learner, enforcer, and refiner configurations, training hyperparameters, random seeds, evaluation budgets, etc., are reported in Appendix H. Anonymous source code and scripts for reproducing the main results are included in the repository at https://anonymous.4open.science/r/ICLR27_submission_3182_ML4MILP-Bench.

## REFERENCES

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Federico Berto, Chuanbo Hua, Junyoung Park, Minsu Kim, Hyeonah Kim, Jiwoo Son, Haeyeon Kim, Joungho Kim, and Jinkyoo Park. RL4CO: a unified reinforcement learning for combinatorial optimization library. In *Neural Information Processing Systems 2023, Workshop: New Frontiers in Graph Learning*, 2023.

Maximilian Böther, Otto Kißig, Martin Taraz, Sarel Cohen, Karen Seidel, and Tobias Friedrich. What's wrong with deep learning in tree search for combinatorial optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

Yanguang Chen, Wenzhi Gao, Wanyu Zhang, Dongdong Ge, Huikang Liu, and Yinyu Ye. Data-driven mixed integer optimization through probabilistic multi-variable branching. In *Forty-third International Conference on Machine Learning*, 2026.

Jianpeng 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 *AAAI Conference on Artificial Intelligence*, 2020.

Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. GOAL: A generalist combinatorial optimization agent learner. In *The Thirteenth International Conference on Learning Representations*, 2025.

William Falcon and The PyTorch Lightning Team. Pytorch lightning, 2019.

Shengyu Feng, Weiwei Sun, Shanda Li, Ameet Talwalkar, and Yiming Yang. FrontierCO: Real-world and large-scale evaluation of machine learning solvers for combinatorial optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical programming*, 98(1):23–47, 2003.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 7474–7482, 2021.

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 32*, 2019.

Maxime Gasse, Simon Bowly, Quentin Cappart, Jonas Charfreitag, Laurent Charlin, Didier Chételat, Antonia Chmiela, Justin Dumouchelle, Ambros Gleixner, Aleksandr M. Kazachkov, Elias Khalil, Pawel Lichocki, Andrea Lodi, Miles Lubin, Chris J. Maddison, Morris Christopher, Dimitri J. Papageorgiou, Augustin Parjadis, Sebastian Pokutta, Antoine Prouvost, Lara Scavuzzo, Giulia Zarpellon, Linxin Yang, Sha Lai, Akang Wang, Xiaodong Luo, Xiang Zhou, Haohan Huang, Shengcheng Shao, Yuanming Zhu, Dong Zhang, Tao Quan, Zixuan Cao, Yang Xu, Zhewei Huang, Shuchang Zhou, Chen Binbin, He Minggui, Hao Hao, Zhang Zhiyu, An Zhiwu, and Mao Kun. 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. *arXiv preprint arxiv:2208.14314*, 2026.

Haoyu Geng, Hang Ruan, Runzhong Wang, Yang Li, Yang Wang, Lei Chen, and Junchi Yan. Benchmarking PtO and PnO methods in the predictive combinatorial optimization regime. In *NeurIPS 2024 Datasets and Benchmarks Track*, 2024.

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

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. Miplib 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, 2021.

LLC Gurobi Optimization. Gurobi optimizer reference manual (2020), 2023.

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.

Taoan Huang, Aaron M Ferber, Yuandong Tian, Bistra Dilkina, and Benoit Steiner. Searching large neighborhoods for integer linear programs with contrastive learning. 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. 13869–13890. PMLR, 23–29 Jul 2023.

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.

Weimin Huang, Taoan Huang, Aaron M Ferber, and Bistra Dilkina. Distributional miplib: a multi-domain library for advancing ml-guided milp methods. *arXiv preprint arXiv:2406.06954*, 2024b.

Yixin Huang, Shengyu Feng, and Yiming Yang. Unsupervised neural langevin sampler for mixed integer linear programming. In *Forty-third International Conference on Machine Learning*, 2026a.

Yixin Huang, Liangliang Shi, and Junchi Yan. One-step diffusion solver for non-binary integer linear programming, 2026b.

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

Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Minwei Kong, Chonghe Jiang, Ao Qu, Wenbin Ouyang, Zhaoming Zeng, Xiaotong Guo, Zhekai Li, Junyi Li, Yi Fan, Xinshou Zheng, et al. Frontieror: Benchmarking llms' capacity for efficient algorithm design in large-scale optimization. *arXiv preprint arXiv:2605.25246*, 2026.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Tae-Hoon Lee and Min-Soo Kim. RL-SPH: Learning to achieve feasible solutions for integer linear programs. In *Forty-third International Conference on Machine Learning*, 2026.

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*, 2026a.

Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. From distribution learning in training to gradient search in testing for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast T2T: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Yang Li, Jiale Ma, Wenzheng Pan, Runzhong Wang, Haoyu Geng, Nianzu Yang, and Junchi Yan. Unify ML4TSP: Drawing methodological principles for tsp and beyond from streamlined design space of learning and search. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yang Li, Lvda Chen, Haonan Wang, Runzhong Wang, and Junchi Yan. Generation as search operator for test-time scaling of diffusion-based combinatorial optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026b.

Chang Liu, Zhichen Dong, Haobo Ma, Weilin Luo, Xijun Li, Bowen Pang, Jia Zeng, and Junchi Yan. L2P-MIP: Learning to presolve for mixed integer programming. In *The Twelfth International Conference on Learning Representations*, 2024.

Defeng Liu, Matteo Fischetti, and Andrea Lodi. Learning to search in local branching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3796–3803, 2022.

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 *The Thirteenth International Conference on Learning Representations*, 2025.

Anqi Lu and Junchi Yan. Learning initial basis selection for linear programming via duality-inspired tripartite graph representation and comprehensive supervision. In *Forty-second International Conference on Machine Learning*, 2025.

Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. COExpander: Adaptive solution expansion for combinatorial optimization. In *Forty-second International Conference on Machine Learning*, 2025a.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. ML4CO-Bench-101: Benchmark machine learning for classic combinatorial problems on graphs. In *Advances in Neural Information Processing Systems*, 2025b.

Stephen Maher, Matthias Miltenberger, João Pedro Pedroso, Daniel Rehfeldt, Robert Schwarz, and Felipe Serrano. PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pp. 301–307. Springer International Publishing, 2016. doi: 10.1007/978-3-319-42432-3_37.

Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, Yichuan Wang, Xieting Chu, Zerui Cheng, Yuan Xu, Tian Xia, Zirui Wang, Tianneng Shi, Jianzhu Yao, Yilong Zhao, Qizheng Zhang, Charlie F. Ruan, Zeyu Shen, Kaiyuan Liu, Zhaoyang Hong, Alex Gu, Ziyi Zhang, Runyuan He, Dong Xing, Zerui Li, Zirong Zeng, Yige Jiang, Lufeng Cheng, Ziyi Zhao, Youran Sun, Suyang Zhong, Junpeng Wang, Donglin Li, Wenyuan Huang, Jialiang Gu, Wesley Kai Zheng, Wangmeiyu Zhang, Ruyi Ji, Xuechang Tu, Zihan Zheng, Zhaozi Wang, Zexing Chen, Jingbang Chen, JIALU ZHANG, Aleksandra Korolova, Peter Henderson, Pramod Viswanath, Vijay Ganesh, Saining Xie, Zhuang Liu, Dawn Song, Sewon Min, Ion Stoica, Joseph E. Gonzalez, Jingbo Shang, and Alvin Cheung. FrontierCS: Evolving challenges for evolving intelligence. In *Forty-third International Conference on Machine Learning*, 2026.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2021.

Wenzheng Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. UniCO: On unified combinatorial optimization via problem reduction to matrix-encoded general TSP. In *The Thirteenth International Conference on Learning Representations*, 2025.

Wenzheng Pan, Jiale Ma, Nuoyan Chen, Yang Li, and Junchi Yan. Problem distributions as tasks: Repurposing meta learning for generative combinatorial optimization towards multi-task pretraining and adaptation. In *Forty-third International Conference on Machine Learning*, 2026.

Antoine Prouvost, Justin Dumouchelle, Lara Scavuzzo, Maxime Gasse, Didier Chételat, and Andrea Lodi. Ecole: A gym-like library for machine learning in combinatorial optimization solvers. *arXiv preprint arxiv:2011.06069*, 2020.

Tianle Pu, Jianing Li, Yingying Gao, Shixuan Liu, Zijie Geng, Haoyang Liu, Chao Chen, and Changjun Fan. Coco-milp: Inter-variable contrastive and intra-constraint competitive milp solution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24882–24890, 2026.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems 35*, 2022.

ravichandra addanki, Vinod Nair, and Mohammad Alizadeh. Neural large neighborhood search. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*, 2020.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Forty-first International Conference on Machine Learning*, 2024.

Jialin Song, ravi lanka, Yisong Yue, and Bistra Dilkina. A general large neighborhood search framework for solving integer linear programs. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20012–20023. Curran Associates, Inc., 2020.

Nicolas Sonnerat, Pengming Wang, Ira Ktena, Sergey Bartunov, and Vinod Nair. Learning a large neighborhood search algorithm for mixed integer programs. *arXiv preprint arXiv:2107.10201*, 2021.

IBM ILOG CPLEX Optimization Studio. V20.1: User's manual for cplex. *IBM Corp*, pp. 1–20, 2020.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arxiv:1807.03748*, 2019.

Runzhong Wang, Ziao Guo, Wenzheng Pan, Jiale Ma, Yikai Zhang, Nan Yang, Qi Liu, Longxuan Wei, Hanxue Zhang, Chang Liu, Zetian Jiang, Xiaokang Yang, and Junchi Yan. Pygmtools: A python graph matching toolkit. *Journal of Machine Learning Research*, 25(33):1–7, 2024.

Yaoxin Wu, Wen Song, Zhiguang Cao, and Jie Zhang. Learning large neighborhood search policy for integer programming. In *Advances in Neural Information Processing Systems*, volume 34, pp. 30075–30087, 2021.

Junchi Yan, Jiaxi Liu, Yihui Tu, Fangyuan Zhou, Wenzheng Pan, Zhongteng Gui, and Liangliang Shi. Design linear constrained neural layers with implicit convex optimization. In *Forty-third International Conference on Machine Learning*, 2026.

Huigen Ye, Hua Xu, Hongyan Wang, Chengming Wang, and Yu Jiang. GNN&GBDT-guided fast optimizing framework for large-scale integer programming. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 39864–39878. PMLR, 23–29 Jul 2023.

Huigen Ye, Hua Xu, and Hongyan Wang. Light-MILPopt: Solving large-scale mixed integer linear programs with lightweight optimizer and small-scale training dataset. In *The Twelfth International Conference on Learning Representations*, 2024.

Huigen Ye, Yaoyang Cheng, Hua Xu, Zhiguang Cao, and Hanzhang Qin. Milpbench: A large-scale benchmark test suite for mixed integer linear programming problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 94–103. Association for Computing Machinery, 2025.

Hao Yuan, Wenli Ouyang, Changwen Zhang, Congrui Li, and Yong Sun. OPTFM: A scalable multi-view graph transformer for hierarchical pre-training in combinatorial optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Hao Yuan, Wenli Ouyang, Changwen Zhang, Yong Sun, Liming Gong, and Junchi Yan. BTBS-LNS: Binarized-tightening, branch and search on learning LNS policies for MIP. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Hao Zeng, Jiaqi Wang, Avirup Das, Junying He, Kunpeng Han, Haoyuan Hu, and Mingfei Sun. Effective generation of feasible solutions for integer programming via guided diffusion. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4107–4118, 2024.

Changwen Zhang, Wenli Ouyang, Hao Yuan, Liming Gong, Yong Sun, Ziao Guo, Zhichen Dong, and Junchi Yan. Towards imitation learning to branch for MIP: A hybrid reinforcement learning based sample augmentation approach. In *The Twelfth International Conference on Learning Representations*, 2024.

Jiayi Zhang, Chang Liu, Xijun Li, Hui-Ling Zhen, Mingxuan Yuan, Yawen Li, and Junchi Yan. A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519: 205–217, 2023.

# Appendix

## CONTENTS


**A Full Related Work and Discussion** **17**

A.1 Representative Works Benchmarking Learning-to-Optimize . . . . . 17

A.2 Scope Clarification . . . . . 17

A.3 Comparison of the Gap Metric in Existing Works . . . . . 18

A.4 Comparison of Solver Versions in Existing Works . . . . . 18

**B Details of the Variable-Constraint Bipartite Representation** **19**

**C Additional Learner Pipeline Discussion** **20**

**D Details of the Enforcer Algorithms** **20**

D.1 Predict-and-Search (PS) . . . . . 21

D.2 Neural Diving (ND) . . . . . 21

D.3 Probabilistic Multi-Variable Branching (PMVB) . . . . . 21

D.4 Alternating Prediction-correction (Apollo) . . . . . 22

**E Details of the Refiner Algorithms** **22**

E.1 Warm-start Solver (WS) . . . . . 22

E.2 Local Branching (LB) . . . . . 23

E.3 Large Neighborhood Search (LNS) with Diverse Destroy Policies . . . . . 23

**F Details of the Datasets** **23**

F.1 Problem Definitions and MILP Formulations . . . . . 23

F.2 Dataset Statistics . . . . . 26

**G Details of the Unified Backbone Architecture** **27**

**H Details of Training and Evaluation Settings** **28**

H.1 Common Protocol . . . . . 28

H.2 Learner-specific Settings . . . . . 28

H.3 Enforcer-specific Settings . . . . . 29

H.4 Refiner-specific Settings . . . . . 29

**I Additional Empirical Results** **30**

I.1 Generalization Results on Mixed-integer and Real-world Instances . . . . . 30

I.2 Training Dynamics for the Learners . . . . . 32

I.3 Supplementary Results on Learners with Different Enforcers . . . . . 36

I.4 Supplementary Results on Learners with Different Refiners . . . . . 42

I.5 Additional Results and Discussion for Stage-wise Correlation . . . . . 45
I.5.1 Full Learner-Enforcer Scatterplot and Discussion . . . . . 45
I.5.2 Full Results and Discussion for Refiner Behavior Study . . . . . 46

**J Minimal Code Example for ML4MILP-Bench Usage** **47**
J.1 Data Generation and Reference Solutions . . . . . 48
J.2 Traditional MILP Solvers . . . . . 49
J.3 Learner Construction and Training . . . . . 50
J.4 Modular Learner-Enforcer-Refiner Evaluation . . . . . 51

**K Limitation and Future Work** **52**

# A Full Related Work and Discussion

## A.1 Representative Works Benchmarking Learning-to-Optimize

To support sustainable progress in learning-to-optimize (L2O), a growing body of surveys, benchmarks, datasets, and toolkits has been developed. For ML-assisted MILP solving, Zhang et al. (2023) review learning paradigms that accelerate or guide solver-internal procedures, while Ecole (Prouvost et al., 2020) exposes such solver decisions through standardized learning environments. MIPLIB 2017 (Gleixner et al., 2021), the ML4CO Competition (Gasse et al., 2022), Distributional MIPLIB (Huang et al., 2024b), and MILPBench (Ye et al., 2025) provide increasingly diverse instance collections and evaluation settings for mathematical optimization. Beyond MILP, RL4CO (Berto et al., 2023) provides a unified reinforcement-learning library for NCO; ML4CO-Bench-101 (Ma et al., 2025b) systematically categorizes and evaluates neural solvers across classical graph CO problems; ML4TSPBench (Li et al., 2025) disentangles learning and search designs for TSP; PredictiveCO (Geng et al., 2024) modularizes predict-then-optimize and predict-and-optimize pipelines for CO under coefficient uncertainty; Böther et al. (2022) proposes to benchmark the maximum independent set problem and Wang et al. (2024) standardizes graph-matching research. Recent Frontier-series benchmarks (Feng et al., 2026; Mang et al., 2026; Kong et al., 2026) further broaden evaluation toward real-world CO instances, open-ended computer-science tasks, and LLM-based optimization modeling and algorithm design. Collectively, these efforts have substantially improved reproducibility, accessibility, and empirical rigor across ML4CO.

## A.2 Scope Clarification

**1) Relation to Existing Benchmarks and Toolkits.** Our work is complementary to these infrastructures but targets a different methodological gap. MIPLIB-style resources primarily standardize *what instances* optimization methods are evaluated on, Ecole focuses on learning within solver decision processes, while recent ML4CO benchmarks mainly organize neural solvers for graph-structured CO or predictive optimization. In contrast, *ML4MILP-Bench* focuses on *how prediction-centric MILP primal heuristics should be decomposed and compared*. Specifically, we study methods that expose explicit variable-wise neural guidance, e.g., logits, marginals, or heatmaps, and disentangle their interaction with solver-free or solver-backed feasibility enforcement and subsequent incumbent refinement. Beyond method reproduction, our infrastructure standardizes MILP data I/O, instance and supervision generation, solution pools and reference solutions, solver APIs, learning and search base classes, evaluation, and pretrained checkpoints. This design is intended to enable controlled component substitution and recombination rather than another method-level leaderboard.

**2) Relation to Solver-internal Learning.** A substantial and influential line of ML4MILP research learns to improve individual procedures inside traditional solvers, including branching (Gasse et al., 2019; Zhang et al., 2024), presolving (Liu et al., 2024), basis selection (Lu & Yan, 2025), and other solver-integrated mechanisms (Ye et al., 2023; Yan et al., 2026; Ye et al., 2024). These studies have established important foundations for learning-augmented mathematical optimization, including the now widely adopted variable-constraint bipartite representation (Gasse et al., 2019). We do not regard them as competing paradigms to the methods benchmarked here. Their learned decisions are typically coupled to fine-grained solver internals, whereas our central object of study is an explicit neural primal signal that can be independently passed across downstream solving modules. Incorporating the former would therefore require a substantially different abstraction and experimental protocol. We deliberately focus on the latter to enable sufficiently controlled stage-wise attribution, while leaving solver-internal learning as a natural direction for future extension of the toolkit.

**3) Relation to Learned Large Neighborhood Search.** Learning-enhanced LNS constitutes another substantial research line for MILP/IP solving. Representative works learn neighborhood selection through imitation or reinforcement learning, combine neural initialization with learned neighborhood policies, or employ contrastive learning to identify promising large neighborhoods (Wu et al., 2021; Huang et al., 2023; ravichandra addanki et al., 2020; Yuan et al., 2025b; Liu et al., 2022; Sonnerat et al., 2021). These approaches make important progress in learning *where to search* around an incumbent and can achieve strong anytime performance by repeatedly destroying part of the current solution and invoking an exact solver for repair or reoptimization. Under our taxonomy, such methods naturally belong to the *Refiner* stage because they generally assume, or explicitly construct before refinement, an incumbent from which iterative improvement proceeds.

As such, our goal is not to exhaust this already broad subfield. Instead, the present benchmark emphasizes the earlier and comparatively under-isolated interaction between *Learners* that produce explicit variable-wise primal guidance and *Enforcers* that translate such guidance into complete or feasible incumbents, followed by representative Refiners that quantify the downstream value of those incumbents. We therefore include canonical learning-free refinement mechanisms and position learned LNS methods within the taxonomy without attempting an exhaustive comparison of their neighborhood-policy designs. This scope permits deeper controlled recombination of the Learner-Enforcer-Refiner pipeline while keeping the benchmark tractable and scientifically attributable.

**Overall Scope.** ML4MILP is a broad and rapidly developing area in which solver-internal learning, neural primal generation, learned decomposition, LNS, and traditional optimization are complementary rather than mutually exclusive directions. Our benchmark does not aim to subsume all of them in a single study. Instead, it concentrates on a well-defined and increasingly prominent family of methods whose neural modules explicitly generate variable-wise primal guidance, and systematically studies how this learned information behaves before and after feasibility enforcement and refinement. The resulting scope covers a large body of recent near/end-to-end ML4MILP methods while enabling substantially deeper reproduction, alignment, and component-wise analysis than would be possible under superficial coverage of all learning-assisted solver paradigms. We view broader integration of solver-internal policies and learned Refiners as part of the continued development of our framework.

### A.3 Comparison of the Gap Metric in Existing Works

Table 5: An illustrative example of inconsistent metric definitions: how primal gap is calculated across works. $BKS$ and $OBJ$ denote the objective value of the best known solution and the model solved solution, respectively.

| **Method** | IP-Guided Diffusion | PS | ConPaS | CoCo-MILP | FMIP | | RL-SPH | DiffILO |
|---|---|---|---|---|---|---|---|---|
| **Gap** | $\frac{\lvert OBJ-BKS\rvert}{\max\{\lvert OBJ\rvert,\lvert BKS\rvert\}}$ | $\frac{\lvert OBJ-BKS\rvert}{\lvert BKS\rvert+10^{-10}}$ | $\frac{\lvert OBJ-BKS\rvert}{\max\{\lvert OBJ\rvert,\lvert BKS\rvert,10^{-8}\}}$ | $\lvert OBJ-BKS\rvert$ | $\begin{cases}\frac{\lvert OBJ-BKS\rvert}{\lvert BKS\rvert+1}, & \text{MIPLIB}\\ \lvert OBJ-BKS\rvert, & \text{else}\end{cases}$ | | $\frac{\lvert OBJ-BKS\rvert}{\max\{\lvert OBJ\rvert,\lvert BKS\rvert,\epsilon\}}$ | $\frac{\lvert OBJ-BKS\rvert}{\lvert BKS\rvert}$ |

### A.4 Comparison of Solver Versions in Existing Works

Table 6: Mathematical programming solvers used in representative ML4MILP studies. Exact reported versions are accompanied by their release dates. "NR" indicates that the solver was used but its exact version was not reported; "–" indicates that the solver was not used in the reported experiments. †MILPBench reports 'SCIP 4.3.0"; however, 4.3.0 corresponds to PySCIPOpt rather than a SCIP solver release, so the underlying SCIP version cannot be identified. ‡FMIP uses COPT for training-label generation and Gurobi for downstream solving and classical baselines, without specifying their exact versions.

| **Method** | **Gurobi** | **SCIP** | **COPT** |
|---|---|---|---|
| Predict-and-Search (PS) (Han et al., 2023) | 9.5.2 (Jul. 2022) | 8.0.1 (Jun. 2022) | – |
| LNS (Song et al., 2020) | 9.0 (Nov. 2019) | – | – |
| CL-LNS (Huang et al., 2023) | – | 8.0.1 (Jun. 2022) | – |
| Apollo-MILP (Liu et al., 2025) | NR | NR | – |
| ConPaS (Huang et al., 2024a) | 10.0.0 (Nov. 2022) | 8.0.1 (Jun. 2022) | – |
| MILPBench (Ye et al., 2025) | 11.0.1 (Mar. 2024) | NR† | – |
| ML4CO-Bench-101 (Ma et al., 2025b) | 11.0.3 (Jul. 2024) | – | – |
| RL-SPH (Lee & Kim, 2026) | NR | 8.1.0 (Nov. 2023) | – |
| CoCo-MILP (Pu et al., 2026) | 11.0.3 (Jul. 2024) | NR | – |
| BTBS-LNS (Yuan et al., 2025b) | 9.5.0 (Nov. 2021) | 7.0.3 (Aug. 2021) | – |
| One-Step Diffusion Solver (Huang et al., 2026b) | NR | NR | NR |
| UNLS4MILP (Huang et al., 2026a) | NR | – | – |
| Guided Diffusion (Zeng et al., 2024) | 9.5.2 (Jul. 2022) | 8.0.1 (Jun. 2022) | – |
| GCNN-MIP (Gasse et al., 2019) | – | 6.0.1 (Jan. 2019) | – |
| LinConLayer (Yan et al., 2026) | NR | – | – |
| ML4CO Competition (Gasse et al., 2022) | – | NR | – |
| OPTFM (Yuan et al., 2025a) | NR | NR | – |
| DiffILO (Geng et al., 2025) | 10.0.1 (Feb. 2023) | 8.1.0 (Nov. 2023) | – |
| FMIP (Li et al., 2026a) | NR | – | NR‡ |
| **ML4MILP-Bench (ours)** | **13.0.1 (Nov. 2025)** | **10.0.0 (Nov. 2025)** | **8.0.3 (Jan. 2026)** |

**Discussion.** Solver evolution is an important but often overlooked confounder in ML4MILP evaluation. As Table 6 shows, several recent studies evaluate marginal learning-based improvements on solver releases that were already multiple years old at publication time; for example, BTBS-LNS (Yuan et al., 2025b) uses Gurobi 9.5.0 and SCIP 7.0.3, released in 2021, while Guided Diffusion (Zeng et al., 2024) relies on Gurobi 9.5.2 and SCIP 8.0.1 from 2022. Meanwhile, modern MIP solvers continuously improve presolve, primal heuristics, branching, cutting planes, and search strategies. Gurobi, for instance, reports further MIP performance improvements in its 13.0 release. Consequently, gains measured against a stale backend may conflate the contribution of learned guidance with limitations of the particular solver generation. We therefore explicitly report solver versions and re-evaluate solver-backed Enforcers and Refiners with substantially newer backends under aligned configurations and computational budgets. This does not invalidate results obtained with earlier solver versions, but provides a more current reference for assessing whether learned components retain their incremental value as conventional solvers evolve.

## B Details of the Variable-Constraint Bipartite Representation

Table 7: Features in the unified MILP bipartite representation. The optional positional encoding is not included in the default dimensions.

| Type | Feature | Dim. | Description |
|---|---|---|---|
| Variable | `obj` | 1 | Objective coefficient $c_j$ |
| | `mean_coeff` | 1 | Mean column coefficient $\frac{1}{m}\sum_{i=1}^{m} A_{ij}$ |
| | `lb` | 1 | Lower bound $\ell_j$ of variable $x_j$ |
| | `ub` | 1 | Upper bound $u_j$ of variable $x_j$ |
| | `is_int` | 1 | Indicator $\mathbb{I}[x_j$ is integer-valued$]$ |
| | `degree` | 1 | Variable degree $\lvert\mathcal{N}(v_j)\rvert$ |
| | `max_coeff` | 1 | Maximum incident coefficient $\max_{i\in\mathcal{N}(v_j)} A_{ij}$ |
| | `min_coeff` | 1 | Minimum incident coefficient $\min_{i\in\mathcal{N}(v_j)} A_{ij}$ |
| Constraint | `mean_coeff` | 1 | Mean incident coefficient $\frac{1}{\lvert\mathcal{N}(c_i)\rvert}\sum_{j\in\mathcal{N}(c_i)} A_{ij}$ |
| | `degree` | 1 | Constraint degree $\lvert\mathcal{N}(c_i)\rvert$ |
| | `rhs` | 1 | Right-hand-side value $b_i$ |
| | `max_coeff` | 1 | Maximum incident coefficient $\max_{j\in\mathcal{N}(c_i)} A_{ij}$ |
| | `min_coeff` | 1 | Minimum incident coefficient $\min_{j\in\mathcal{N}(c_i)} A_{ij}$ |
| Edge | `coeff` | 1 | Signed matrix coefficient $A_{ij}$ |
| BIP only | `pos_emb` | 12 | Binary encoding of the instance-local variable index |

Following established practice, we apply the deterministic representation operator $\Phi$ defined in the main text to map each MILP instance $\mathcal{I}$ to $G_{\mathcal{I}} = \Phi(\mathcal{I}) = (\mathcal{V}, \mathcal{C}, \mathcal{E}, \mathbf{X}^v, \mathbf{X}^c, \mathbf{X}^e)$. Consistent with Eq. 1, $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{C} = \{c_1, \ldots, c_m\}$ denote the variable and constraint nodes, and $(c_i, v_j) \in \mathcal{E}$ iff $A_{ij} \neq 0$, where $A_{ij}$ is the $(i, j)$-th entry of $\mathbf{A}$. Inputs parsed from LP or MPS files are first canonicalized to $\mathbf{Ax} \leq \mathbf{b}$: a greater-than constraint is multiplied by $-1$, while an equality is represented by two opposite inequalities. Maximization objectives are handled under the minimization convention specified in the main text.

Throughout ML4MILP-Bench, the resulting common representation uses $\mathbf{X}^v \in \mathbb{R}^{n\times d_v}$, $\mathbf{X}^c \in \mathbb{R}^{m\times d_c}$, and $\mathbf{X}^e \in \mathbb{R}^{|\mathcal{E}|\times d_e}$, where $(d_v, d_c, d_e) = (8, 5, 1)$. Let $\mathcal{N}(v_j) = \{i \in [m] : A_{ij} \neq 0\}$ and $\mathcal{N}(c_i) = \{j \in [n] : A_{ij} \neq 0\}$. Table 7 specifies each feature channel[2]. All features are computed solely from the canonical MILP coefficients, variable bounds, and sparsity pattern. Feature extraction requires $O(m + n + \mathrm{nnz}(\mathbf{A}))$ time, where $\mathrm{nnz}(\mathbf{A})$ denotes the number of nonzero entries in the constraint matrix $\mathbf{A}$.

**Note.** Following Han et al. (2023), the 12-dimensional positional encoding is enabled only for the BIP task whose original protocol relies on variable ordering; it is excluded from the default representation used for cross-method alignment. Iterative and generative learners may additionally append their

[2]The problem instances considered in our evaluation contain no variables with non-finite bounds: all lower and upper bounds supplied to the bipartite encoder are finite.

current solution or noisy state as a method-specific dynamic channel. Such dynamic inputs are supplied after $\Phi(\mathcal{I})$ is constructed and do not modify the shared static feature contract above.

**Discussion.** Our static feature set is designed to cover the core instance information broadly exploited by prior bipartite-graph MILP models (Ding et al., 2020; Gasse et al., 2019; Zeng et al., 2024; Geng et al., 2025; Li et al., 2026a), including objective coefficients, variable bounds and integrality, constraint right-hand sides, graph degrees, and signed matrix coefficients together with simple local statistics. In contrast, we deliberately exclude solver-state features introduced for solver-internal decision making in Gasse et al. (2019), such as `basis_status`, `reduced_cost`, `is_tight`, LP solution values, dual values, and LP `age`. Such quantities are only available after invoking and progressing within an LP/MIP solver, and would therefore entangle instance representation with a particular solving state. This choice follows the central design principle of our benchmark: the shared representation $\Phi(\mathcal{I})$ should remain *solver-agnostic*, so that neural solution learning can be cleanly separated from subsequent solver-backed enforcement and refinement.

## C Additional Learner Pipeline Discussion

**Self-supervised Representation Pre-training.** OPTFM (Yuan et al., 2025a) differs conceptually from solution-level unsupervised learning: it first learns transferable MILP representations through dual-level self-supervision. At the node level, edges in the variable-constraint bipartite graph are masked and reconstructed. Let $\mathcal{M}$ denote the set of masked variable-constraint pairs, $e_{ij} = \mathbb{I}[A_{ij} \neq 0]$ their adjacency labels, and $\widehat{e}_{ij} \in [0, 1]$ the predicted edge probabilities. The reconstruction loss is

$$\mathcal{L}_{\text{rec}} = -\frac{1}{|\mathcal{M}|} \sum_{(i,j)\in\mathcal{M}} \left[ e_{ij} \log \widehat{e}_{ij} + (1 - e_{ij}) \log(1 - \widehat{e}_{ij}) \right]. \tag{7}$$

At the instance level, OPTFM additionally applies ProxyAnchor-based contrastive learning to pull different graph partitions of the same MILP together while separating different instances. The resulting frozen representations are then consumed by task-specific downstream predictors; in our framework, they are adapted to the common variable-score interface.

**Reinforcement Learning.** RL-SPH (Lee & Kim, 2026) formulates primal solution generation as sequential variable rewriting and trains an ILP-specific Actor-Critic policy. At step $t$, the policy $\pi_\theta$ selects variable-update actions from the current observation and maximizes the expected cumulative reward

$$J_{\text{RL}}(\theta) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{T-1} \gamma^t \mathcal{R}_{t,\text{total}} \right], \tag{8}$$

where $\gamma \in [0, 1]$ is the discount factor and $\mathcal{R}_{t,\text{total}}$ combines optimization and exploration rewards. Crucially, RL-SPH adopts a two-phase objective:

$$\mathcal{R}_{t,\text{opt}} = \begin{cases} \mathcal{R}_{t,\text{p1}}, & \text{before the first feasible solution is found,} \\ \mathcal{R}_{t,\text{p2}}, & \text{after a feasible incumbent is available,} \end{cases} \tag{9}$$

where $\mathcal{R}_{t,\text{p1}}$ prioritizes reduction of bound and linear-constraint violations, while $\mathcal{R}_{t,\text{p2}}$ rewards objective improvement subject to maintaining feasibility. Unlike static variable-score predictors, RL-SPH learns a sequential policy explicitly aligned with the transition from infeasibility to feasibility and subsequent incumbent improvement. Also, the authors have not yet open-sourced their work. Therefore, we do not catalog RL-SPH into our benchmark in the current stage.

## D Details of the Enforcer Algorithms

**Common interface and notation.** An Enforcer maps learner scores to a complete primal solution, potentially by restricting the search space of an exact solver. For an instance $\mathcal{I}$, let $\mathbf{s} \in \mathbb{R}^n$ be the variable-wise learner output and $p_j = \sigma(s_j)$ the corresponding probability for a binary variable. We write $\mathcal{B} = \{j : x_j \in \{0, 1\}\}$ for the binary-variable index set. The solver-assisted Enforcers considered below preserve the original feasible set $\mathcal{X}(\mathcal{I})$ and add prediction-dependent fixing or cardinality constraints. Hence, whenever the augmented problem returns an incumbent, the result is feasible for the original MILP. Our implementations of Neural Diving (ND), Predict-and-Search (PS),

PMVB, and Apollo are bench-native adaptations of the public downstream-solver implementations released with FMIP (Li et al., 2026a), which instantiate the methods of Nair et al. (2021), Han et al. (2023), Chen et al. (2026), and Liu et al. (2025), respectively. The current implementations target binary integer predictions; variables not selected by an Enforcer, including continuous variables, remain free for the backend solver.

### D.1 Predict-and-Search (PS)

PS (Han et al., 2023) replaces irreversible variable fixing with a trust-region search. We follow the FMIP-style `threshold` mode to let $\mathcal{B}$ denote the set of candidate binary variables. Fractional parameters $k_0, k_1 \in [0, 1]$ are converted into cardinalities $K_0 = \lfloor k_0|\mathcal{B}| \rfloor$ and $K_1 = \lfloor k_1|\mathcal{B}| \rfloor$. Let $\mathcal{S} \subseteq \mathcal{B}$ denote the variables selected by PS. We partition $\mathcal{S}$ into $\mathcal{S}_0 = \{j \in \mathcal{S} : \hat{x}_j = 0\}, \mathcal{S}_1 = \{j \in \mathcal{S} : \hat{x}_j = 1\}$, such that $\mathcal{S} = \mathcal{S}_0 \dot{\cup} \mathcal{S}_1$. Specifically, the $K_0$ binary variables with the smallest $p_j$ form $\mathcal{S}_0$ and are predicted as zero, while the $K_1$ variables with the largest $p_j$ form $\mathcal{S}_1$ and are predicted as one (any overlap with $\mathcal{S}_0$ is excluded from $\mathcal{S}_1$). For the fractional radius $\Delta \in [0, 1]$ reported in Table 12, the implementation computes the absolute Hamming budget as

$$D = \lfloor \Delta|\mathcal{S}_0 \cup \mathcal{S}_1| \rfloor . \tag{10}$$

The backend then solves

$$\min_{\mathbf{x} \in \mathcal{X}(\mathcal{I})} f_{\mathcal{I}}(\mathbf{x}) \quad \text{s.t.} \quad \sum_{j \in \mathcal{S}_0} x_j + \sum_{j \in \mathcal{S}_1} (1 - x_j) \leq D, \tag{11}$$

which imposes a Hamming-distance budget over the selected partial assignment. Unlike direct fixing, a prediction may be overturned as long as the total number of disagreements does not exceed $D$.

The parameters that determine PS execution are `delta`, `backend`, and `time_limit`, together with `k0` and `k1`. Integer values of `k0`, `k1`, and `delta` denote absolute counts. Floating-point values in $[0, 1]$ denote ratios: `k0` and `k1` are normalized by $|\mathcal{B}|$, whereas `delta` is normalized by the number of variables included in the selected center. All resulting counts are floored. For Gurobi, `MIPFocus=1` is enabled to prioritize incumbent discovery.

### D.2 Neural Diving (ND)

ND (Nair et al., 2021) constructs several reduced MILPs by fixing subsets of integer variables. For each of `sample_times` trials, it independently samples a binary assignment from $[1 - p_j, p_j]$, uniformly selects $\max\{1, \lfloor r|\mathcal{B}| \rfloor\}$ variables with $r =$ `fixed_rate`, and enforces their sampled values by setting their lower and upper bounds equal. The backend optimizes every reduced MILP, and the best feasible solution according to $f_{\mathcal{I}}$ is returned.

The core parameters are `sample_times`, `fixed_rate`, `parallel`, `backend`, and `time_limit`. In serial mode, `time_limit` is a global wall-clock budget: before each trial, the remaining time is divided equally among the remaining subproblems. In parallel mode, subproblems are submitted concurrently and use the backend's own limit; because their wall-clock trajectories cannot be totally ordered, parallel ND is disabled when primal-trajectory recording is requested. Although the code identifies variables through the integer mask, its assignment sampler is binary and should not be used for general integer domains without an adapted sampler.

### D.3 Probabilistic Multi-Variable Branching (PMVB)

PMVB (Chen et al., 2026) converts marginal probabilities into aggregate cardinality restrictions rather than fixing every confident variable. Let $\mathcal{U} = \{j : p_j \geq \tau,\ p_j < \tau_f\}$ and $\mathcal{L} = \{j : 1 - p_j \geq \tau,\ 1 - p_j < \tau_f\}$, where $\tau =$ `tmvb` and $\tau_f =$ `fixthresh`. After excluding variables that are directly fixed, the implementation computes Hoeffding-style bounds

$$\xi_{\mathcal{U}} = \sum_{j \in \mathcal{U}} p_j - \sqrt{\frac{|\mathcal{U}|}{2} \log \frac{1}{1 - \rho_{\mathcal{U}}}}, \qquad \xi_{\mathcal{L}} = \sum_{j \in \mathcal{L}} p_j + \sqrt{\frac{|\mathcal{L}|}{2} \log \frac{1}{1 - \rho_{\mathcal{L}}}}, \tag{12}$$

and adds $\sum_{j \in \mathcal{U}} x_j \geq \lceil \xi_{\mathcal{U}} \rceil$ and $\sum_{j \in \mathcal{L}} x_j \leq \lfloor \xi_{\mathcal{L}} \rfloor$. Here $\rho_{\mathcal{U}} =$ `p_success_up` and $\rho_{\mathcal{L}} =$ `p_success_low`. The lower bound is clipped to $[0, |\mathcal{U}|]$, and the upper bound is lower-clipped at

zero, exactly as in the FMIP downstream implementation. Its numerical guard additionally caps $\xi_{\mathcal{U}}$ by $\lfloor |\mathcal{U}|/5 \rfloor$ when $1 - \rho_{\mathcal{U}} \leq 10^{-10}$ and sets $\xi_{\mathcal{L}} = 0$ when $1 - \rho_{\mathcal{L}} \geq 0.5$.

The parameter `fixratio` additionally fixes the lowest and highest $\lfloor |\mathcal{B}|$ `fixratio` $\rfloor$ variables, ranked by the margin $p_j - (1-p_j)$, to zero and one, respectively. When `ratio_involve=True`, $\mathcal{L}$ and $\mathcal{U}$ are instead formed from the lowest `ratio_low` and highest `ratio_up` fractions of $p_j$. The switches `low_cut` and `up_cut` independently enable the two cardinality cuts. The `score_type` parameter specifies whether the input is a logit vector, a probability vector, or a two-class matrix; multiple heatmaps are averaged in probability space. The remaining execution parameters are `backend` and `time_limit`. Following the FMIP configuration, we set `fixratio=0` and `fixthresh=1.1` by default, while we denote the core hyperparameters `tmvb` by $\tau$ and the shared success probability by $\delta := \rho_{\mathcal{U}} = \rho_{\mathcal{L}}$ with their values reported in Table 12. When `robust=True`, the Gurobi backend additionally sets `Cuts=1` and `MIPFocus=2`.

### D.4 Alternating Prediction-correction (Apollo)

Apollo-MILP (Liu et al., 2025) alternates neural prediction and solver-based correction. Our `ApolloEnforcer` follows the released FMIP downstream implementation: it uses agreement-based fixing and does not reproduce the full UEBO training mechanism of the original Apollo paper. At iteration $t$, the learner predicts the currently unfixed binary variables. Fractions $k_0^{(t)}$ and $k_1^{(t)}$ form a PS trust region, with radius $D^{(t)} = \lfloor \Delta^{(t)}(|\mathcal{S}_0^{(t)}| + |\mathcal{S}_1^{(t)}|) \rfloor$. After the correction subproblem is solved, every currently free binary variable whose rounded solver value agrees with the learner's full argmax prediction is permanently fixed. The reduced instance is then re-encoded and predicted again. The best feasible solution across all iterations is returned.

The execution is controlled by `schedule`, whose rows are $(k_0^{(t)}, k_1^{(t)}, \Delta^{(t)})$, and by `epochs` when a constant schedule is used. A callable `learner_predict_fn` is required for genuine multi-round inference; without it, the input scores are used only in the first round. `time_limits` may assign an explicit budget to each round. If only a scalar `time_limit` is supplied, it is divided equally across the configured rounds. Prediction time counts against the corresponding round and global budgets. If neither `time_limits` nor a scalar `time_limit` is supplied, the Enforcer inherits the backend limit and `per_iter_time_cap` bounds each individual correction solve. The remaining core parameters are `score_type` and `backend`. Without losing taxonomic consistency, this iterative process naturally extends the feedforward pipeline specified in Eq. 2 to a more extensible form:

$$G_{\mathcal{I}} \xrightarrow{\mathsf{L}_\theta} \mathbf{s} \xrightarrow{\mathsf{E}_\alpha(\mathsf{L}_\theta, G_{\mathcal{I}})} \left(\widetilde{\mathbf{x}}^0, \delta^0\right) \xrightarrow{\mathsf{R}_\beta} \widetilde{\mathbf{x}}. \tag{13}$$

## E Details of the Refiner Algorithms

**Common interface.** Unlike an Enforcer, a Refiner never consumes learner logits. It requires a complete feasible incumbent $\mathbf{x}^{(0)} \in \mathcal{X}(\mathcal{I})$ and attempts to produce a non-worse solution. Before refinement, our implementation explicitly validates dimension, finiteness, variable bounds, integrality, and all linear constraints within tolerance `tol`. An invalid incumbent raises an error rather than silently invoking a feasibility repair. All three Refiners retain $\mathbf{x}^{(0)}$ whenever the backend returns no solution or a worse candidate. Gurobi, SCIP, COPT, and CPLEX are supported, with `time_limit`, `mip_gap`, `threads`, and `seed` passed to the selected backend.

### E.1 Warm-start Solver (WS)

The `WarmStartSolverRefiner` is a solver-only baseline. It injects $\mathbf{x}^{(0)}$ through the native MIP-start mechanism and solves the original MILP without fixing variables, adding constraints, or modifying heuristic parameters. Thus, it directly measures the value of the incumbent under an otherwise standard exact-solver run. Its search behavior is determined by `backend`, `time_limit`, `mip_gap`, `threads`, and `seed`; `tol` affects only incumbent validation and the decision of whether an objective improvement is strict.

### E.2 Local Branching (LB)

The `LocalBranchingRefiner` implements the local-branching neighborhood of Fischetti & Lodi (2003). Let $\mathcal{B}_0 = \{j \in \mathcal{B} : x_j^{(0)} = 0\}$ and $\mathcal{B}_1 = \{j \in \mathcal{B} : x_j^{(0)} = 1\}$. For binary ILPs, the backend solves

$$\min_{\mathbf{x} \in \mathcal{X}(\mathcal{I})} f_{\mathcal{I}}(\mathbf{x}) \quad \text{s.t.} \quad \sum_{j \in \mathcal{B}_0} x_j + \sum_{j \in \mathcal{B}_1} (1 - x_j) \leq r. \tag{14}$$

For a general integer variable, the same budget includes $|x_j - x_j^{(0)}|$, linearized by a nonnegative auxiliary variable. Continuous variables remain unrestricted. The incumbent is also supplied as a MIP start.

Our implementation performs one exact solve inside a single neighborhood; it does not implement the original paper's iterative neighborhood reversal, diversification, or tabu phases. The key parameter `radius` specifies $r$: an integer is an absolute distance, while a float in $[0, 1]$ is multiplied by the number of integer variables and floored. Values larger than the number of integer variables are capped. The other execution parameters are the same backend controls as for warm-start refinement.

### E.3 Large Neighborhood Search (LNS) with Diverse Destroy Policies

Our `GenericLNSRefiner` class follows the generic destroy-and-repair semantics of large neighborhood search (Song et al., 2020; Huang et al., 2023). At iteration $t$, a destroy policy selects $\mathcal{D}^{(t)} \subseteq \mathcal{J}$, where $\mathcal{J}$ is the set of integer variables. Variables outside the destroy set are fixed to the current incumbent, defining

$$\mathcal{X}(\mathcal{D}^{(t)}; \mathbf{x}^{(t)}) = \{\mathbf{x} \in \mathcal{X}(\mathcal{I}) : x_j = x_j^{(t)},\ \forall j \in \mathcal{J} \setminus \mathcal{D}^{(t)}\}. \tag{15}$$

The exact backend repairs the partial assignment by optimizing over this set; all continuous variables and the destroyed integer variables remain free. The current incumbent is provided as a MIP start, and a candidate is accepted only if it is no worse than $\mathbf{x}^{(t)}$ up to `tol`.

The implemented learning-free destroy policies are:

- **`random`**: independently samples the requested number of integer variables without replacement at every iteration.
- **`random_partition`**: generates a random permutation of all integer variables and visits consecutive, non-overlapping blocks. Once the permutation is exhausted, or the adaptive neighborhood size changes, it reshuffles.
- **`graph_bfs`**: samples a random integer pivot and performs a breadth-first traversal on $G_{\mathcal{I}}$ through alternating variable and constraint nodes. Integer variables encountered by the traversal enter the destroy set; continuous variables may connect the traversal but are not selected. Disconnected shortfalls are filled by random unselected integers.

The `neighborhood_size` parameter is either an absolute count or a "destroy ratio" $k \in [0, 1]$; a positive ratio uses $\max\{1, \lceil k|\mathcal{J}| \rceil\}$. The global `time_limit` bounds the complete LNS wall-clock run, whereas `repair_time_limit` $\tau$ bounds each repair solve and is truncated by the remaining global budget. `max_iters` provides an alternative or additional stopping condition. At least one of `time_limit` and `max_iters` must be finite.

## F Details of the Datasets

### F.1 Problem Definitions and MILP Formulations

We present the problem descriptions and binary integer linear programming formulations for the four benchmark families used in this work: combinatorial auction (CA), maximum independent set (MIS), set covering (SC), and binary capacitated facility location (CFL). Throughout this section, we use $[n] = \{1, \ldots, n\}$ and present all objectives under a minimization convention. Therefore, maximization problems are equivalently written by negating their objectives.

**Combinatorial Auction (CA).** In the CA problem, we are given a set of $m$ items and $n$ bids $\{(B_i, p_i) : i \in [n]\}$, where $B_i \subseteq [m]$ denotes the bundle of items requested by bid $i$ and $p_i \geq 0$ is

its bidding price. The goal is to select a subset of mutually compatible bids so as to maximize the total accepted price, where each item can be allocated to at most one accepted bid. Let $x_i \in \{0,1\}$ indicate whether bid $i$ is accepted. The CA instance is formulated as

$$\begin{aligned} \min_{x} \quad & -\sum_{i\in[n]} p_i x_i \\ \text{s.t.} \quad & \sum_{i:\, j\in B_i} x_i \le 1, \quad \forall j \in [m], \\ & x_i \in \{0,1\}, \qquad \forall i \in [n]. \end{aligned} \tag{16}$$

**Maximum Independent Set (MIS).** In the MIS problem, we are given an undirected graph $G = (V, E)$. The goal is to select the largest subset of vertices such that no two selected vertices are adjacent. Let $x_v \in \{0,1\}$ indicate whether vertex $v \in V$ is selected. The unweighted MIS problem is formulated as

$$\begin{aligned} \min_{x} \quad & -\sum_{v\in V} x_v \\ \text{s.t.} \quad & x_u + x_v \le 1, \quad \forall (u,v) \in E, \\ & x_v \in \{0,1\}, \qquad \forall v \in V. \end{aligned} \tag{17}$$

The edge constraints enforce the independence condition by preventing the simultaneous selection of both endpoints of any edge.

**Set Covering (SC).** In the SC problem, we are given a universe of $m$ elements and $n$ candidate sets $\{\mathcal{S}_i : i \in [n]\}$, where $\mathcal{S}_i \subseteq [m]$ is the subset of elements covered by set $i$. Each set has a nonnegative selection cost $c_i$. The goal is to choose a minimum-cost collection of sets such that every element is covered by at least one selected set. Let $x_i \in \{0,1\}$ indicate whether set $i$ is selected. The SC problem is formulated as

$$\begin{aligned} \min_{x} \quad & \sum_{i\in[n]} c_i x_i \\ \text{s.t.} \quad & \sum_{i:\, j\in\mathcal{S}_i} x_i \ge 1, \quad \forall j \in [m], \\ & x_i \in \{0,1\}, \qquad \forall i \in [n]. \end{aligned} \tag{18}$$

Equivalently, with a binary coverage matrix $A \in \{0,1\}^{m\times n}$ where $A_{ji} = 1$ if set $i$ covers element $j$, the covering constraints can be written as $Ax \ge \mathbf{1}$.

**Binary Capacitated Facility Location (CFL).** In the binary CFL problem, we are given a set of customers $\mathcal{I}$ and a set of candidate facilities $\mathcal{J}$. Each customer $i \in \mathcal{I}$ has demand $d_i > 0$. Each facility $j \in \mathcal{J}$ has capacity $Q_j > 0$ and fixed opening cost $f_j \ge 0$. Assigning customer $i$ to facility $j$ incurs assignment cost $c_{ij} \ge 0$. The goal is to decide which facilities to open and how to assign customers to open facilities so as to minimize the total opening and assignment cost while respecting facility capacities. Let $y_j \in \{0,1\}$ indicate whether facility $j$ is opened, and let $x_{ij} \in \{0,1\}$ indicate whether customer $i$ is assigned to facility $j$. The binary CFL formulation used in our benchmark is

$$\begin{aligned} \min_{x,y} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}x_{ij} + \sum_{j\in\mathcal{J}} f_j y_j \\ \text{s.t.} \quad & \sum_{j\in\mathcal{J}} x_{ij} \ge 1, && \forall i \in \mathcal{I}, \\ & \sum_{i\in\mathcal{I}} d_i x_{ij} \le Q_j y_j, && \forall j \in \mathcal{J}, \\ & x_{ij} \le y_j, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\ & \sum_{j\in\mathcal{J}} Q_j y_j \ge \sum_{i\in\mathcal{I}} d_i, \\ & x_{ij} \in \{0,1\}, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\ & y_j \in \{0,1\}, && \forall j \in \mathcal{J}. \end{aligned} \tag{19}$$

The first set of constraints requires each customer to be served by at least one facility. Since assignment costs and demands are nonnegative, an optimal solution can be chosen in which each customer is assigned to exactly one facility. The second set of constraints enforces facility capacities and disables assignments to closed facilities. The linking constraints $x_{ij} \leq y_j$ explicitly prohibit assigning customers to unopened facilities. The total-capacity constraint is a valid strengthening retained in our generated instances.

**Balanced Item Placement (BIP).** The Balanced Item Placement problem originates from the NeurIPS 2021 ML4CO competition and models the balanced allocation of items, such as files or processes, across a collection of containers, such as disks or machines. Let $\mathcal{I}$ denote the set of items, $\mathcal{J}$ the set of containers, and $\mathcal{K}$ the set of resource dimensions. For each item $i \in \mathcal{I}$ and dimension $k \in \mathcal{K}$, let $a_{ik} \geq 0$ denote its resource consumption, $d_{ik} \geq 0$ its normalized contribution to the desired load, and $b_k > 0$ the capacity of each container in dimension $k$. Let $x_{ij} \in \{0, 1\}$ indicate whether item $i$ is placed in container $j$. The continuous variable $y_{jk} \geq 0$ measures the load deficit of container $j$ in dimension $k$, while $z_k \geq 0$ represents the maximum deficit across all containers in that dimension. With nonnegative penalty weights $\alpha_k$ and $\beta_k$, the formulation is

$$
\begin{aligned}
\min_{x,y,z} \quad & \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} \alpha_k y_{jk} + \sum_{k\in\mathcal{K}} \beta_k z_k \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}} x_{ij} = 1, && \forall i \in \mathcal{I}, \\
& \sum_{i\in\mathcal{I}} a_{ik} x_{ij} \leq b_k, && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}, \\
& \sum_{i\in\mathcal{I}} d_{ik} x_{ij} + y_{jk} \geq 1, && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}, \\
& y_{jk} \leq z_k, && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}, \\
& x_{ij} \in \{0, 1\}, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}, \\
& y_{jk} \geq 0,\ z_k \geq 0, && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}.
\end{aligned}
\tag{20}
$$

The assignment constraints place every item in exactly one container, while the multi-dimensional knapsack constraints enforce the container capacities. The last two families of constraints quantify both container-wise and worst-case imbalance, which are jointly penalized in the objective. Each released instance contains $|\mathcal{I}| = 105$ items, $|\mathcal{J}| = 10$ containers, and $|\mathcal{K}| = 3$ resource dimensions, resulting in $1{,}050$ binary placement variables and 33 continuous imbalance variables.

**Workload Apportionment (WA).** The Workload Apportionment problem, also introduced in the NeurIPS 2021 ML4CO competition, models the robust allocation of divisible workloads across a minimum number of active workers. Let $\mathcal{M}$ denote the set of tasks and $\mathcal{N}$ the set of workers. Each task $i \in \mathcal{M}$ has workload demand $a_i > 0$ and can only be assigned to a subset of accessible workers $\mathcal{N}_i \subseteq \mathcal{N}$. Each worker $j \in \mathcal{N}$ has capacity $b_j > 0$. Let $y_j \in \{0, 1\}$ indicate whether worker $j$ is activated, and let $x_{ij} \geq 0$ denote the amount of workload from task $i$ assigned to worker $j$. The allocation must remain sufficient after the failure of any single accessible worker. The resulting formulation is

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{j\in\mathcal{N}} y_j \\
\text{s.t.} \quad & x_{ij} \leq a_i y_j, && \forall i \in \mathcal{M},\ \forall j \in \mathcal{N}_i, \\
& \sum_{i\in\mathcal{M}:\, j\in\mathcal{N}_i} x_{ij} \leq b_j, && \forall j \in \mathcal{N}, \\
& \sum_{j\in\mathcal{N}_i\setminus\{j'\}} x_{ij} \geq a_i, && \forall i \in \mathcal{M},\ \forall j' \in \mathcal{N}_i, \\
& y_j \in \{0, 1\}, && \forall j \in \mathcal{N}, \\
& 0 \leq x_{ij} \leq b_j, && \forall i \in \mathcal{M},\ \forall j \in \mathcal{N}_i.
\end{aligned}
\tag{21}
$$

The linking constraints prevent workload from being assigned to inactive workers, and the capacity constraints bound the aggregate load processed by each worker. The third family of constraints requires the remaining allocation for every task to cover its full demand after removing any one

Table 8: Statistics of the training datasets and offline supervision assets. Ref. denotes one incumbent obtained with Gurobi-3600s, while $\mathcal{P}^+$ and $\mathcal{P}^-$ denote feasible positive and infeasible negative solution pools, respectively. Pool columns report the actual mean number of stored solutions per instance. †For one MIS instance, Gurobi returned only five positive solutions; its corresponding negative pool contains 50 perturbations.

| Dataset | Format | # Train | Avg. # Cons. | Avg. # Bin. | Avg. # Cont. | Ref./inst. | $\lvert\mathcal{P}^+\rvert$/inst. | $\lvert\mathcal{P}^-\rvert$/inst. |
|---|---|---|---|---|---|---|---|---|
| *Main binary-ILP training distributions* | | | | | | | | |
| SC | LP | 300 | 3,000.0 | 2,000 | 0 | 1 | 500.00 | 500.00 |
| CA | LP | 300 | 578.4 | 1,500 | 0 | 1 | 500.00 | 500.00 |
| MIS | LP | 300 | 5,941.5 | 1,500 | 0 | 1 | 498.35† | 498.50† |
| CFL | LP | 300 | 5,151.0 | 5,050 | 0 | 1 | 500.00 | 500.00 |

Table 9: Statistics of the test datasets used in ML4MILP-Bench.

| Dataset | Type | Sense | # Inst. | Avg. # Cons. | Avg. # Bin. | Avg. # Cont. | Mean BKS |
|---|---|---|---|---|---|---|---|
| *Main distributional evaluation* | | | | | | | |
| SC | ILP | Min | 100 | 3,000.0 | 2,000 | 0 | 86.43 |
| CA | ILP | Max | 100 | 578.8 | 1,500 | 0 | 22,272.55 |
| MIS | ILP | Max | 100 | 5,941.9 | 1,500 | 0 | 684.14 |
| CFL | ILP | Min | 100 | 5,151.0 | 5,050 | 0 | 11,388.79 |
| *Mixed-MILP generalization* | | | | | | | |
| WA | MILP | Min | 100 | 64,304.6 | 1,000 | 60,000 | 707.82 |
| BIP | MILP | Min | 100 | 195.0 | 1,050 | 33 | 12.63 |
| (Selected) MIPLIB | ILP | Min | 39 | 41,439.3 | 60,888.0 | 0 | 29,637,202.98 |

accessible worker, thereby enforcing robustness against an arbitrary single-worker failure. The objective minimizes the number of activated workers. In the released benchmark instances used in this work, each MILP contains approximately 1000 binary activation variables and 60000 continuous apportionment variables.

### F.2 Dataset Statistics

**Training Data.** Table 8 summarizes both the shared training corpora and the method-dependent supervision made available by our benchmark. Following Geng et al. (2025), the four main binary-ILP families contain 300 training instances each, while the mixed-MILP extensions use deterministic 1000-instance subsets of the WA and BIP training distributions. The single-reference labels are finite-budget (3600s) Gurobi incumbents.

Positive pools are collected once and shared across pool-supervised learners, using Gurobi solution-pool search with a 1000-second limit and a target of at most 500 feasible solutions per instance. Under the aligned training setting, PS-SL, CoCo-MILP, and ConPaS consume the first 50 positive solutions selected according to the stored pool order. Label-free learners use the same instances but ignore these offline supervision assets.

For ConPaS-Inf, $\mathcal{P}^-$ is constructed by perturbing the first 50 positive binary assignments with $\beta = 10$, targeting ten infeasible samples per source solution, or 500 negatives per instance. The reported pool sizes are measurements of the released files rather than nominal generation limits.

**Test Data.** Our ML4MILP-Bench combines four synthetic binary ILP families with two larger mixed-integer workloads. SC, CA, and MIS follow the DiffILO/ML4CO-style distributions, while CFL is regenerated with the binary formulation used by the diffusion baselines. WA and BIP serve as mixed-MILP generalization tests: unlike the main ILP suite, they contain continuous variables that must be completed by a solver-backed enforcer. All instances are stored in the unified LP/MPS interface, and each test instance has a corresponding reference solution for feasibility and BKS-relative primal-gap evaluation. For each dataset, the BKS entry in Table 9 is the arithmetic mean of the 100 reference objective values (obtained with Gurobi-3600s), rather than a single global optimum shared across instances.

# G Details of the Unified Backbone Architecture

**Overview.** Our unified backbone is developed upon the bipartite GNN predictor of DiffILO (Geng et al., 2025) for its adaptable and effective implementation. We retain its alternating variable-to-constraint and constraint-to-variable message passing, mean aggregation, graph-wise normalization, multi-layer feature concatenation, and variable-wise output interface, while reimplementing the architecture with sparse native PyTorch operations and extending it to support batched MILPs, mixed variable types, and time-conditioned generative learners. Given $G_{\mathcal{I}} = \Phi(\mathcal{I})$, the backbone produces one scalar score $s_j \in \mathbb{R}$ for each variable $v_j$. The architectural parameters are shared across all nodes and instances and are independent of the problem dimensions $m$ and $n$.

**Input Embedding.** Let $d_h$ be the hidden dimension and $L$ the number of message-passing rounds. Variable and constraint features are embedded independently as

$$\mathbf{h}_{v_j}^{(0)} = \rho_v(\mathbf{X}_{j,:}^{v}), \qquad \mathbf{h}_{c_i}^{(0)} = \rho_c(\mathbf{X}_{i,:}^{c}), \qquad \mathbf{e}_{ij} = \mathrm{BN}_e(\mathbf{X}_{(i,j),:}^{e}), \tag{22}$$

where $\rho_v$ and $\rho_c$ follow $\mathrm{BN} \to \mathrm{Linear} \to \mathrm{ReLU} \to \mathrm{Linear} \to \mathrm{ReLU}$ and map their inputs to $\mathbb{R}^{d_h}$. The signed scalar edge coefficient is batch-normalized but remains one-dimensional.

**Alternating Bipartite Message Passing.** At layer $\ell$, information is first propagated from variables to constraints. For every $(c_i, v_j) \in \mathcal{E}$, we compute

$$\mathbf{m}_{j\to i}^{(\ell)} = \phi_{v\to c}^{(\ell)}\left(\mathbf{h}_{v_j}^{(\ell-1)} \| \mathbf{e}_{ij} \| \mathbf{h}_{c_i}^{(\ell-1)}\right), \tag{23}$$

where $\phi^{(\ell)}$ is a two-layer MLP and $\|$ denotes concatenation. Messages are averaged over the incident variables and combined with the previous constraint representation:

$$\mathbf{a}_{c_i}^{(\ell)} = \mathrm{BN}_{v\to c}^{(\ell)}\left(\frac{1}{|\mathcal{N}(c_i)|}\sum_{j\in\mathcal{N}(c_i)} \mathbf{m}_{j\to i}^{(\ell)}\right), \qquad \widetilde{\mathbf{h}}_{c_i}^{(\ell)} = \psi_{v\to c}^{(\ell)}\left(\mathbf{a}_{c_i}^{(\ell)} \| \mathbf{h}_{c_i}^{(\ell-1)}\right). \tag{24}$$

The reverse constraint-to-variable pass applies the same construction using the updated constraint embeddings:

$$\mathbf{m}_{i\to j}^{(\ell)} = \phi_{c\to v}^{(\ell)}\left(\mathbf{h}_{c_i}^{(\ell)} \| \mathbf{e}_{ij} \| \mathbf{h}_{v_j}^{(\ell-1)}\right), \qquad \widetilde{\mathbf{h}}_{v_j}^{(\ell)} = \psi_{c\to v}^{(\ell)}\left(\mathbf{a}_{v_j}^{(\ell)} \| \mathbf{h}_{v_j}^{(\ell-1)}\right), \tag{25}$$

where $\mathbf{a}_{v_j}^{(\ell)}$ is the batch-normalized mean of incoming constraint messages. Mean aggregation avoids uncontrolled scaling with node degree, while degree information remains explicitly available in the input features.

**Graph-wise Normalization.** After each directional pass, GraphNorm is applied separately to the variable and constraint nodes of every instance in a batch. For a node representation $\mathbf{h}_u$ in graph $g$, it computes $\mathrm{GN}(\mathbf{h}_u) = \boldsymbol{\gamma} \odot (\mathbf{h}_u - \boldsymbol{\alpha} \odot \boldsymbol{\mu}_g)/\sqrt{\boldsymbol{\nu}_g + \epsilon} + \boldsymbol{\beta}$, where $\boldsymbol{\mu}_g$ and $\boldsymbol{\nu}_g$ are the per-channel graph mean and variance, and $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\beta}$ are learnable. Consequently, the final embeddings for variable and constraint nodes are

$$\mathbf{h}_{c_i}^{(\ell)} = \mathrm{GN}_c^{(\ell)}(\widetilde{\mathbf{h}}_{c_i}^{(\ell)}), \quad \mathbf{h}_{v_j}^{(\ell)} = \mathrm{GN}_v^{(\ell)}(\widetilde{\mathbf{h}}_{v_j}^{(\ell)}). \tag{26}$$

**Multi-scale Variable Readout.** Following DiffILO (Geng et al., 2025), representations from the input embedding and all message-passing layers are concatenated: $\overline{\mathbf{h}}_{v_j} = \mathbf{h}_{v_j}^{(0)} \| \mathbf{h}_{v_j}^{(1)} \| \cdots \| \mathbf{h}_{v_j}^{(L)} \in \mathbb{R}^{(L+1)d_h}$. A shared MLP with dimensions $(L+1)d_h \to 512 \to 128 \to 1$ then produces $s_j$. For binary variables, $p_j = \sigma(s_j)$ is interpreted as the predicted probability of $x_j = 1$, equivalently using two-class logits $(0, s_j)$. Different learners share this output contract while applying their own supervised, contrastive, unsupervised, flow-matching, or diffusion objectives.

**Dynamic Conditioning and Mixed MILPs.** For FMIP and DDIM, the diffusion- or flow-based noisy or intermediate state is appended as one additional variable feature, and a sinusoidal embedding of the graph-level timestep is projected and added to each variable representation after every message-passing round. The timestep is broadcast only within its corresponding graph. For mixed binary-continuous MILPs, the common backbone still propagates information through all variable and constraint nodes, whereas training losses and downstream scores are restricted to the integer-variable

mask; continuous variables are completed by solver-backed Enforcers. The implementation additionally provides an optional split-variable extension with separate integer and continuous embeddings and gated aggregation into constraint nodes, but this is not used by the default aligned backbone.

**Computational Properties.** Instances are batched as disjoint sparse graphs, and all aggregations are implemented by scatter operations without constructing a dense constraint matrix. For fixed $d_h$, each message-passing round is linear in $m + n + \mathrm{nnz}(\mathbf{A})$, and the model parameters do not depend on instance size. In the absence of optional positional features, the backbone is equivariant to independent permutations of variable and constraint indices.

# H Details of Training and Evaluation Settings

## H.1 Common Protocol

**Software.** Main development and experiments are implemented in Python 3.11.14 with PyTorch and PyTorch-Lightning v2.1.0. We adopt the latest versions of solver backends: Gurobi-13.0.1 (Nov. 2025), SCIP-10.0 (Nov. 2025), COPT-8.0.3 (Jan. 2026), through their corresponding Python APIs.

**Hardware.** All experiments are conducted on a single NVIDIA RTX 4090 GPU with an AMD EPYC 7302 16-core CPU and 128 GB memory.

**Common Settings.** Table 10 summarizes the settings shared across the four main binary-ILP datasets. Unless otherwise stated, slash-separated task-specific values throughout this appendix follow the order SC/MIS/CA/CFL.

Table 10: Common training and evaluation settings for the main binary-ILP suite.

| Item | Setting |
|---|---|
| Reference solution | Gurobi, 3600s per instance |
| Graph representation | $d_v = 8$, $d_c = 5$, $d_e = 1$; instance-wise normalization |
| Hidden dimension | 32, whenever the shared GNN backbone applies |
| Message-passing layers | 6/10/10/10 |
| Batch size | 5 |
| Validation samples | 500 hard Gumbel samples, temperature 1 |
| Deterministic decoding | Threshold 0.5 |
| Feasibility tolerance | $10^{-5}$ |
| Checkpoint selection | Feasibility first; feasible objective quality second |
| Test seed | 0 |
| Solver threads | 1 per run |

The shared architecture is applied only when compatible with the original method; method-defining modules, such as diffusion/flow dynamics and OPTFM pretraining, are retained. Checkpoint selection is performed without solver-backed Enforcers. OPTFM selects the pretrained encoder by validation F1 before fitting its downstream GBDT head. Solver versions are reported separately in Table 6.

For learner-only testing, no exact solver is invoked after neural inference. Rounding uses threshold 0.5, while sampling draws 1000 independent Bernoulli candidates with temperature 1 and seed 0. The raw-prediction metrics are defined in Sec. 3.1.

## H.2 Learner-specific Settings

All Learners consume the unified representation unless otherwise specified. Base-SL uses one reference solution; PS-SL, ConPaS, and CoCo-MILP share the same feasible solution-pool assets, with ConPaS additionally using generated infeasible candidates. DiffILO is trained without reference-solution supervision, while DDIM and FMIP retain their generative training mechanisms. OPTFM follows its separate self-supervised pretraining and frozen-head pipeline. We use the stated epoch counts as training caps rather than introducing an additional early-stopping patience hyperparameter.

For PS-SL, i.e., the trainable component in Han et al. (2023), the objective normalizers follow SC/MIS/CA/CFL as 100/-100/-1000/100. For ConPaS, the infeasible pool is generated by perturbing feasible assignments with perturbation factor $\beta = 10$, with at most 500 negative candidates per

Table 11: Learner-specific training configurations. $\eta$ denotes the initial learning rate, $E$ the maximum number of epochs, and WD the weight decay. Slash-separated entries follow SC/MIS/CA/CFL.

| Learner | $\eta$ | $E$ | WD | Method-specific parameters or settings |
|---|---|---|---|---|
| Base-SL | $10^{-3}/8\times10^{-5}/10^{-3}/10^{-3}$ | 1200/1200/2400/1200 | 0 | Single-reference BCE |
| PS-SL | $10^{-3}/8\times10^{-5}/10^{-3}/10^{-3}$ | 1200/1200/2400/1200 | 0 | Positive pool $K^+ = 50$; objective scales 100/-100/-1000/100 |
| ConPaS | $10^{-5}$ | 300 | $10^{-4}$ | $K^+ = 50$; $\tau^+ = \tau^- = 1$; minimum quality weight 0.05; BCE weight 0 |
| CoCo-MILP | $10^{-4}/10^{-4}/10^{-4}/10^{-3}$ | 1200/1200/2400/1200 | 0 | $K = 50$; $\tau = 0.1$; margin 0.9; ranking weight 0.01; ICC/MSCL enabled |
| DiffILO | $10^{-4}/8\times10^{-5}/8\times10^{-5}/10^{-4}$ | 1200/1200/2400/2400 | 0 | 15 hard Gumbel samples; temperature 1; cosine $T_{\max} = 200$ |
| DDIM | $2\times10^{-4}$ | 1200 | $10^{-4}$ | 1000 diffusion steps; 50 DDIM steps; linear noise; cosine inference schedule |
| FMIP | $2\times10^{-4}$ | 1200/1200/2400/2400 | $10^{-4}$ | 50 flow steps; cosine schedule; time-conditioned GNN |
| OPTFM | $2\times10^{-4}$ | 100 | 0 | Mask ratio 0.1; negative ratio 1; subgraph cap 20000; decoder width 128 |

instance. CoCo-MILP uses objective normalizers 100/-100/-10000/10000 and enables both multi-solution contrastive learning and ranking.

DiffILO uses normalized loss with Adam (Kingma & Ba, 2015), separate inner/output parameter groups, and the task-specific penalty schedules $(\mu_0, \Delta\mu) = (5, 0.01)/(100, 1)/(15, 0.001)/(30, 1)$. DDIM uses a linear forward noise schedule from $10^{-4}$ to $2 \times 10^{-2}$, unit BCE and denoising-MSE weights, and disables additional objective/constraint guidance in the aligned learner comparison. FMIP uses a binary categorical head for the main ILP suite[3]. For OPTFM, the pretrained encoder uses one attention head, $\beta = 0.5$, and zero dropout; the downstream GBDT uses 20 trees, learning rate 0.1, maximum depth 15, and minimum split size 2.

### H.3 Enforcer-specific Settings

Solver-free Enforcers consume only Learner scores. Solver-backed Enforcers operate on the original MILP and use the scores only to determine fixing, branching, or correction decisions; variables not selected by an Enforcer remain free for the backend. Parameter semantics follow Appendix D.

Table 12: Main hyperparameters of solver-backed Enforcers.

| Task \ Enforcer | ND $(K, \alpha)$ | PS $(k_0, k_1, \Delta)$ | PMVB $(\delta, \tau)$ | Apollo $(k_0, k_1, \Delta, K)$ |
|---|---|---|---|---|
| SC | (50, 0.1) | (0.3, 0.04, 0.2) | (0.7, 0.9) | (0.3, 0.04, 0.2, 2) |
| MIS | (50, 0.1) | (0.3, 0.2, 0.3) | (0.7, 0.9) | (0.3, 0.2, 0.3, 2) |
| CA | (50, 0.1) | (0.3, 0.06, 0.3) | (0.7, 0.9) | (0.3, 0.06, 0.3, 2) |
| CFL | (50, 0.1) | (0.04, 0.04, 0.1) | (0.7, 0.9) | (0.04, 0.04, 0.1, 2) |

The exact-solver backend and time limit are treated as independent experimental factors rather than Enforcer hyperparameters. Thus, the same Enforcer can be paired with different backends and stage-local budgets without changing its Learner input. The reported Enforcer time includes neural inference and solver-backed completion. For the generalization study in Appendix I.1, we set the $(k_0, k_1, \Delta)$ as (0.0, 0.04, 0.5), (0.3, 0.04, 0.2), and (0.3, 0.04, 0.2) for the `PS` Enforcer for WA, BIP, and MIPLIB instances, respectively.

### H.4 Refiner-specific Settings

**Initial Incumbent Acquisition.** Refiners receive only a complete feasible incumbent $\mathbf{x}^0$ and never access Learner logits. To isolate post-feasibility improvement, we cache one feasible $\mathbf{x}^0$ for every learner-instance pair and provide the identical incumbent to all compared Refiners. For SC, MIS, and CA, the cache is generated by PS with a 1s Gurobi budget; for CFL, we use ND with the same 1s budget. Each cached incumbent is verified against variable bounds, integrality, and all linear constraints before refinement.

**Baseline.** A Gurobi run from scratch with the same thread count, random seed, and 1000s budget serves as the solver-only reference.

[3]While the original formulation of FMIP accommodates continuous regression flows, to keep comparable behavior among the Learners within our benchmark's standardized evaluation, FMIP is instantiated strictly as a discrete generative prior over integer variables, delegating continuous variables to solver-backed completion.

Table 13: Refiner configurations in the main study. All settings use one thread, seed 0, and a stage-local limit of 1000s. These settings remain consistent across tasks.

| Refiner | Parameter | Value |
|---|---|---|
| Warm-start Solver (`WS`) | Neighborhood restriction | None |
| Local Branching (`LB`) | Radius ($r$) | 0.2 |
| Local Branching (`LB`) | Number of neighborhood solves | 1 |
| LNS-Random (`LNS-Rand`) | Destroy ratio ($k$) | 0.5 |
| LNS-Random (`LNS-Rand`) | Repair limit ($\tau$) | 10s |
| LNS-Partition (`LNS-Part`) | Destroy ratio ($k$) | 0.5 |
| LNS-Partition (`LNS-Part`) | Repair limit ($\tau$) | 10s |
| LNS-BFS (`LNS-BFS`) | Destroy ratio ($k$) | 0.5 |
| LNS-BFS (`LNS-BFS`) | Repair limit ($\tau$) | 10s |

WS passes $\mathbf{x}^0$ directly as a native MIP start and otherwise leaves the original MILP unrestricted. LB performs one exact solve within the prescribed Hamming neighborhood. Generic LNS iteratively releases the selected integer variables while fixing the remaining integer variables to the current incumbent; continuous variables remain free during repair. The framework additionally supports partition-based, and graph-BFS-based destroy policies, while the controlled main comparison uses random destroy.

# I Additional Empirical Results

## I.1 Generalization Results on Mixed-integer and Real-world Instances

**Protocol.** We first evaluate out-of-distribution generalization on **WA and BIP** problems. All Learners use checkpoints trained on SC without fine-tuning and share the same `PS` (`Gurobi-1000s`) Enforcer. Following the mixed-variable convention (Han et al., 2023; Pu et al., 2026), the Learner scores only binary variables; continuous variables remain free and are completed by the Enforcer. The SC-trained checkpoints are again applied directly without fine-tuning to selected **MIPLIB** instances. We restrict this set to pure binary instances so that every decision variable is exposed to the Learner, enabling a cleaner assessment of cross-domain predictive generalization without confounding solver-only completion of continuous variables. Notably, even under this controlled setting, our results show that a strong external solver can substantially compress Learner-dependent differences. Thus, extending the evaluation to the general and quite complex mixed-integer instances from MIPLIB, where continuous-variable completion further complicates stage-wise attribution, is of less meaning at current stage and hence left for future work. Among the 40 instances satisfying our selection criteria, `supportcase22` is omitted because `Gurobi-1000s` finds no incumbent, leaving 39 reported instances. These instances span 214 to 364,417 variables, 64 to 188,879 constraints (under our unified graph representation, equality constraints are converted into pairs of opposite inequalities), and 12,528 to 4,898,304 nonzero coefficients, covering broad variations in scale and sparsity. Gaps are computed against the official reference objectives with $\epsilon = 1$ to avoid near-zero denominators, i.e., $\frac{\mathbf{c}^\top \mathbf{x} - \mathbf{c}^\top \mathbf{x}^{\text{ref}}}{\mathbf{c}^\top \mathbf{x}^{\text{ref}} + 1} \times 100\%$ for minimization problems.

Table 14: Results on WA and BIP. All learners are equipped with a consistent `PS(Gurobi-1000s)` Enforcer. We report the primal gaps (%) at 10/100/1000 seconds (G@10,$10^2$,$10^3$), the primal integral over the 1000s horizon, and the final objective value.

| Learner | Enforcer | **WA** (100 inst.) G@10 | G@$10^2$ | G@$10^3$ | Obj↓. | PI↓ | **BIP** (100 inst.) G@10 | G@$10^2$ | G@$10^3$ | Obj↓. | PI↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Gurobi 3600s | | – | – | – | 707.820* | – | – | – | – | 12.633* | – |
| Gurobi 1000s | | 13.885 | 0.040 | 0.013 | 707.910 | 648.971 | 61.856 | 26.023 | 6.881 | 13.294 | 15071.344 |
| Base-SL | `PS` | 23.938 | 0.101 | **0.036** | **708.070** | 879.283 | 28.710 | 8.653 | -0.730 | 12.532 | 6664.295 |
| PS-SL | `PS` | 23.847 | 0.102 | **0.036** | **708.070** | 889.061 | 27.475 | 8.708 | -0.767 | 12.530 | 6561.658 |
| ConPaS | `PS` | **19.759** | 2.934 | 2.871 | 728.080 | 3542.961 | 34.889 | 13.154 | 0.306 | 12.579 | 8078.768 |
| CoCo | `PS` | 25.804 | 0.102 | **0.036** | **708.070** | 895.290 | **27.237** | **8.624** | -0.751 | 12.534 | 6557.448 |
| OPTFM | `PS` | 22.978 | 0.176 | 0.126 | 708.700 | 915.714 | 27.402 | 8.677 | -0.763 | 12.530 | 6549.601 |
| DiffILO | `PS` | 24.821 | 0.099 | **0.036** | **708.070** | **876.040** | 27.383 | 8.704 | -0.710 | 12.538 | 6570.207 |
| DDIM | `PS` | 22.959 | 0.104 | **0.036** | **708.070** | 890.190 | 29.868 | 7.425 | **-1.829** | **12.427** | **6194.450** |
| FMIP | `PS` | 24.924 | **0.099** | 0.037 | 708.080 | 885.509 | 31.053 | 9.490 | -0.645 | 12.547 | 7114.672 |

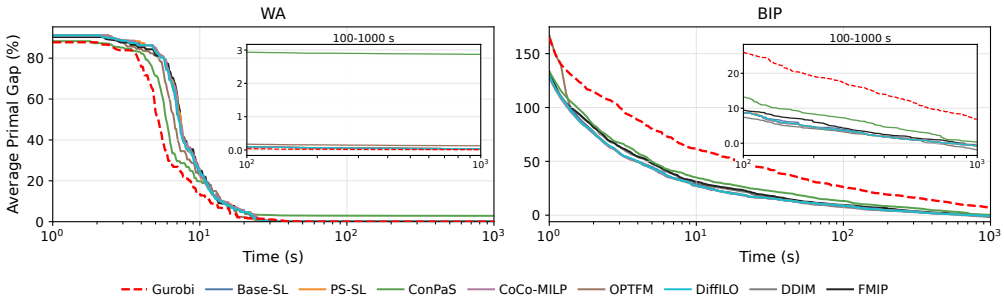



Figure 6: Primal gap curves on WA and BIP.

Table 15: Per-instance results on 39 selected MIPLIB instances, **BKS** denotes the official reference objective $\mathbf{c}^\top \mathbf{x}^{\text{ref}}$. The blue number in parentheses denotes the number of feasible instances out of the 39 reported instances.

| Instance | BKS↓ | Gurobi-1000s | | Base-SL | | CoCo-MILP | | DiffILO | | OPTFM | | FMIP | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Obj | Gap | Obj | Gap | Obj | Gap | Obj | Gap | Obj | Gap | Obj | Gap |
| academictimetablesmall | 0.000 | 1.000 | 100.000% | 1.000 | 100.000% | 1.000 | 100.000% | 5.000 | 500.000% | 12.000 | 1200.000% | 0.000 | 0.000% |
| air05 | 26374.000 | 26374.000 | 0.000% | 26374.000 | 0.000% | 26374.000 | 0.000% | 26374.000 | 0.000% | 26374.000 | 0.000% | 26374.000 | 0.000% |
| bab2 | -357544.312 | -357531.493 | 0.004% | -357544.312 | 0.000% | -357544.312 | 0.000% | -357542.784 | 0.000% | -357544.314 | 0.000% | -357542.793 | 0.000% |
| bab6 | -284248.231 | -284248.233 | 0.000% | -284248.233 | 0.000% | -284248.233 | 0.000% | -284248.234 | 0.000% | -284248.239 | 0.000% | -284248.233 | 0.000% |
| bnatt400 | 1.000 | 1.000 | 0.000% | (0.000) | N/A | 1.000 | 0.000% | 1.000 | 0.000% | (0.000) | N/A | (0.000) | N/A |
| chromaticindex1024-7 | 4.000 | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% |
| chromaticindex512-7 | 4.000 | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% | 4.000 | 0.000% |
| co-100 | 2639942.060 | 2639942.060 | 0.000% | 2639942.060 | 0.000% | 2639942.060 | 0.000% | 2639942.060 | 0.000% | 2639942.060 | 0.000% | 2639942.060 | 0.000% |
| cod105 | -12.000 | -12.000 | 0.000% | -12.000 | 0.000% | -12.000 | 0.000% | -12.000 | 0.000% | -12.000 | 0.000% | -12.000 | 0.000% |
| cvs16r128-89 | -97.000 | -96.000 | 1.020% | -97.000 | 0.000% | -97.000 | 0.000% | -97.000 | 0.000% | -97.000 | 0.000% | -97.000 | 0.000% |
| decomp2 | -160.000 | -160.000 | 0.000% | -160.000 | 0.000% | -160.000 | 0.000% | -160.000 | 0.000% | -160.000 | 0.000% | -160.000 | 0.000% |
| eil33-2 | 934.008 | 934.008 | 0.000% | 934.008 | 0.000% | 934.008 | 0.000% | 934.008 | 0.000% | 934.008 | 0.000% | 934.008 | 0.000% |
| eilA101-2 | 880.920 | 923.006 | 4.772% | 923.006 | 4.772% | 904.680 | 2.694% | 904.680 | 2.694% | 923.006 | 4.772% | 880.920 | 0.000% |
| fast0507 | 174.000 | 174.000 | 0.000% | 174.000 | 0.000% | 174.000 | 0.000% | 174.000 | 0.000% | 174.000 | 0.000% | 174.000 | 0.000% |
| glass-sc | 23.000 | 23.000 | 0.000% | 23.000 | 0.000% | 23.000 | 0.000% | 23.000 | 0.000% | 23.000 | 0.000% | 23.000 | 0.000% |
| graph20-20-1rand | -9.000 | -9.000 | 0.000% | -9.000 | 0.000% | -9.000 | 0.000% | -9.000 | 0.000% | -9.000 | 0.000% | -9.000 | 0.000% |
| irp | 12159.493 | 12159.493 | 0.000% | 12159.493 | 0.000% | 12159.493 | 0.000% | 12159.493 | 0.000% | 12159.493 | 0.000% | 12159.493 | 0.000% |
| n2seq36q | 52200.000 | 52200.000 | 0.000% | 52200.000 | 0.000% | 52200.000 | 0.000% | 52200.000 | 0.000% | 52200.000 | 0.000% | 52200.000 | 0.000% |
| n3div36 | 130800.000 | 130800.000 | 0.000% | 130800.000 | 0.000% | 130800.000 | 0.000% | 130800.000 | 0.000% | 130800.000 | 0.000% | 130800.000 | 0.000% |
| neos-3555904-turama | -34.700 | -34.700 | 0.000% | -34.700 | 0.000% | -34.700 | 0.000% | -34.700 | 0.000% | -34.700 | 0.000% | -34.700 | 0.000% |
| neos-5052403-cygnet | 182.000 | 182.000 | 0.000% | 182.000 | 0.000% | 182.000 | 0.000% | 182.000 | 0.000% | 182.000 | 0.000% | 182.000 | 0.000% |
| neos-631710 | 203.000 | 203.000 | 0.000% | 203.000 | 0.000% | 203.000 | 0.000% | 203.000 | 0.000% | 203.000 | 0.000% | 203.000 | 0.000% |
| neos-787933 | 30.000 | 30.000 | 0.000% | 30.000 | 0.000% | 30.000 | 0.000% | 30.000 | 0.000% | 30.000 | 0.000% | 30.000 | 0.000% |
| neos-957323 | -237.757 | -237.757 | 0.000% | -237.757 | 0.000% | -237.757 | 0.000% | -237.757 | 0.000% | -237.757 | 0.000% | -237.757 | 0.000% |
| neos-960392 | -238.000 | -238.000 | 0.000% | -238.000 | 0.000% | -238.000 | 0.000% | -238.000 | 0.000% | -238.000 | 0.000% | -238.000 | 0.000% |
| netdiversion | 242.000 | 242.000 | 0.000% | 242.000 | 0.000% | 289.000 | 19.342% | 242.000 | 0.000% | 292.000 | 20.576% | 242.000 | 0.000% |
| nw04 | 16862.000 | 16862.000 | 0.000% | 16862.000 | 0.000% | 16862.000 | 0.000% | 16862.000 | 0.000% | 16862.000 | 0.000% | 16862.000 | 0.000% |
| opm2-z10-s4 | -33269.000 | -33269.000 | 0.000% | -27606.000 | 17.021% | -27606.000 | 17.021% | -27660.000 | 16.859% | -27606.000 | 17.021% | -27606.000 | 17.021% |
| peg-solitaire-a3 | 1.000 | 1.000 | 0.000% | 1.000 | 0.000% | 1.000 | 0.000% | 1.000 | 0.000% | (0.000) | N/A | 1.000 | 0.000% |
| qap10 | 340.000 | 340.000 | 0.000% | 340.000 | 0.000% | 340.000 | 0.000% | 340.000 | 0.000% | 340.000 | 0.000% | 340.000 | 0.000% |
| rail01 | -70.570 | -70.570 | 0.000% | -70.570 | 0.000% | -70.570 | 0.000% | -70.570 | 0.000% | -70.570 | 0.000% | -70.570 | 0.000% |
| rail02 | -200.450 | -198.517 | 0.960% | -198.394 | 1.020% | (0.000) | N/A | -198.383 | 1.026% | -198.400 | 1.018% | -198.394 | 1.020% |
| reblock115 | -36800603.233 | -36800603.233 | 0.000% | -10520356.022 | 71.413% | -10520356.022 | 71.413% | -20774374.158 | 43.549% | -10520356.022 | 71.413% | -10520356.022 | 71.413% |
| s100 | -0.170 | -0.170 | 0.005% | -0.170 | 0.012% | -0.169 | 0.032% | -0.169 | 0.053% | -0.169 | 0.056% | -0.166 | 0.331% |
| sorrell3 | -16.000 | -16.000 | 0.000% | -16.000 | 0.000% | -16.000 | 0.000% | -16.000 | 0.000% | -16.000 | 0.000% | -16.000 | 0.000% |
| sp97ar | 660705645.759 | 660705645.760 | 0.000% | 662755562.880 | 0.310% | 661220812.160 | 0.078% | 661270366.720 | 0.085% | 661493955.200 | 0.119% | 661699053.760 | 0.150% |
| sp98ar | 529740623.200 | 529740623.200 | 0.000% | 529740623.200 | 0.000% | 529740623.200 | 0.000% | 529740623.200 | 0.000% | 529740623.200 | 0.000% | 529740623.200 | 0.000% |
| supportcase10 | 7.000 | 8.000 | 12.500% | (0.000) | N/A | (0.000) | N/A | (0.000) | N/A | (0.000) | N/A | (0.000) | N/A |
| tbfp-network | 24.163 | 24.163 | 0.000% | 24.163 | 0.000% | 24.163 | 0.000% | 24.163 | 0.000% | 24.163 | 0.000% | 24.163 | 0.000% |
| **Mean** | **29637202.979** | **29637204.513** | **3.058%** | **32005048.126** | **5.258%**(37) | **31963574.541** | **5.691%**(37) | **30853881.594** | **14.849%**(38) | **32859034.249** | **36.527%**(36) | **31976492.702** | **2.431%**(37) |

**Results on BIP and WA.** Table 14 reports the primal gaps at 10, 100, and 1000 seconds, the final objective, and the 1000-second primal integral, while Fig. 6 shows the corresponding gap-time trajectories. This protocol introduces substantial distribution and formulation shifts: WA contains approximately 1000 binary activation variables and 60000 continuous allocation variables per instance, whereas BIP contains 1050 binary variables and 33 continuous variables.

The two tasks exhibit markedly different transfer behavior. **1) On WA**, `Gurobi-1000s` remains strongest. Although ConPaS gives the best learned result at 10 seconds, its progress subsequently stalls at 2.871%. In contrast, the other Learner-`PS` combinations are repaired toward a narrow final range of 0.036-0.126% (PIs between 876.040 and 915.714). The convergence of substantially different predictions to similar final solutions indicates that, on WA, solver-backed completion largely determines long-horizon quality and can correct, but does not consistently benefit from, transferred neural guidance. **2) BIP** displays more favorable transfer. Every Learner-guided pipeline substantially improves upon `Gurobi-1000s`: gaps decrease from 61.856% to 27.237-34.889% at 10 seconds and from 26.023% to 7.425-13.154% at 100 seconds, reducing the PI by approximately 46-59%. ***DDIM performs best overall***, reaching a final gap of -1.829%. Here, a negative gap is valid because the reported BKS is the finite-budget `Gurobi-3600s` incumbent rather than a certificate of optimality; several learned pipelines find better incumbents. Nevertheless, the relatively concentrated trajectories across Learners show that ***most of the robust gain comes from how the Enforcer converts transferred scores into solver interventions***, rather than from a consistently preserved ranking of raw Learner quality. Together, WA and BIP show that cross-task predictions can provide useful initialization, but their utility remains strongly dependent on compatibility with the downstream Enforcer.

**Per-instance Results on MIPLIB Instances.** Table 15 reports objectives and BKS-relative gaps for 39 selected binary MIPLIB instances under the same `PS (Gurobi-1000s)` Enforcer. The reported mean gap is taken over instances for which a feasible incumbent is obtained. The results reveal a substantial feasibility-quality trade-off under real-world structural shift. `Gurobi-1000s` produces incumbents for all 39 reported instances with a mean gap of 3.058%. Among the transferred pipelines, ***FMIP attains the lowest mean gap*** of 2.431% over 37 feasible instances, while ***DiffILO obtains the widest coverage***, solving 38 instances but with a mean gap of 14.849%. Base-SL and CoCo-MILP solve 37 instances with mean gaps of 5.258% and 5.691%, respectively, whereas OPTFM solves 36 with a mean gap of 36.527%. The lower FMIP mean should therefore not be interpreted as uniform dominance over Gurobi, since it excludes two instances for which no feasible incumbent is found.

The per-instance results further expose highly heterogeneous transfer. Neural guidance is beneficial on cases such as `academictimetablesmall` and `eilA101-2`, where FMIP closes gaps left by the solver-only baseline. Conversely, all transferred pipelines deteriorate substantially on `opm2-z10-s4` and `reblock115`, and none obtains a feasible solution for `supportcase10`, for which Gurobi does. On many remaining instances, different Learners converge to the same zero-gap solution under the shared Enforcer. These common successes and failures suggest that ***current Learners provide useful but fragile out-of-distribution signals, while feasibility recovery and final solution quality are still governed predominantly by solver-backed completion***. Accordingly, evaluating MILP generalization requires reporting feasible coverage, anytime performance, and per-instance tail behavior in addition to aggregate gaps, and motivates closer co-design of transferable prediction semantics with the Enforcer that consumes them.

### I.2 Training Dynamics for the Learners

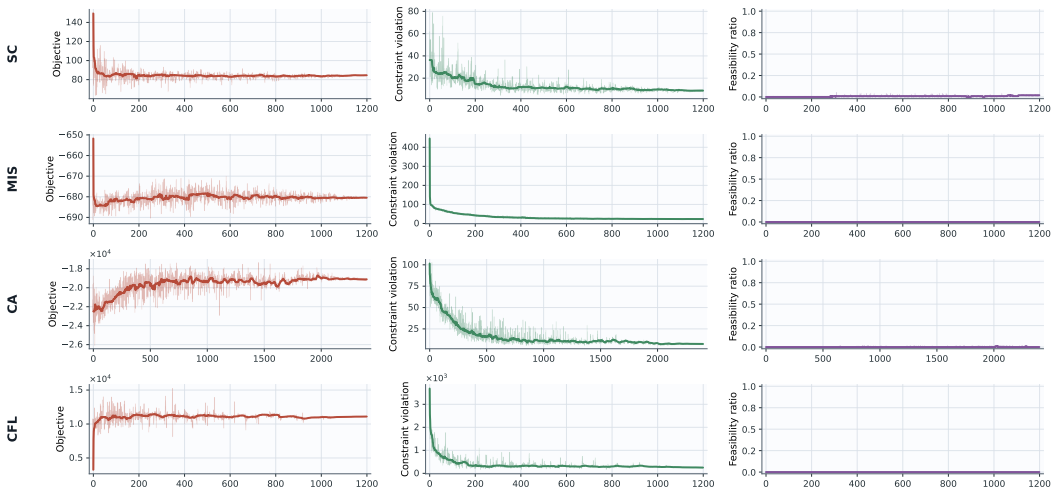



Figure 7: Training dynamics for vanilla supervised learning (SL).

Figures 7, 8, 9, 10, 11, 12, 13, and 14 demonstrate the core training dynamics (e.g., objective value, constraint violation, instance-wise feasibility rate) upon validation set during training of the Learners.

**Supervised Objectives.** Figures 7 and 8 show that pointwise supervised Learners rapidly stabilize their validation objectives and reduce aggregate constraint violations, while their instance-wise feasibility remains close to zero. Replacing a single reference solution with objective-weighted solution-pool marginals in PS-SL does not fundamentally resolve this behavior. Both objectives supervise variables independently and therefore do not retain all dependencies contained in the original feasible assignments. The structured supervised objectives provide stronger, but still task-dependent, feasibility signals. ConPaS contrasts complete positive and negative solutions and consequently reaches high feasibility on SC and CA and improves progressively on MIS, as shown in Fig. 9. CoCo-MILP instead learns variable-level contrast and ranking; Fig. 10 shows near-complete feasibility on SC but little feasibility improvement on the other tasks. These results suggest that

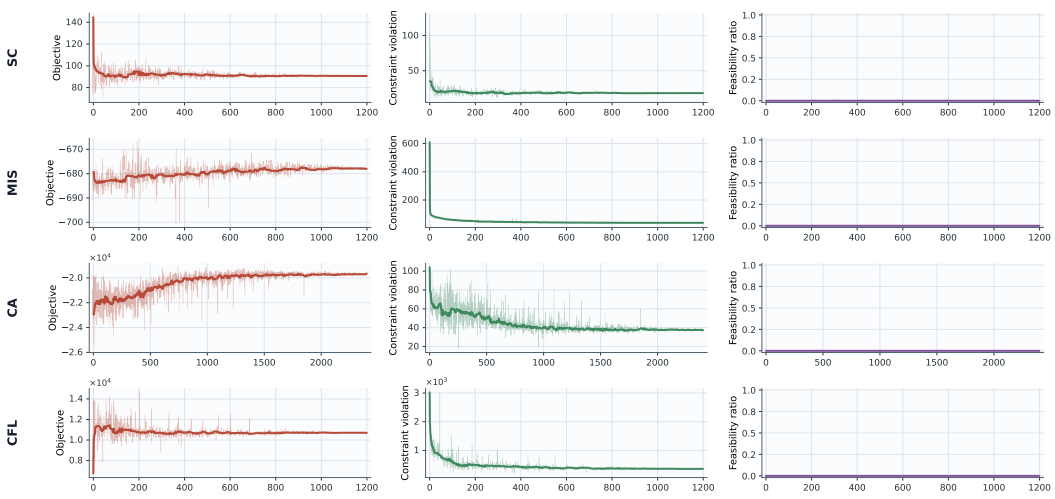



Figure 8: Training dynamics for the supervised learner model in PS (Han et al., 2023) (PS-SL).

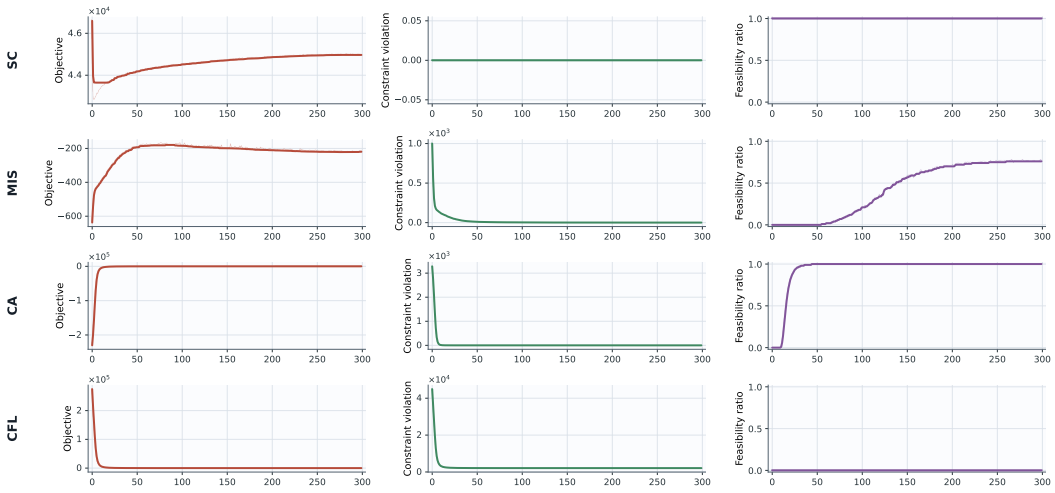



Figure 9: Training dynamics for instance-level contrastive learning (ConPaS) (Huang et al., 2024a).

relative solution or variable preferences can be more informative than pointwise label fitting, but do not by themselves ensure that all constraints are jointly satisfied.

**Generative Objectives.** The diffusion and flow Learners in Figures 12 and 13 model distributions over complete solutions rather than a single set of independent marginals. Their training dynamics nevertheless show that solver-free feasibility can be non-monotonic: the feasible ratio often rises during intermediate training and subsequently declines, even when the decoded objective and average constraint violation appear stable. This behavior is particularly visible on MIS and CA and indicates that improved denoising or trajectory matching does not necessarily translate monotonically into a larger feasible probability mass after discrete decoding. Their ability to represent correlated assignments remains valuable, especially under sampling, but persistent solver-free feasibility requires feasibility information to remain active throughout distribution learning rather than being treated only as auxiliary guidance.

**Unsupervised Objectives and Beyond.** Fig. 11 presents a distinct pattern for DiffILO. By directly optimizing an objective-violation merit function from the MILP formulation, it rapidly suppresses violations and maintains high feasibility across all four tasks, with only occasional fluctuations. This provides evidence that formulation-aware supervision can align the learned scores with the feasible region without relying on solver-generated labels. The resulting feasibility does not necessarily

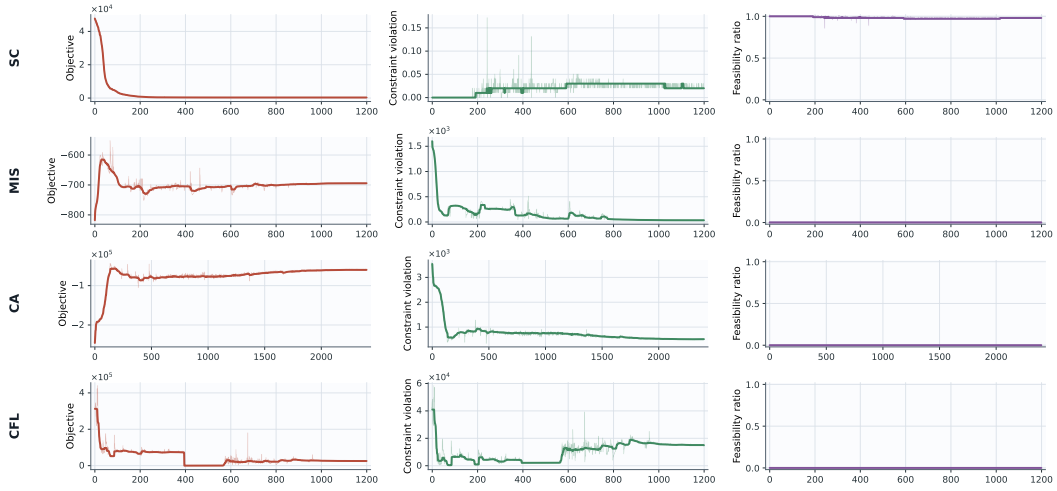



Figure 10: Training dynamics for variable-level contrastive learning (CoCo-MILP) (Pu et al., 2026).

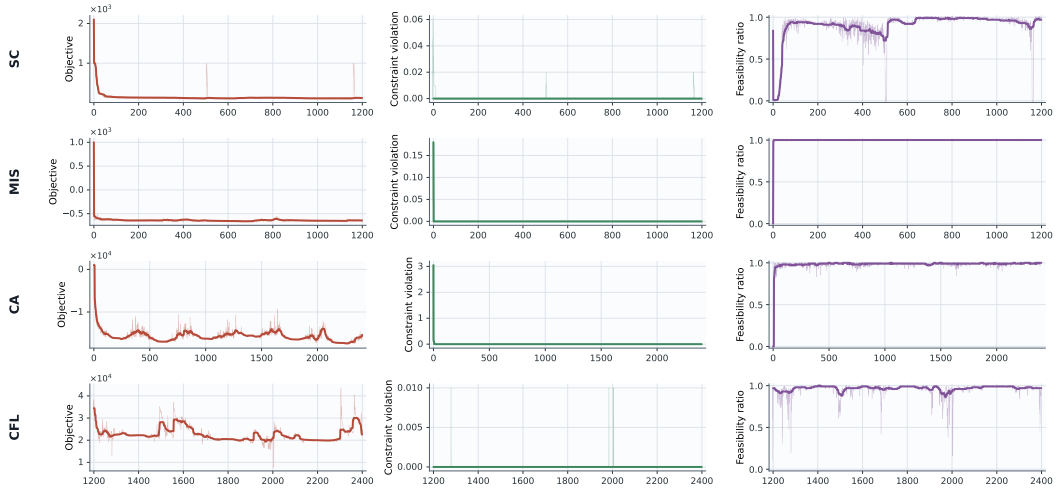



Figure 11: Training dynamics for unsupervised learning (DiffILO) (Geng et al., 2025).

imply the best objective value, however, since the penalty weight must still balance feasibility against solution quality. OPTFM represents a hybrid alternative combining learned representations with a supervised predictor. Figure 14 shows fast and stable convergence in F1, precision, and recall, particularly on MIS, CA, and CFL. These metrics quantify variable-level label recovery rather than joint constraint satisfaction, so high classification performance should not be interpreted as a direct feasibility guarantee.

Overall, the results indicate that solver-free feasibility is promoted neither by accurate marginal prediction nor by low average violation alone. A useful training design should ***expose the Learner to explicit formulation-level constraint signals***, preserve dependencies among variables through solution-level or distributional modeling, and optimize the probability mass assigned to complete feasible solutions rather than only the accuracy of individual variables. ***Feasibility penalties should remain balanced with the objective throughout training***, since generative models can move away from feasible regions after initially reaching them. ***Finally, checkpoints should be selected using exact instance- and sample-wise feasibility together with solution quality***, such as IFR, SFR, and FQS, rather than native training losses or classification metrics alone. This combination is most likely to produce guidance that remains feasible under solver-free rounding or sampling while retaining useful objective information.

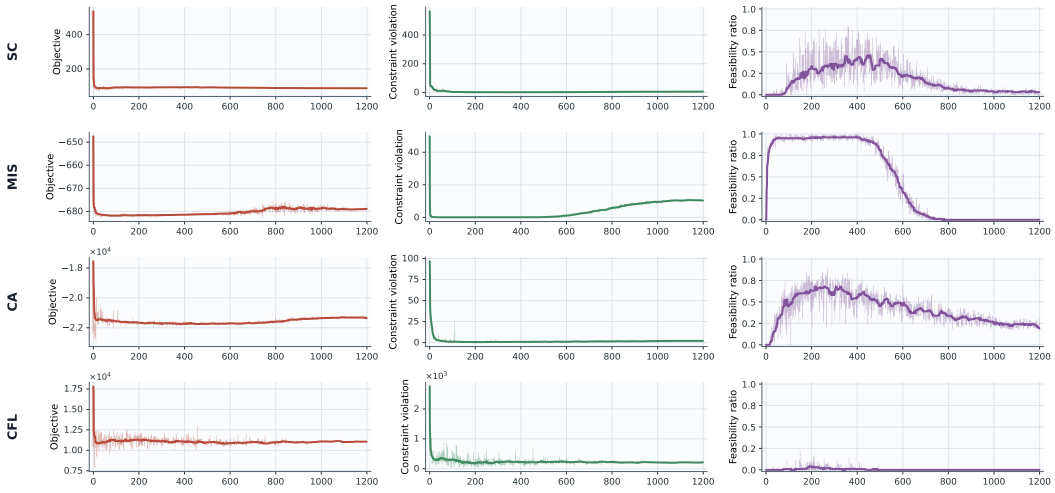



Figure 12: Training dynamics for Diffusion model (DDIM) (Zeng et al., 2024).

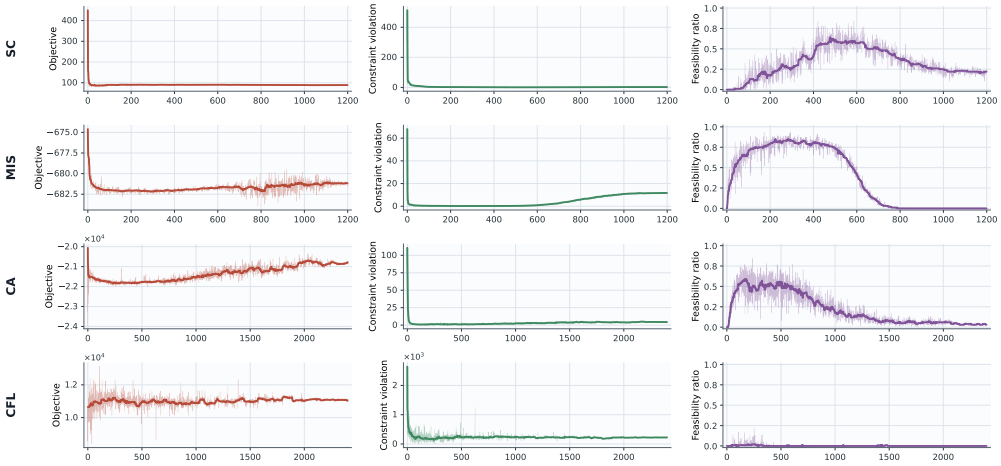



Figure 13: Training dynamics for Flow model (FMIP) (Li et al., 2026a).

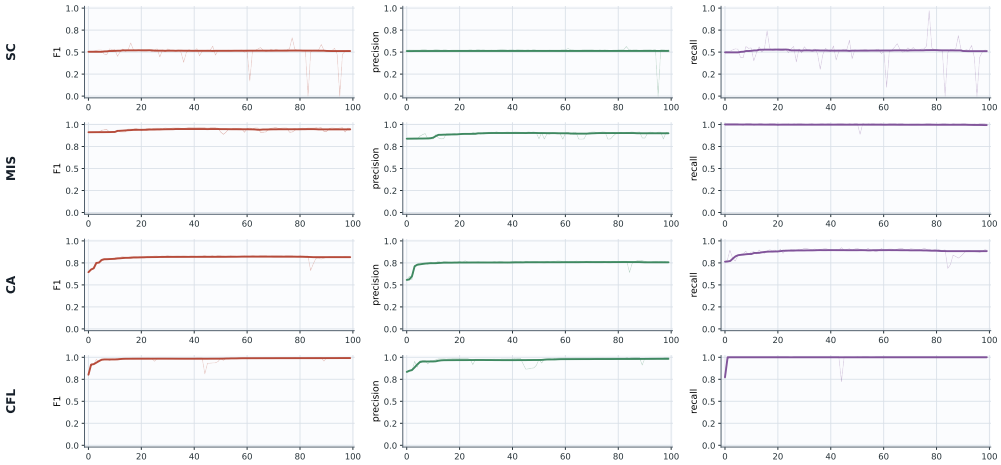



Figure 14: Training dynamics for OPTFM (Yuan et al., 2025a).

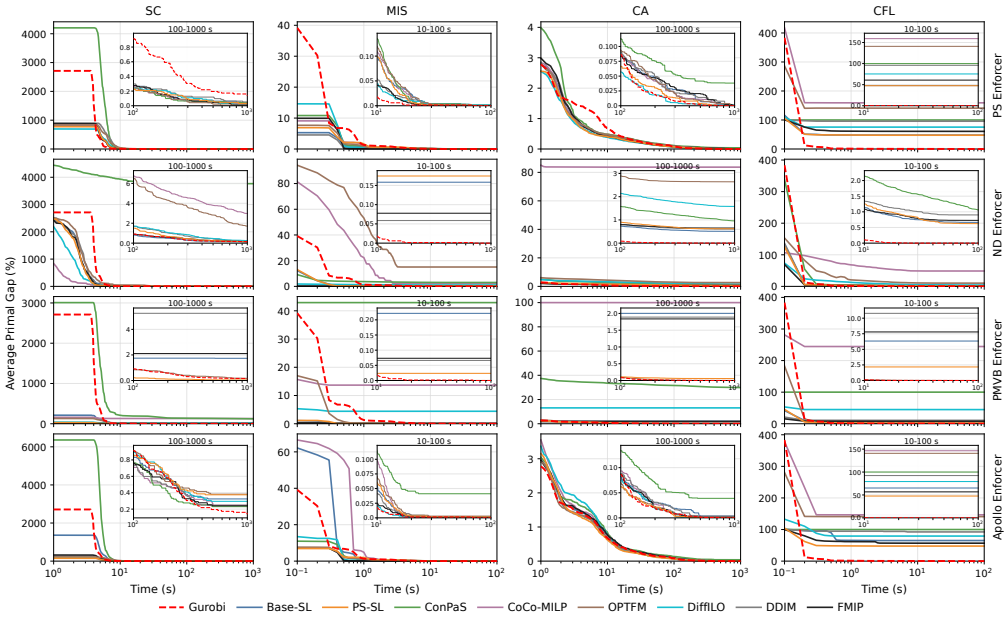



Figure 15: Complete gap-time curves for different Learners with different Enforcers. $S_E$: `Gurobi`.

### I.3 Supplementary Results on Learners with Different Enforcers

Table 16, Table 17, Table 18, and Table 19 report the complete solver-backed Enforcer results on SC, MIS, CA, and CFL, respectively. The reference BKS values are obtained with `Gurobi-3600s`. Figures 15 and 16 provide two complementary views of the corresponding gap-time trajectories, grouping curves by Enforcer and by Learner, respectively.

**Enforcer effectiveness is strongly task and Learner dependent.** Under the Gurobi backend, `PS` and `Apollo` drive most Learners to zero or near-zero gaps on MIS and CA, while `PS` also reaches gaps of 0.01-0.05% on SC at 1000 seconds. This pattern does not transfer to CFL, where PS and `Apollo` frequently plateau at large gaps or fail to produce feasible solutions on all instances. ND exhibits an almost opposite preference: on CFL, its 100-second gaps for Base-SL, PS-SL, DDIM, and FMIP are 0.66%, 0.62%, 0.90%, and 0.73%, respectively, substantially below those of PS for the same Learners. `PMVB` is even more compatibility sensitive. For example, OPTFM-`PMVB` reaches a zero mean gap on 98 CFL instances, whereas OPTFM paired with `PS` or `Apollo` remains at approximately 140.82%. Likewise, CoCo-MILP reaches zero gap on CA with `PS` or `Apollo`, but `PMVB` finds no feasible solution and `ND` remains feasible on only 17 instances. Thus, an Enforcer that is effective for one prediction distribution or task can be ineffective for another, ruling out a universal downstream ranking.

**Solver backends affect both attainable quality and anytime behavior.** The difference between Gurobi and SCIP is most pronounced when the Enforcer leaves substantial search to the backend. On MIS, Base-SL-`PS` with Gurobi reaches a zero gap and a PI of 5.83 at 100 seconds, whereas the same pairing with SCIP retains a gap of 1.67% and a PI of 252.24. Even when terminal quality agrees, the search trajectories can differ considerably: on CA, Base-SL-`PS` reaches essentially zero gap with both backends, but its PI increases from 44.82 with Gurobi to 80.68 with SCIP. In contrast, `PMVB` and rapidly terminating `ND` configurations often return identical or nearly identical gaps across the two backends, indicating that their outcomes are primarily limited by the completion decisions induced by the Learner scores rather than by subsequent backend search. Similar plateaus persist across backends on CFL, where replacing the solver cannot recover information excluded or misused by the Enforcer. The backend should therefore be treated as part of the Learner-Enforcer interface rather than as an interchangeable implementation detail.

**Solving Stability.** The reported standard deviations show that several strong pairings are also stable across instances. In particular, Gurobi-backed PS attains near-zero terminal gaps with small dispersion on MIS and CA across most Learners. However, stable terminal quality does not imply

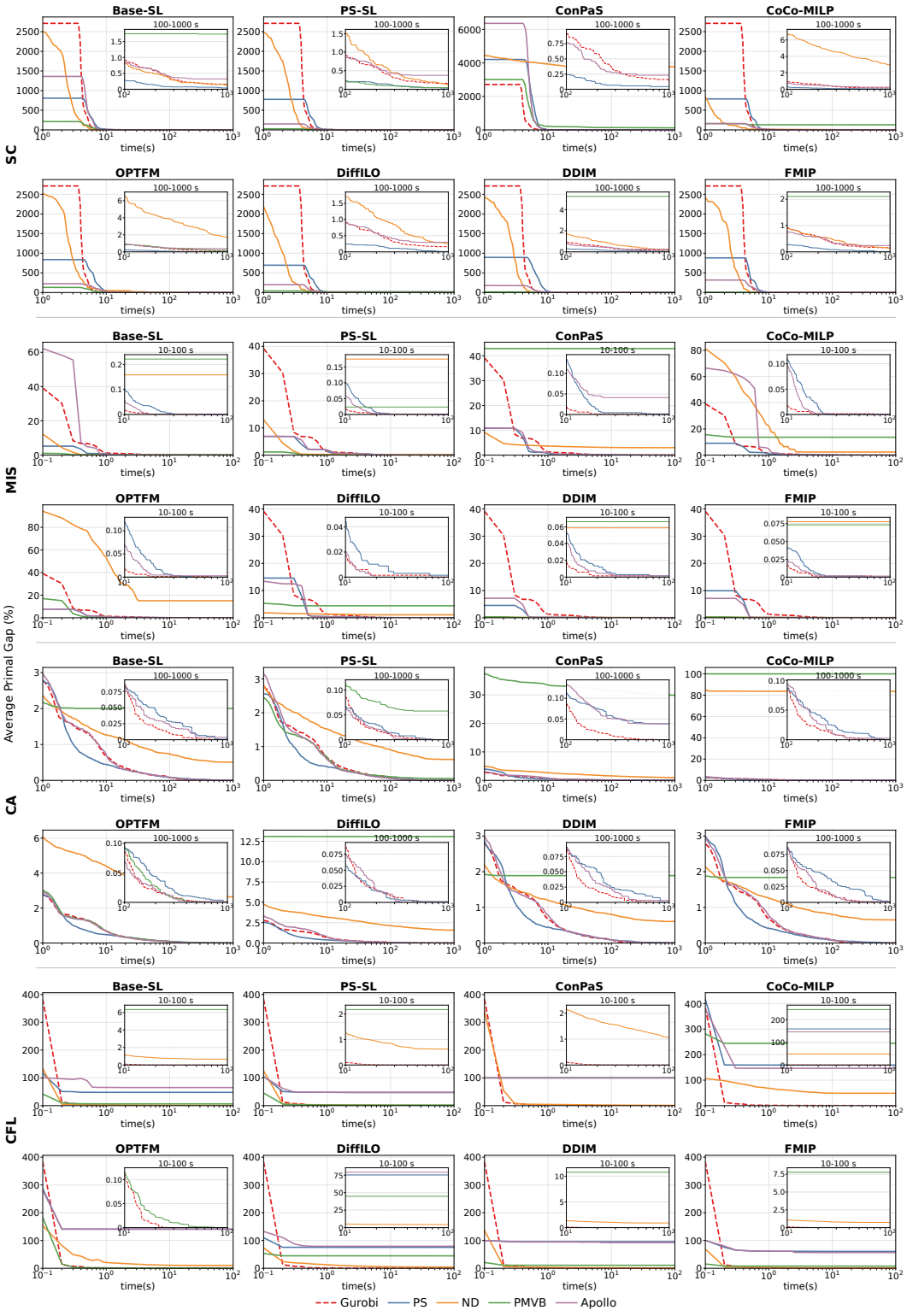



Figure 16: Gap-time curves for fixed Learner with different Enforcers. $S_E$: `Gurobi`.

stable anytime performance. For example, PS-SL-`PS` on CA has a terminal gap of $0.00 \pm 0.00\%$ but a PI of $35.51 \pm 44.45$, while Base-SL-`PS` on SC obtains $0.03 \pm 0.17\%$ with a PI of $884.66 \pm 315.88$. The final incumbents are therefore consistent even though the time required to obtain them varies

Table 16: Complete solver-backed enforcer results on SC under different time budgets.

| Learner | Enforcer | Backend | 10s Obj.↓ | 10s Gap↓ | 10s PI↓ | 10s Time | 100s Obj.↓ | 100s Gap↓ | 100s PI↓ | 100s Time | 1000s Obj.↓ | 1000s Gap↓ | 1000s PI↓ | 1000s Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BKS | – | – | 86.43* | 0.00±0.00 | – | – | 86.43* | 0.00±0.00 | – | – | 86.43* | 0.00±0.00 | – | – |
| Gurobi | – | – | 93.08 | 7.52±4.25 | 669.07±65.18 | 10.38s | 87.26 | 0.93±1.11 | 789.32±150.01 | 100.39s | 86.56 | 0.14±0.46 | 973.68±512.61 | 899.55s |
| Base-SL | PS | Gurobi | 102.49±76.39 | 17.78±30.28 | 732.11±75.87 | 10.79s | 86.68±9.04 | 0.28±0.70 | 784.39±106.86 | 100.92s | 86.36±8.93 | 0.03±0.17 | 884.66±315.86 | 973.79s |
| Base-SL | ND | Gurobi | 1912.35±157.89 | 2136.23±297.66 | 1000.00±0.00 | 10.70s | 104.77±41.89 | 20.23±42.10 | 3441.08±2890.06 | 100.90s | 86.54±8.98 | 0.12±0.40 | 725.99±528.06 | 998.69s |
| Base-SL | PMVB | Gurobi | 87.95±9.24 | 1.75±1.54 | 421.13±140.82 | 7.36s | 87.94±9.23 | 1.74±1.54 | 547.46±201.09 | 6.95s | 87.94±9.23 | 1.74±1.54 | 2210.81±1554.73 | 13.58s |
| Base-SL | Apollo | Gurobi | 1043.35±392.78 | 1120.35±479.59 | 993.22±50.23 | 6.47s | 87.62±9.25 | 1.36±1.48 | 874.13±165.11 | 51.52s | 86.72±9.02 | 0.32±0.60 | 1187.94±634.37 | 482.78s |
| PS-SL | PS | Gurobi | 101.87±90.58 | 17.45±100.83 | 712.73±70.40 | 10.71s | 86.64±9.14 | 0.22±0.51 | 800.88±121.40 | 100.72s | 86.46±8.91 | 0.03±0.18 | 870.77±248.68 | 974.21s |
| PS-SL | ND | Gurobi | 1903.77±157.92 | 2126.35±300.74 | 1000.00±0.00 | 10.47s | 121.70±194.65 | 47.54±300.78 | 3371.61±2746.35 | 105.70s | 86.53±8.92 | 0.11±0.34 | 901.44±508.62 | 999.31s |
| PS-SL | PMVB | Gurobi | 87.84±9.39 | 1.59±1.56 | 147.53±53.35 | 10.67s | 86.63±9.02 | 0.22±0.57 | 189.99±85.42 | 92.61s | 86.47±8.92 | 0.04±0.21 | 242.75±246.41 | 440.24s |
| PS-SL | Apollo | Gurobi | 132.77±52.02 | 53.72±59.22 | 682.49±203.77 | 6.31s | 87.66±9.41 | 1.39±1.35 | 763.06±179.87 | 52.10s | 86.77±9.09 | 0.37±0.67 | 1089.59±670.70 | 482.12s |
| CoCo | PS | Gurobi | 92.92±10.85 | 7.41±4.37 | 718.21±80.88 | 10.81s | 86.68±9.01 | 0.28±0.58 | 834.06±122.48 | 100.88s | 86.46±8.91 | 0.03±0.18 | 853.56±235.57 | 974.38s |
| CoCo | ND | Gurobi | 1589.58±241.61 | 1757.86±340.49 | 1000.00±0.00 | 10.61s | 93.29±10.30 | 7.88±2.83 | 1158.17±293.91 | 101.25s | 89.00±9.18 | 2.98±1.75 | 4677.16±1862.98 | 883.53s |
| CoCo | PMVB | Gurobi | 198.38±29.52 | 129.55±23.93 | 990.95±31.35 | 5.76s | 198.38±29.52 | 129.55±23.93 | 9840.53±540.87 | 6.28s | 198.38±29.52 | 129.55±23.93 | 98338.85±5633.08 | 5.42s |
| CoCo | Apollo | Gurobi | 196.57±43.10 | 127.86±46.19 | 938.36±133.76 | 7.74s | 87.49±9.30 | 1.19±1.23 | 772.36±177.44 | 52.67s | 86.65±8.97 | 0.25±0.54 | 1031.96±588.31 | 481.46s |
| DiffILO | PS | Gurobi | 92.99±10.80 | 7.51±4.37 | 762.17±44.06 | 10.89s | 86.68±9.05 | 0.27±0.64 | 854.73±250.93 | 104.23s | 86.44±8.89 | 0.01±0.10 | 905.08±270.23 | 974.31s |
| DiffILO | ND | Gurobi | 1819.68±161.23 | 2028.08±290.66 | 1000.00±0.00 | 10.68s | 91.62±12.97 | 5.73±6.26 | 1385.80±1016.44 | 100.70s | 86.66±9.03 | 0.25±0.52 | 1081.00±748.61 | 994.14s |
| DiffILO | PMVB | Gurobi | 104.86±12.72 | 21.37±8.50 | 318.58±90.13 | 6.43s | 104.18±11.62 | 20.60±6.55 | 2149.97±652.82 | 5.09s | 104.18±11.62 | 20.60±6.55 | 20698.54±6546.84 | 5.25s |
| DiffILO | Apollo | Gurobi | 237.69±68.85 | 174.35±71.77 | 955.11±103.34 | 6.15s | 87.57±9.20 | 1.30±1.30 | 724.64±154.67 | 51.60s | 86.70±9.08 | 0.29±0.67 | 1062.10±689.87 | 481.61s |
| DDIM | PS | Gurobi | 95.19±24.37 | 10.09±26.55 | 719.30±70.35 | 18.86s | 86.67±9.04 | 0.26±0.51 | 839.05±148.33 | 108.94s | 86.48±8.93 | 0.05±0.27 | 1035.82±374.40 | 995.89s |
| DDIM | ND | Gurobi | 1900.37±153.43 | 2122.00±291.03 | 1000.00±0.00 | 18.63s | 109.69±46.32 | 25.34±45.35 | 3936.30±3243.01 | 109.00s | 86.61±8.99 | 0.20±0.48 | 1080.80±630.82 | 1016.69s |
| DDIM | PMVB | Gurobi | 91.05±9.35 | 5.38±2.81 | 97.93±60.24 | 14.18s | 90.95±9.41 | 5.25±2.61 | 577.79±269.91 | 17.12s | 90.95±9.41 | 5.25±2.61 | 5302.38±2610.98 | 24.85s |
| DDIM | Apollo | Gurobi | 180.65±90.58 | 109.78±78.37 | 827.05±209.33 | 18.18s | 87.36±9.31 | 1.04±1.13 | 681.93±144.84 | 72.60s | 86.66±9.08 | 0.25±0.56 | 1056.77±566.16 | 516.41s |
| FMIP | PS | Gurobi | 92.84±10.96 | 7.31±4.40 | 776.34±69.10 | 19.11s | 86.68±9.03 | 0.28±0.57 | 816.19±111.88 | 111.61s | 86.44±8.89 | 0.01±0.11 | 842.77±240.78 | 982.53s |
| FMIP | ND | Gurobi | 1905.14±159.37 | 2127.43±295.17 | 1000.00±0.00 | 18.71s | 101.72±37.88 | 16.85±38.13 | 3227.16±2895.24 | 108.80s | 86.56±8.98 | 0.14±0.37 | 782.43±516.76 | 1007.54s |
| FMIP | PMVB | Gurobi | 88.25±9.16 | 2.11±1.62 | 48.05±56.54 | 14.71s | 88.25±9.16 | 2.11±1.62 | 242.45±174.58 | 15.15s | 88.25±9.16 | 2.11±1.62 | 2138.08±1616.66 | 14.60s |
| FMIP | Apollo | Gurobi | 174.84±91.77 | 104.07±107.81 | 773.31±219.21 | 21.33s | 87.29±9.25 | 0.97±1.15 | 707.41±148.79 | 67.76s | 86.65±9.03 | 0.24±0.50 | 986.52±541.57 | 498.83s |
| ConPaS | PS | Gurobi | 92.68±11.01 | 7.11±4.12 | 723.94±64.96 | 10.76s | 86.60±9.00 | 0.19±0.44 | 865.86±155.64 | 100.86s | 86.47±8.91 | 0.04±0.22 | 922.43±273.69 | 974.77s |
| ConPaS | ND | Gurobi | 3691.37±206.56 | 4220.63±546.55 | 1000.00±0.00 | 10.63s | 3514.41±193.52 | 4008.08±471.55 | 10000.00±0.00 | 23.99s | 3306.76±206.52 | 3765.62±461.60 | 100000.00±0.00 | 22.79s |
| ConPaS | PMVB | Gurobi | 620.47±235.64 | 623.74±271.80 | 1000.00±0.00 | 10.78s | 524.31±62.29 | 512.51±91.20 | 10000.00±0.00 | 100.88s | 190.90±30.29 | 122.40±37.91 | 95564.46±1025.89 | 1000.78s |
| ConPaS | Apollo | Gurobi | 2505.00±2350.51 | 2822.52±2791.75 | 990.69±65.23 | 8.15s | 140.46±527.88 | 57.54±560.67 | 1098.55±913.12 | 74.81s | 86.64±9.02 | 0.23±0.50 | 1119.46±507.54 | 487.13s |
| OPTFM | PS | Gurobi | 94.01±13.32 | 8.68±10.08 | 785.06±68.98 | 10.71s | 86.65±9.04 | 0.24±0.58 | 809.71±102.06 | 100.70s | 86.44±8.89 | 0.01±0.10 | 942.75±231.48 | 973.86s |
| OPTFM | ND | Gurobi | 1971.13±161.42 | 2204.68±303.05 | 1000.00±0.00 | 10.47s | 184.95±79.80 | 110.34±80.08 | 8202.90±2426.93 | 100.67s | 87.93±9.32 | 1.71±1.41 | 4733.22±2209.67 | 1000.49s |
| OPTFM | PMVB | Gurobi | 90.65±11.12 | 4.74±4.10 | 486.47±131.13 | 10.65s | 87.20±9.17 | 0.87±1.09 | 630.84±188.07 | 100.87s | 86.59±9.04 | 0.17±0.56 | 920.21±666.95 | 894.93s |
| OPTFM | Apollo | Gurobi | 218.10±81.62 | 152.45±94.06 | 907.01±164.69 | 6.31s | 87.61±9.44 | 1.32±1.36 | 912.91±184.21 | 51.95s | 86.77±9.01 | 0.38±0.62 | 1329.29±695.92 | 481.38s |
| SCIP | – | – | 90.39 | 4.50±2.71 | 274.79±32.49 | 10.78s | 87.73 | 1.45±1.49 | 545.63±189.97 | 100.79s | 86.63 | 0.21±0.59 | 992.73±792.65 | 956.52s |
| Base-SL | PS | SCIP | 90.40±10.08 | 4.52±2.59 | 232.41±67.31 | 11.46s | 87.90±9.73 | 1.65±1.52 | 605.02±186.92 | 101.42s | 86.63±9.10 | 0.21±0.61 | 1071.02±841.01 | 954.59s |
| Base-SL | ND | SCIP | 90350.99±1148.06 | 105539.24±10916.47 | 1000.00±0.00 | 11.04s | 94.29±32.58 | 8.87±34.45 | 896.26±960.31 | 100.90s | 86.66±9.09 | 0.25±0.58 | 831.49±790.58 | 1001.33s |
| Base-SL | PMVB | SCIP | 87.96±9.26 | 1.76±1.55 | 217.21±25.38 | 8.15s | 87.94±9.23 | 1.74±1.54 | 369.70±149.85 | 8.51s | 87.94±9.23 | 1.74±1.54 | 1956.76±1538.37 | 8.95s |
| Base-SL | Apollo | SCIP | 93.92±11.07 | 8.54±3.73 | 309.82±33.21 | 7.62s | 88.92±9.82 | 2.81±1.99 | 640.66±202.11 | 52.84s | 86.78±9.19 | 0.38±0.83 | 1066.78±897.80 | 492.89s |
| PS-SL | PS | SCIP | 90.53±10.33 | 4.65±2.60 | 332.63±65.59 | 11.35s | 87.95±9.63 | 1.69±1.60 | 605.29±181.96 | 101.56s | 86.59±9.10 | 0.16±0.51 | 1014.09±767.73 | 952.47s |
| PS-SL | ND | SCIP | 90385.31±1162.26 | 105574.61±10874.95 | 1000.00±0.00 | 11.03s | 92.36±10.65 | 6.76±2.93 | 1020.16±433.10 | 100.91s | 86.74±9.09 | 0.34±0.66 | 1228.84±794.15 | 1001.34s |
| PS-SL | PMVB | SCIP | 89.64±10.52 | 3.59±3.04 | 292.83±38.89 | 11.31s | 87.03±9.38 | 0.65±1.04 | 419.11±137.61 | 99.83s | 86.48±8.99 | 0.05±0.39 | 512.75±492.84 | 589.32s |
| PS-SL | Apollo | SCIP | 94.83±13.01 | 9.41±5.33 | 324.61±46.29 | 7.69s | 88.68±9.82 | 2.53±2.03 | 589.38±203.18 | 52.96s | 86.86±9.26 | 0.46±0.89 | 1090.74±973.08 | 493.27s |
| CoCo | PS | SCIP | 90.58±10.28 | 4.71±2.69 | 384.34±78.81 | 11.95s | 87.84±9.68 | 1.56±1.46 | 654.37±207.68 | 102.05s | 86.70±9.18 | 0.28±0.79 | 1175.06±964.88 | 951.37s |
| CoCo | ND | SCIP | 90435.96±1182.85 | 105635.92±10909.19 | 1000.00±0.00 | 11.60s | 94.40±10.11 | 9.21±2.69 | 1172.83±272.63 | 101.52s | 89.49±9.40 | 3.53±1.89 | 5417.22±1955.79 | 981.47s |
| CoCo | PMVB | SCIP | 198.38±29.52 | 129.55±23.93 | 986.94±44.16 | 4.01s | 198.38±29.52 | 129.55±23.93 | 9836.34±554.99 | 3.78s | 198.38±29.52 | 129.55±23.93 | 98335.51±5644.08 | 3.47s |
| CoCo | Apollo | SCIP | 93.54±11.77 | 8.03±4.20 | 300.92±40.14 | 9.03s | 88.81±10.03 | 2.66±2.00 | 580.79±196.23 | 54.08s | 86.85±9.36 | 0.44±0.95 | 1122.74±1053.91 | 494.28s |
| DiffILO | PS | SCIP | 90.78±10.46 | 4.93±2.85 | 407.25±87.73 | 11.28s | 87.86±9.57 | 1.60±1.62 | 634.84±211.68 | 101.39s | 86.62±9.12 | 0.20±0.56 | 1152.13±878.53 | 954.37s |
| DiffILO | ND | SCIP | 90390.04±1131.32 | 105589.11±10969.42 | 1000.00±0.00 | 10.88s | 91.37±10.50 | 5.62±2.86 | 813.48±277.83 | 100.82s | 86.80±9.05 | 0.41±0.67 | 1268.37±885.05 | 1001.73s |
| DiffILO | PMVB | SCIP | 104.18±11.62 | 20.60±6.55 | 368.23±53.85 | 3.34s | 104.18±11.62 | 20.60±6.55 | 2213.37±643.44 | 3.15s | 104.18±11.62 | 20.60±6.55 | 20774.33±6536.44 | 3.88s |
| DiffILO | Apollo | SCIP | 93.12±11.44 | 7.57±3.95 | 304.85±62.62 | 7.61s | 88.96±9.98 | 2.84±1.93 | 589.83±213.08 | 52.61s | 86.84±9.30 | 0.44±0.89 | 1089.33±961.56 | 493.85s |
| DDIM | PS | SCIP | 90.91±10.55 | 5.07±2.81 | 408.36±73.01 | 20.66s | 87.68±9.61 | 1.38±1.57 | 649.31±183.42 | 108.61s | 86.68±9.18 | 0.26±0.88 | 1085.31±986.63 | 965.36s |
| DDIM | ND | SCIP | 90357.73±1161.04 | 105544.52±10904.54 | 1000.00±0.00 | 18.06s | 91.89±10.54 | 6.23±3.24 | 869.04±319.30 | 107.87s | 86.87±9.11 | 0.49±0.73 | 1305.76±852.74 | 1010.77s |
| DDIM | PMVB | SCIP | 90.95±9.41 | 5.25±2.61 | 247.71±22.70 | 13.46s | 90.95±9.41 | 5.25±2.61 | 740.26±257.63 | 28.19s | 90.95±9.41 | 5.25±2.61 | 5447.87±2599.19 | 13.34s |
| DDIM | Apollo | SCIP | 94.89±17.04 | 9.34±10.44 | 311.44±80.53 | 21.21s | 88.78±9.98 | 2.63±2.15 | 584.88±216.96 | 96.06s | 86.80±9.25 | 0.39±0.86 | 1084.46±939.51 | 878.44s |
| FMIP | PS | SCIP | 91.59±10.74 | 5.62±3.63 | 421.27±77.82 | 18.71s | 88.00±9.69 | 1.75±1.65 | 686.34±216.59 | 108.66s | 86.64±9.12 | 0.22±0.66 | 1148.54±899.88 | 958.25s |
| FMIP | ND | SCIP | 90370.99±1168.25 | 105561.94±10917.81 | 1000.00±0.00 | 18.08s | 122.82±164.87 | 42.28±196.06 | 1546.33±1841.82 | 115.23s | 86.65±9.00 | 0.25±0.48 | 885.21±713.80 | 1008.77s |
| FMIP | PMVB | SCIP | 88.29±9.22 | 2.15±1.64 | 269.10±30.44 | 17.17s | 88.25±9.16 | 2.11±1.62 | 451.56±157.12 | 32.11s | 88.25±9.16 | 2.11±1.62 | 2312.73±1618.97 | 16.14s |
| FMIP | Apollo | SCIP | 95.37±12.38 | 10.11±5.01 | 332.33±41.12 | 21.11s | 88.95±9.96 | 2.83±1.97 | 610.17±207.29 | 67.27s | 86.90±9.38 | 0.50±0.96 | 1152.33±998.45 | 878.63s |
| ConPaS | PS | SCIP | 95.14±11.32 | 9.96±4.41 | 473.91±63.90 | 11.74s | 87.84±9.63 | 1.57±1.59 | 706.00±198.32 | 101.25s | 86.56±9.03 | 0.14±0.41 | 1117.56±769.52 | 946.78s |
| ConPaS | ND | SCIP | 19744.30±33953.91 | 23562.77±41301.17 | 1000.00±0.00 | 11.41s | 3306.76±206.52 | 3765.62±461.60 | 10000.00±0.00 | 51.97s | 3306.76±206.52 | 3765.62±461.60 | 100000.00±0.00 | 50.58s |
| ConPaS | PMVB | SCIP | 1335.97±294.64 | 1459.82±372.43 | 999.38±6.14 | 11.58s | 253.45±47.96 | 195.63±61.23 | 9910.02±446.76 | 101.20s | 204.15±30.16 | 137.87±38.42 | 97557.28±8651.98 | 1001.36s |
| ConPaS | Apollo | SCIP | 133.02±33.93 | 52.39±29.77 | 684.10±175.25 | 11.86s | 89.06±9.89 | 2.97±2.22 | 792.00±227.33 | 57.69s | 86.78±9.28 | 0.37±0.84 | 1221.01±911.89 | 497.45s |
| OPTFM | PS | SCIP | 92.48±11.57 | 6.83±4.66 | 418.82±74.48 | 11.10s | 87.94±9.71 | 1.67±1.64 | 674.11±220.18 | 101.19s | 86.62±9.10 | 0.20±0.69 | 1180.78±941.38 | 951.69s |
| OPTFM | ND | SCIP | 90435.04±1171.57 | 105632.36±10885.25 | 1000.00±0.00 | 10.64s | 101.03±22.50 | 16.64±20.49 | 2161.02±1668.13 | 100.77s | 88.39±9.48 | 2.23±1.62 | 5191.59±2399.18 | 1008.91s |
| OPTFM | PMVB | SCIP | 93.40±11.96 | 7.83±4.00 | 293.60±38.49 | 11.19s | 87.94±9.68 | 1.68±1.72 | 599.31±201.13 | 101.28s | 86.63±9.06 | 0.21±0.60 | 1007.23±791.49 | 942.09s |
| OPTFM | Apollo | SCIP | 96.47±19.98 | 11.07±14.35 | 423.49±115.84 | 7.41s | 89.10±9.98 | 3.00±2.03 | 750.26±240.80 | 52.56s | 86.81±9.36 | 0.39±0.96 | 1192.78±1018.71 | 500.88s |
| COPT | – | – | 1514.74 | 1666.98±241.04 | (10000.00) | 11.19s | 1388.64 | 1519.55±220.95 | (10000.00) | 101.25s | 86.60 | 0.18±0.76 | 29676.31±4301.93 | 978.59s |
| Base-SL | PS | COPT | 1520.55±162.65 | 1674.97±246.55 | 1000.00±0.00 | 12.15s | 1366.46±148.28 | 1493.11±206.98 | 10000.00±0.00 | 102.30s | 86.73±9.29 | 0.31±1.21 | 38925.55±7596.20 | 989.97s |
| Base-SL | ND | COPT | 26564.00±40630.03 | 29291.19±44922.36(7) | 1000.00±0.00 | 11.32s | 1289.71±141.62 | 1405.45±210.68 | 10000.00±0.00 | 101.53s | 86.70±9.01 | 0.30±0.62 | 1347.95±1012.55 | 1002.71s |
| Base-SL | PMVB | COPT | 177.25±80.76 | 104.78±91.14 | 999.53±4.66 | 12.17s | 129.74±52.45 | 49.49±55.91 | 9988.14±118.00 | 102.33s | 87.94±9.23 | 1.74±1.54 | 29763.32±5928.95 | 296.90s |
| Base-SL | Apollo | COPT | 218.41±73.18 | 152.85±81.17 | 989.65±22.41 | 12.85s | 216.96±64.45 | 151.55±72.16 | 9640.93±800.18 | 58.72s | 96.69±29.79 | 11.68±31.56 | 43399.37±17631.78 | 506.02s |
| PS-SL | PS | COPT | 1540.08±163.89 | 1696.86±239.61 | 1000.00±0.00 | 11.81s | 1388.91±148.58 | 1519.83±210.84 | 10000.00±0.00 | 101.91s | 86.55±9.09 | 0.12±0.55 | 29232.76±3638.03 | 977.03s |
| PS-SL | ND | COPT | 28534.33±40271.29 | 33610.38±47743.97(27) | 1000.00±0.00 | 11.22s | 1264.31±134.72 | 1377.10±225.02 | 10000.00±0.00 | 101.42s | 86.64±9.01 | 0.23±0.50 | 1333.20±712.23 | 1001.85s |
| PS-SL | PMVB | COPT | 777.91±166.72 | 806.35±201.43 | 1000.00±0.00 | 11.75s | 680.66±157.22 | 691.72±184.57 | 10000.00±0.00 | 102.05s | 86.45±8.90 | 0.02±0.15 | 31784.69±3348.16 | 774.82s |
| PS-SL | Apollo | COPT | 97.96±10.92 | 13.37±5.47 | 878.09±54.68 | 12.68s | 98.78±11.42 | 14.27±5.55 | 6029.52±254.66 | 57.93s | 87.34±9.83 | 0.97±1.77 | 30345.92±3106.04 | 504.22s |
| CoCo | PS | COPT | 1541.04±149.42 | 1699.85±242.36 | 1000.00±0.00 | 12.70s | 1384.38±139.94 | 1515.94±215.32 | 10000.00±0.00 | 102.72s | 86.73±9.28 | 0.31±0.88 | 35658.83±4891.29 | 985.85s |
| CoCo | ND | COPT | 4758.48±17594.60 | 5521.52±20920.93 | 1000.00±0.00 | 11.70s | 1033.20±131.00 | 1105.41±180.15 | 10000.00±0.00 | 101.92s | 89.54±9.49 | 3.58±1.91 | 5691.41±1931.72 | 986.60s |
| CoCo | PMVB | COPT | 211.85±37.58 | 144.97±32.78 | 998.72±8.21 | 12.56s | 200.88±29.45 | 132.42±23.52 | 9951.52±243.29 | 94.69s | 198.38±29.52 | 129.55±23.93 | 98420.54±5351.69 | 72.38s |
| CoCo | Apollo | COPT | 164.72±90.81 | 91.58±105.82 | 964.91±54.17 | 14.26s | 100.69±10.64 | 16.58±5.30 | 6200.49±247.40 | 68.43s | 88.16±10.19 | 1.90±2.70 | 36579.79±5142.66 | 505.81s |
| DiffILO | PS | COPT | 1560.81±174.00 | 1720.92±248.05 | 1000.00±0.00 | 11.76s | 1390.70±159.28 | 1521.67±220.02 | 10000.00±0.00 | 101.76s | 86.61±9.10 | 0.19±0.65 | 29776.20±4366.17 | 977.68s |
| DiffILO | ND | COPT | 6558.79±21311.06 | 7256.43±23808.99 | 1000.00±0.00 | 11.60s | 1191.57±143.36 | 1292.84±215.25 | 10000.00±0.00 | 101.10s | 86.78±9.09 | 0.39±0.74 | 1535.92±1063.85 | 1001.72s |
| DiffILO | PMVB | COPT | 112.70±12.38 | 30.45±6.27 | 917.69±107.53 | 11.86s | 104.67±12.04 | 21.14±6.83 | 6983.62±1669.98 | 82.11s | 104.18±11.62 | 20.60±6.55 | 25796.00±5960.98 | 86.40s |
| DiffILO | Apollo | COPT | 101.52±13.83 | 17.34±8.99 | 896.45±18.82 | 12.75s | 103.25±13.54 | 19.37±8.65 | 6358.24±396.82 | 60.06s | 87.26±9.81 | 0.87±1.59 | 30945.37±4045.36 | 502.71s |
| DDIM | PS | COPT | 1230.86±141.92 | 1334.89±189.15 | 1000.00±0.00 | 23.19s | 1120.58±140.54 | 1205.01±174.17 | 10000.00±0.00 | 109.75s | 86.66±9.20 | 0.24±0.74 | 34693.08±4721.36 | 999.53s |
| DDIM | ND | COPT | 203.18±480.75 | 146.09±609.93 | 1000.00±0.00 | 24.21s | 1291.53±155.25 | 1408.13±222.38 | 10000.00±0.00 | 109.04s | 86.77±9.15 | 0.37±0.68 | 1511.45±810.51 | 1016.70s |
| DDIM | PMVB | COPT | 121.12±30.15 | 39.84±29.25 | 993.58±25.57 | 39.39s | 100.28±12.84 | 15.90±6.15 | 9854.37±574.31 | 111.61s | 90.95±9.41 | 5.25±2.61 | 19626.70±4639.50 | 171.22s |
| DDIM | Apollo | COPT | 1549.85±323.73 | 1709.10±404.46 | 1000.00±0.00 | 47.38s | 102.13±11.72 | 18.26±7.68 | 6963.52±307.67 | 76.30s | 87.51±10.05 | 1.14±2.06 | 31583.58±4637.24 | 545.53s |
| FMIP | PS | COPT | 1474.30±146.89 | 1621.24±227.72 | 1000.00±0.00 | 19.12s | 1306.74±138.71 | 1423.95±197.09 | 10000.00±0.00 | 108.93s | 86.56±9.04 | 0.13±0.41 | 29044.18±3090.91 | 984.86s |
| FMIP | ND | COPT | 8735.09±25743.29 | 9999.87±30008.50 | 1000.00±0.00 | 18.89s | 1264.91±142.41 | 1378.39±221.81 | 10000.00±0.00 | 109.44s | 86.65±9.08 | 0.24±0.53 | 1227.85±840.20 | 1011.61s |
| FMIP | PMVB | COPT | 129.31±40.98 | 48.77±40.67 | 996.33±25.20 | 18.96s | 103.08±19.38 | 18.76±13.99 | 9940.92±404.41 | 110.34s | 88.25±9.16 | 2.11±1.62 | 21237.58±4466.97 | 213.87s |
| FMIP | Apollo | COPT | 1558.56±159.59 | 1719.28±243.87 | 1000.00±0.00 | 21.69s | 104.90±18.53 | 21.47±19.16 | 6926.01±636.73 | 72.39s | 87.34±9.70 | 0.98±1.67 | 30017.52±3379.30 | 522.92s |
| ConPaS | PS | COPT | 673.77±124.85 | 685.29±155.63 | 1000.00±0.00 | 11.78s | 599.60±117.17 | 598.15±141.86 | 10000.00±0.00 | 102.45s | 86.66±9.15 | 0.24±0.62 | 32910.17±6265.84 | 982.15s |
| ConPaS | ND | COPT | 3740.18±275.25 | 4274.53±558.16 | 1000.00±0.00 | 11.03s | 3312.45±180.64 | 3772.93±452.38 | 10000.00±0.00 | 90.73s | 3306.76±206.52 | 3765.62±461.60 | 100000.00±0.00 | 88.49s |
| ConPaS | PMVB | COPT | 1563.90±177.45 | 1724.99±255.05 | 1000.00±0.00 | 11.72s | 1417.15±153.23 | 1553.03±218.14 | 10000.00±0.00 | 102.30s | 227.31±33.38 | 165.27±44.99 | 99226.90±3555.82 | 1002.06s |
| ConPaS | Apollo | COPT | 713.11±136.24 | 731.76±171.77 | 1000.00±0.00 | 13.30s | 592.73±116.27 | 590.75±142.93 | 10000.00±0.00 | 65.08s | 97.91±72.67 | 12.16±74.52 | 37209.74±11139.71 | 512.19s |
| OPTFM | PS | COPT | 1445.02±199.88 | 1591.39±301.55 | 1000.00±0.00 | 11.86s | 1299.80±183.36 | 1420.52±268.66 | 10000.00±0.00 | 102.06s | 86.54±9.00 | 0.12±0.45 | 29661.49±6975.37 | 977.68s |
| OPTFM | ND | COPT | 2628.00±594.74 | 3114.29±819.43(10) | 1000.00±0.00 | 11.28s | 1327.47±143.58 | 1449.48±214.50 | 10000.00±0.00 | 101.04s | 88.71±9.72 | 2.58±2.10 | 6044.66±2828.92 | 1001.79s |
| OPTFM | PMVB | COPT | 1328.86±183.12 | 1450.46±245.83 | 1000.00±0.00 | 11.85s | 1112.14±304.99 | 1201.91±369.55 | 9471.72±2091.55 | 101.72s | 86.58±9.06 | 0.16±0.54 | 27118.89±8118.27 | 975.67s |
| OPTFM | Apollo | COPT | 131.08±26.93 | 51.85±28.05 | 923.87±63.16 | 12.25s | 131.03±35.05 | 51.17±34.27 | 7559.76±1207.39 | 56.46s | 88.69±12.13 | 2.55±8.78 | 31007.94±9356.23 | 502.86s |

substantially across instances. Conversely, a small standard deviation can be misleading when feasibility is incomplete or the metric is censored: CoCo-MILP-ND on CA reports $3.85 \pm 0.88\%$ over only 17 feasible instances, and DDIM-PS on CFL reports $19.72 \pm 10.46\%$ over only 39. Mean performance, dispersion, feasible-instance coverage, and capped failures must consequently be interpreted together.

**The trajectories distinguish preservation from effective use of predictions.** Figures 15 and 16 show that PS and Apollo often compress Learner differences as solver search proceeds, especially on MIS and CA, whereas PMVB frequently produces flat, widely separated trajectories whose gaps change little with additional budget. The latter behavior exposes differences among Learner outputs but does not necessarily exploit them effectively. ND instead exhibits delayed and highly pairing-dependent improvement, ranging from rapid completion to persistent failure. These patterns explain why terminal gaps alone can understate meaningful early differences, while large separation among Learners can simply reflect insufficient correction. A desirable Enforcer should therefore preserve calibrated, high-confidence information while retaining enough search flexibility to repair

Table 17: Complete solver-backed enforcer results on MIS under different time budgets. The blue number in the parenthesis henceforth denotes the number of feasible instances.

| Learner | Enforcer | Backend | 1s | | | | 10s | | | | 100s | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Obj.↑ | Gap↓ | PI↓ | Time | Obj.↑ | Gap↓ | PI↓ | Time | Obj.↑ | Gap↓ | PI↓ | Time |
| BKS | – | – | 684.14* | 0.00±0.00 | – | – | 684.14* | 0.00±0.00 | – | – | 684.14* | 0.00±0.00 | – | – |
| Gurobi | – | – | 675.18 | 1.31±1.00 | 12.79±1.75 | 1.09s | 684.02 | 0.02±0.05 | 15.75±2.71 | 9.47s | 684.14 | 0.00±0.00 | 15.99±3.09 | 54.98s |
| Base-SL | PS | Gurobi | 678.12±11.07 | 0.88±1.18 | 2.84±0.84 | 1.15s | 683.47±7.37 | 0.10±0.18 | 5.21±2.91 | 10.30s | 684.14±6.62 | 0.00±0.00 | 5.83±3.92 | 77.77s |
| Base-SL | ND | Gurobi | 667.87±49.06 | 2.40±6.84 | 5.32±9.84 | 1.39s | 683.06±6.98 | 0.16±0.15 | 5.16±6.82 | 4.28s | 683.06±6.98 | 0.16±0.15 | 19.53±16.87 | 4.52s |
| Base-SL | PMVB | Gurobi | 682.62±7.18 | 0.22±0.20 | 0.51±0.23 | 0.65s | 682.63±7.18 | 0.22±0.20 | 2.50±2.04 | 0.65s | 682.63±7.18 | 0.22±0.20 | 22.42±20.26 | 0.64s |
| Base-SL | Apollo | Gurobi | 674.85±10.33 | 1.36±1.07 | 26.10±15.01 | 1.20s | 683.07±6.94 | 0.16±0.17 | 27.27±14.26 | 5.70s | 684.13±6.64 | 0.00±0.01 | 26.78±15.03 | 71.91s |
| PS-SL | PS | Gurobi | 672.89±16.09 | 1.65±1.81 | 4.02±1.25 | 1.14s | 683.46±7.47 | 0.10±0.18 | 7.28±3.49 | 10.04s | 684.14±6.62 | 0.00±0.00 | 7.76±4.28 | 77.13s |
| PS-SL | ND | Gurobi | 673.04±38.70 | 1.64±5.35 | 4.47±8.30 | 1.11s | 682.95±7.04 | 0.17±0.14 | 5.77±6.15 | 4.08s | 682.95±7.04 | 0.17±0.14 | 21.69±15.57 | 4.19s |
| PS-SL | PMVB | Gurobi | 683.91±6.78 | 0.03±0.08 | 0.46±0.16 | 0.95s | 683.98±6.74 | 0.02±0.06 | 0.68±0.67 | 3.25s | 683.98±6.74 | 0.02±0.06 | 2.80±6.21 | 8.35s |
| PS-SL | Apollo | Gurobi | 678.37±9.72 | 0.85±0.72 | 7.10±0.73 | 0.84s | 682.31±7.72 | 0.27±0.26 | 11.22±3.80 | 5.28s | 684.13±6.64 | 0.00±0.01 | 11.14±4.38 | 42.10s |
| CoCo | PS | Gurobi | 677.16±12.79 | 1.02±1.46 | 4.89±1.51 | 1.17s | 683.40±7.28 | 0.11±0.15 | 7.59±3.33 | 10.11s | 684.14±6.62 | 0.00±0.00 | 8.12±3.84 | 83.92s |
| CoCo | ND | Gurobi | 566.85±85.69 | 17.21±12.41(84) | 52.40±33.37 | 1.10s | 674.41±7.62 | 1.43±0.42(99) | 78.47±106.98 | 3.53s | 674.41±7.62 | 1.43±0.42(99) | 297.49±977.63 | 3.62s |
| CoCo | PMVB | Gurobi | 671.99±10.95 | 1.91±1.10(88) | 14.11±31.73 | 0.29s | 671.99±10.95 | 1.91±1.10(88) | 137.24±318.76 | 0.28s | 671.99±10.95 | 1.91±1.10(88) | 1368.40±3189.10 | 0.65s |
| CoCo | Apollo | Gurobi | 667.87±19.87 | 2.38±2.56 | 10.00±1.73 | 0.84s | 682.05±7.45 | 0.31±0.27 | 13.37±3.86 | 5.32s | 684.14±6.62 | 0.00±0.00 | 46.94±26.63 | 38.63s |
| DiffILO | PS | Gurobi | 681.12±8.37 | 0.44±0.38 | 6.81±0.84 | 1.20s | 683.78±6.96 | 0.05±0.10 | 8.35±2.12 | 10.19s | 684.13±6.64 | 0.00±0.01 | 8.41±3.43 | 93.06s |
| DiffILO | ND | Gurobi | 555.83±79.99 | 18.81±11.26 | 20.12±11.08 | 1.16s | 676.74±6.93 | 1.08±0.28 | 12.38±2.72 | 5.30s | 676.74±6.93 | 1.08±0.28 | 109.58±27.92 | 4.75s |
| DiffILO | PMVB | Gurobi | 653.91±8.27 | 4.42±0.77 | 4.71±0.76 | 0.39s | 653.91±8.27 | 4.42±0.77 | 44.47±7.70 | 0.39s | 653.91±8.27 | 4.42±0.77 | 442.12±77.12 | 0.32s |
| DiffILO | Apollo | Gurobi | 656.82±22.65 | 4.00±2.97 | 15.23±2.19 | 0.90s | 683.49±6.98 | 0.10±0.14 | 13.46±2.22 | 5.39s | 684.14±6.62 | 0.00±0.00 | 10.70±1.71 | 42.53s |
| DDIM | PS | Gurobi | 683.04±7.04 | 0.16±0.25 | 2.09±0.42 | 3.93s | 683.79±6.89 | 0.05±0.10 | 2.95±1.56 | 12.74s | 684.13±6.64 | 0.00±0.01 | 3.47±2.92 | 84.58s |
| DDIM | ND | Gurobi | 671.32±44.82 | 1.89±6.28 | 2.80±6.48 | 3.86s | 683.74±6.86 | 0.06±0.09 | 1.29±0.96 | 6.63s | 683.74±6.86 | 0.06±0.09 | 6.56±9.31 | 16.48s |
| DDIM | PMVB | Gurobi | 683.61±6.94 | 0.08±0.11 | 0.24±0.13 | 3.60s | 683.69±6.88 | 0.07±0.10 | 0.85±1.04 | 5.83s | 683.69±6.88 | 0.07±0.10 | 6.80±10.30 | 15.31s |
| DDIM | Apollo | Gurobi | 682.35±10.92 | 0.26±1.17 | 3.19±1.10 | 6.27s | 683.64±6.87 | 0.07±0.12 | 3.83±1.39 | 14.32s | 684.13±6.64 | 0.00±0.01 | 4.11±2.68 | 53.63s |
| FMIP | PS | Gurobi | 683.62±6.79 | 0.08±0.12 | 4.63±0.86 | 5.93s | 683.86±6.79 | 0.04±0.09 | 4.71±1.23 | 20.85s | 684.14±6.62 | 0.00±0.00 | 4.98±1.74 | 93.59s |
| FMIP | ND | Gurobi | 634.59±78.34 | 7.28±11.16 | 8.49±11.32 | 6.53s | 683.61±6.78 | 0.08±0.10 | 1.40±1.01 | 14.93s | 683.61±6.78 | 0.08±0.10 | 8.39±9.84 | 14.89s |
| FMIP | PMVB | Gurobi | 683.45±6.88 | 0.10±0.11 | 0.27±0.13 | 7.44s | 683.64±6.76 | 0.07±0.10 | 0.92±1.03 | 11.92s | 683.64±6.76 | 0.07±0.10 | 7.52±10.25 | 11.05s |
| FMIP | Apollo | Gurobi | 683.02±7.66 | 0.16±0.52 | 3.15±0.87 | 19.56s | 683.69±6.71 | 0.07±0.11 | 3.73±1.33 | 26.54s | 684.13±6.64 | 0.00±0.01 | 3.94±1.98 | 48.38s |
| ConPaS | PS | Gurobi | 678.91±10.53 | 0.77±1.16 | 5.42±1.11 | 1.17s | 683.21±7.21 | 0.14±0.19 | 7.62±2.74 | 10.09s | 684.13±6.64 | 0.00±0.01 | 8.34±4.42 | 81.03s |
| ConPaS | ND | Gurobi | 456.93±10.42 | 33.21±1.56 | 34.28±1.48 | 1.10s | 663.07±6.66 | 3.08±0.40 | 35.35±4.61 | 9.07s | 663.46±6.32 | 3.02±0.38 | 308.08±38.38 | 13.29s |
| ConPaS | PMVB | Gurobi | 390.60±41.82 | 42.90±6.11 | 44.36±6.07 | 0.19s | 390.60±41.82 | 42.90±6.11 | 430.46±61.08 | 0.19s | 390.60±41.82 | 42.90±6.11 | 4291.71±611.31 | 0.20s |
| ConPaS | Apollo | Gurobi | 675.31±17.37 | 1.29±2.19 | 5.49±1.44 | 0.83s | 682.71±7.17 | 0.21±0.21 | 7.78±2.85 | 5.29s | 684.02±6.79 | 0.02±0.15 | 12.34±37.79 | 58.70s |
| OPTFM | PS | Gurobi | 678.25±11.63 | 0.86±1.30 | 4.15±1.26 | 1.28s | 683.31±7.40 | 0.12±0.19 | 6.62±2.97 | 10.15s | 684.14±6.62 | 0.00±0.00 | 7.30±4.03 | 78.85s |
| OPTFM | ND | Gurobi | 483.67±10.91 | 29.28±1.56(49) | 80.06±24.16 | 1.19s | 659.98±8.28 | 3.48±0.68(88) | 242.11±290.99 | 3.70s | 659.98±8.28 | 3.48±0.68(88) | 1597.84±3104.63 | 3.71s |
| OPTFM | PMVB | Gurobi | 683.44±7.18 | 0.10±0.16 | 4.53±0.82 | 1.20s | 684.14±6.62 | 0.00±0.00 | 4.54±0.88 | 10.11s | 684.14±6.62 | 0.00±0.00 | 4.85±1.35 | 85.40s |
| OPTFM | Apollo | Gurobi | 675.04±15.26 | 1.34±1.76 | 4.10±1.30 | 0.98s | 682.61±7.35 | 0.22±0.21 | 6.81±3.22 | 5.44s | 684.12±6.67 | 0.00±0.03 | 7.04±4.95 | 36.63s |
| SCIP | – | – | 660.79 | 3.41±0.56 | 11.01±0.61 | 1.08s | 660.79 | 3.41±0.56 | 41.76±5.62 | 10.08s | 680.25 | 0.57±0.56 | 189.68±53.03 | 100.08s |
| Base-SL | PS | SCIP | 664.80±7.74 | 2.83±0.49 | 11.19±0.56 | 1.47s | 664.80±7.74 | 2.83±0.49 | 36.62±4.82 | 10.48s | 672.73±11.02 | 1.67±1.04 | 252.24±53.49 | 100.57s |
| Base-SL | ND | SCIP | 287.97±290.04 | 58.03±42.20 | 59.39±41.72 | 1.47s | 683.06±6.98 | 0.16±0.15 | 7.39±9.65 | 5.94s | 683.06±6.98 | 0.16±0.15 | 21.47±17.70 | 5.75s |
| Base-SL | PMVB | SCIP | 680.35±8.74 | 0.55±0.86 | 14.24±11.73 | 1.13s | 682.57±7.15 | 0.23±0.20 | 17.17±12.68 | 2.76s | 682.63±7.18 | 0.22±0.20 | 37.22±23.14 | 2.98s |
| Base-SL | Apollo | SCIP | 676.43±7.70 | 1.13±0.40 | 10.36±0.49 | 1.43s | 676.43±7.70 | 1.13±0.40 | 26.14±4.24 | 5.98s | 677.57±8.20 | 0.96±0.47 | 173.35±45.37 | 51.00s |
| PS-SL | PS | SCIP | 667.54±8.18 | 2.43±0.53 | 10.98±0.59 | 1.15s | 667.54±8.18 | 2.43±0.53 | 32.91±5.31 | 10.17s | 674.30±9.40 | 1.44±0.88 | 224.54±52.20 | 100.30s |
| PS-SL | ND | SCIP | 141.38±190.33 | 79.37±27.70 | 80.68±27.31 | 1.12s | 682.95±7.04 | 0.17±0.14 | 9.11±9.52 | 5.68s | 682.95±7.04 | 0.17±0.14 | 25.01±17.07 | 5.73s |
| PS-SL | PMVB | SCIP | 674.74±9.09 | 1.37±1.00 | 10.22±1.03 | 1.13s | 680.48±7.11 | 0.53±0.45 | 16.48±4.06 | 9.29s | 683.98±6.74 | 0.02±0.06 | 27.77±14.21 | 25.69s |
| PS-SL | Apollo | SCIP | 676.97±7.86 | 1.05±0.40 | 10.24±0.52 | 0.83s | 676.97±7.86 | 1.05±0.40 | 25.9±4.24 | 5.34s | 676.97±7.86 | 1.05±0.40 | 183.28±42.48 | 51.12s |
| CoCo | PS | SCIP | 663.72±8.28 | 2.99±0.57 | 11.69±0.70 | 1.76s | 663.72±8.28 | 2.99±0.57 | 38.40±5.70 | 10.76s | 675.83±9.95 | 1.22±0.94 | 247.82±58.21 | 100.82s |
| CoCo | ND | SCIP | 110.89±143.13 | 83.82±20.83(66) | 92.07±18.08 | 1.75s | 674.41±7.62 | 1.43±0.42(99) | 101.60±118.38 | 5.46s | 674.41±7.62 | 1.43±0.42(99) | 318.77±976.94 | 5.46s |
| CoCo | PMVB | SCIP | 671.99±10.95 | 1.91±1.10(88) | 23.11±29.04 | 0.93s | 671.99±10.95 | 1.91±1.10(88) | 146.13±315.57 | 0.92s | 671.99±10.95 | 1.91±1.10(88) | 1377.52±3185.79 | 0.83s |
| CoCo | Apollo | SCIP | 663.72±8.28 | 2.99±0.57 | 11.44±0.67 | 1.87s | 663.96±8.15 | 2.95±0.55 | 38.14±5.53 | 6.59s | 665.99±8.70 | 2.65±0.75 | 288.94±57.17 | 51.53s |
| DiffILO | PS | SCIP | 658.65±8.69 | 3.73±0.64 | 12.66±1.18 | 1.19s | 658.65±8.69 | 3.73±0.64 | 46.00±6.43 | 10.17s | 679.42±9.06 | 0.69±0.66 | 205.45±77.54 | 100.32s |
| DiffILO | ND | SCIP | 186.99±237.54 | 72.78±34.46 | 73.63±33.59 | 1.13s | 676.74±6.93 | 1.08±0.28 | 13.85±2.72 | 5.66s | 676.74±6.93 | 1.08±0.28 | 111.16±27.94 | 5.75s |
| DiffILO | PMVB | SCIP | 653.91±8.27 | 4.42±0.77 | 13.22±1.26 | 0.39s | 653.91±8.27 | 4.42±0.77 | 52.56±7.70 | 0.38s | 653.91±8.27 | 4.42±0.77 | 449.95±77.03 | 0.40s |
| DiffILO | Apollo | SCIP | 662.01±8.14 | 3.24±0.60 | 11.68±0.67 | 0.82s | 662.01±8.14 | 3.24±0.60 | 43.90±5.58 | 5.39s | 675.64±10.43 | 1.25±0.91 | 226.50±79.38 | 50.34s |
| DDIM | PS | SCIP | 666.07±8.17 | 2.64±0.55 | 11.77±1.27 | 3.70s | 666.07±8.17 | 2.64±0.55 | 35.10±5.57 | 12.57s | 671.05±9.64 | 1.91±0.88 | 251.07±57.59 | 102.13s |
| DDIM | ND | SCIP | 246.59±273.75 | 64.05±39.83 | 64.90±39.23 | 3.61s | 683.74±6.86 | 0.06±0.09 | 2.83±1.05 | 8.35s | 683.74±6.86 | 0.06±0.09 | 7.99±9.52 | 7.22s |
| DDIM | PMVB | SCIP | 674.45±8.97 | 1.42±0.71 | 10.03±0.91 | 3.59s | 682.73±7.30 | 0.21±0.26 | 14.98±2.94 | 10.73s | 683.69±6.88 | 0.07±0.10 | 20.85±11.08 | 11.56s |
| DDIM | Apollo | SCIP | 667.36±8.18 | 2.45±0.53 | 11.09±0.90 | 5.61s | 680.79±7.24 | 0.49±0.28 | 28.54±4.44 | 10.56s | 681.06±7.43 | 0.45±0.28 | 155.31±37.68 | 53.99s |
| FMIP | PS | SCIP | 672.99±8.49 | 1.63±0.50 | 10.33±0.62 | 16.88s | 672.99±8.49 | 1.63±0.50 | 25.64±5.07 | 25.80s | 679.10±8.78 | 0.74±0.49 | 141.45±45.48 | 114.88s |
| FMIP | ND | SCIP | 239.93±270.82 | 65.01±39.43 | 66.02±38.59 | 16.91s | 683.61±6.78 | 0.08±0.10 | 4.55±10.55 | 22.58s | 683.61±6.78 | 0.08±0.10 | 9.90±10.36 | 7.37s |
| FMIP | PMVB | SCIP | 674.60±9.20 | 1.40±0.71 | 10.25±0.95 | 16.75s | 683.01±7.12 | 0.17±0.20 | 14.73±2.91 | 24.93s | 683.64±6.76 | 0.07±0.10 | 20.97±10.96 | 11.51s |
| FMIP | Apollo | SCIP | 667.30±8.43 | 2.46±0.59 | 11.08±0.66 | 32.56s | 676.16±11.78 | 1.17±1.16 | 29.75±6.06 | 23.88s | 682.51±7.06 | 0.24±0.17 | 146.20±33.63 | 54.52s |
| ConPaS | PS | SCIP | 661.65±8.26 | 3.29±0.56 | 11.86±0.61 | 1.31s | 661.65±8.26 | 3.29±0.56 | 43.09±6.06 | 10.34s | 677.99±9.21 | 0.90±0.71 | 217.28±67.09 | 100.17s |
| ConPaS | ND | SCIP | 27.38±4.44 | 96.00±0.65 | 96.24±0.58 | 1.28s | 657.32±12.67 | 3.93±1.24 | 48.59±12.76 | 10.15s | 662.82±8.06 | 3.12±0.62 | 354.65±78.18 | 78.93s |
| ConPaS | PMVB | SCIP | 390.60±41.82 | 42.90±6.11 | 47.10±5.68 | 0.40s | 390.60±41.82 | 42.90±6.11 | 433.58±60.66 | 0.29s | 390.60±41.82 | 42.90±6.11 | 4294.52±610.89 | 0.27s |
| ConPaS | Apollo | SCIP | 661.67±8.28 | 3.29±0.56 | 12.21±1.22 | 1.15s | 661.67±8.28 | 3.29±0.56 | 41.17±5.59 | 5.48s | 671.33±11.82 | 1.88±1.16 | 246.92±83.34 | 50.35s |
| OPTFM | PS | SCIP | 660.53±8.61 | 3.45±0.64 | 12.29±0.74 | 1.23s | 660.53±8.61 | 3.45±0.64 | 43.52±6.39 | 10.25s | 673.78±10.75 | 1.52±1.11 | 280.18±67.44 | 100.29s |
| OPTFM | ND | SCIP | 67.16±5.42 | 90.19±0.78(32) | 98.20±2.96 | 1.21s | 659.82±8.80 | 3.50±0.75(88) | 272.51±288.18 | 4.83s | 659.98±8.28 | 3.48±0.68(88) | 1628.08±3094.29 | 4.84s |
| OPTFM | PMVB | SCIP | 661.03±8.42 | 3.38±0.59 | 11.22±0.65 | 1.21s | 661.03±8.42 | 3.38±0.59 | 41.58±5.91 | 10.23s | 675.04±9.89 | 1.33±1.15 | 265.22±69.46 | 100.36s |
| OPTFM | Apollo | SCIP | 661.02±8.58 | 3.38±0.62 | 12.08±0.81 | 1.00s | 661.02±8.58 | 3.38±0.62 | 43.12±6.32 | 5.51s | 664.71±11.69 | 2.84±1.13 | 320.34±77.93 | 50.49s |
| COPT | – | – | 589.18 | 13.88±1.26 | 48.94±1.52 | 1.12s | 611.97 | 10.55±2.00 | 163.25±11.25 | 10.15s | 682.70 | 0.21±0.31 | 392.32±151.23 | 95.80s |
| Base-SL | PS | COPT | 606.87±25.21 | 11.30±3.57 | 54.55±6.41 | 1.49s | 675.55±13.69 | 1.26±1.59 | 113.26±27.40 | 10.52s | 682.66±8.04 | 0.22±0.31 | 163.79±74.43 | 97.11s |
| Base-SL | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.31s | 682.55±7.08 | 0.23±0.17 | 23.15±28.64 | 10.28s | 683.06±6.98 | 0.16±0.15 | 36.96±31.57 | 22.24s |
| Base-SL | PMVB | COPT | 675.84±8.13 | 1.21±0.45 | 99.33±6.70 | 1.49s | 682.63±7.18 | 0.22±0.20 | 193.77±24.67 | 2.86s | 682.63±7.18 | 0.22±0.20 | 201.95±28.95 | 2.88s |
| Base-SL | Apollo | COPT | 653.90±12.14 | 4.42±1.49 | 38.37±2.73 | 1.44s | 681.88±7.40 | 0.33±0.43 | 94.99±15.38 | 5.96s | 682.57±7.32 | 0.23±0.26 | 141.98±48.85 | 50.73s |
| PS-SL | PS | COPT | 591.74±24.45 | 13.50±3.53 | 52.35±2.87 | 1.41s | 657.32±20.48 | 3.93±2.52 | 119.61±21.80 | 10.38s | 682.43±8.77 | 0.25±0.42 | 225.85±108.55 | 96.27s |
| PS-SL | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.27s | 682.57±7.02 | 0.23±0.16 | 21.07±19.35 | 10.25s | 682.95±7.04 | 0.17±0.14 | 38.86±30.60 | 17.93s |
| PS-SL | PMVB | COPT | 670.18±8.91 | 2.01±0.85(71) | 56.98±33.23 | 1.43s | 683.90±6.77 | 0.04±0.08 | 68.34±48.75 | 8.05s | 683.98±6.74 | 0.02±0.06 | 68.85±47.22 | 10.14s |
| PS-SL | Apollo | COPT | 648.58±13.67 | 5.20±1.77 | 38.76±1.69 | 1.37s | 675.83±10.13 | 1.22±0.83 | 104.89±12.73 | 6.09s | 681.12±9.23 | 0.44±0.52 | 230.68±108.55 | 51.66s |
| CoCo | PS | COPT | 584.56±25.97 | 14.56±3.70 | 58.88±12.66 | 1.53s | 668.23±22.77 | 2.33±2.99 | 143.07±42.63 | 10.51s | 682.62±8.01 | 0.22±0.30 | 210.33±126.45 | 97.52s |
| CoCo | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.33s | 672.79±8.34 | 1.70±0.57(87) | 341.31±318.02 | 10.31s | 674.41±7.62 | 1.43±0.42(99) | 605.67±1009.03 | 23.26s |
| CoCo | PMVB | COPT | 645.70±26.63 | 5.72±3.63(81) | 99.62±2.68 | 1.51s | 671.99±10.95 | 1.91±1.10(88) | 369.83±238.70 | 3.26s | 671.99±10.95 | 1.91±1.10(88) | 1553.81±3120.70 | 2.70s |
| CoCo | Apollo | COPT | 620.19±19.17 | 9.35±2.61 | 44.59±4.56 | 1.74s | 671.07±18.61 | 1.92±2.34 | 135.19±35.80 | 6.13s | 679.48±13.04 | 0.69±1.37 | 241.19±165.94 | 50.96s |
| DiffILO | PS | COPT | 604.80±26.26 | 11.60±3.77 | 47.44±8.51 | 1.67s | 679.40±8.21 | 0.69±0.41 | 86.18±22.58 | 10.70s | 682.34±7.73 | 0.26±0.25 | 126.08±38.73 | 97.33s |
| DiffILO | ND | COPT | 638.00±8.86 | 6.74±0.92 | 100.00±0.00 | 1.81s | 675.76±7.16 | 1.23±0.31 | 20.61±4.28 | 10.52s | 676.74±6.93 | 1.08±0.28 | 118.23±28.03 | 21.88s |
| DiffILO | PMVB | COPT | 642.32±9.75 | 6.11±1.10 | 68.41±33.92 | 1.79s | 653.75±8.68 | 4.44±0.83 | 172.11±136.46 | 4.20s | 653.91±8.27 | 4.42±0.77 | 569.61±168.47 | 3.89s |
| DiffILO | Apollo | COPT | 642.32±9.98 | 6.11±1.12 | 38.41±2.62 | 1.84s | 680.04±7.75 | 0.60±0.37 | 87.30±15.87 | 6.59s | 681.11±7.97 | 0.44±0.31 | 130.66±32.60 | 50.90s |
| DDIM | PS | COPT | 607.08±23.26 | 11.26±3.30 | 55.01±5.95 | 3.86s | 682.37±7.25 | 0.26±0.39 | 110.61±24.27 | 13.49s | 683.39±7.18 | 0.11±0.14 | 128.33±31.81 | 99.81s |
| DDIM | ND | COPT | 681.94±7.06 | 0.32±0.18 | 100.00±0.00 | 4.77s | 683.55±6.88 | 0.09±0.10 | 7.51±1.77 | 13.43s | 683.74±6.86 | 0.06±0.09 | 12.93±9.63 | 28.08s |
| DDIM | PMVB | COPT | 679.89±7.60 | 0.62±0.50 | 64.89±32.76 | 4.10s | 683.63±6.95 | 0.07±0.11 | 94.77±70.63 | 16.87s | 683.69±6.88 | 0.07±0.10 | 106.30±74.88 | 11.39s |
| DDIM | Apollo | COPT | 598.90±18.12 | 12.46±2.48 | 42.96±2.63 | 5.36s | 682.76±6.90 | 0.20±0.20 | 96.89±17.61 | 22.74s | 683.23±6.98 | 0.13±0.14 | 124.18±22.18 | 56.22s |
| FMIP | PS | COPT | 597.13±21.48 | 12.72±2.97 | 43.61±4.19 | 17.48s | 682.57±6.96 | 0.23±0.28 | 73.34±10.34 | 12.20s | 683.14±7.21 | 0.15±0.17 | 88.71±18.85 | 99.10s |
| FMIP | ND | COPT | 682.12±7.17 | 0.30±0.19 | 100.00±0.00 | 17.87s | 683.60±6.78 | 0.08±0.11 | 6.42±1.75 | 12.05s | 683.61±6.78 | 0.08±0.10 | 13.63±10.11 | 23.75s |
| FMIP | PMVB | COPT | 681.02±7.32 | 0.46±0.39 | 65.56±36.24 | 17.58s | 683.50±6.90 | 0.09±0.12 | 93.25±67.69 | 9.24s | 683.64±6.76 | 0.07±0.10 | 99.12±66.51 | 9.82s |
| FMIP | Apollo | COPT | 602.62±20.49 | 11.91±2.90 | 38.57±3.14 | 5.16s | 683.47±6.87 | 0.10±0.12 | 79.67±15.39 | 9.43s | 683.47±6.87 | 0.10±0.12 | 117.43±27.04 | 55.31s |
| ConPaS | PS | COPT | 581.49±19.24 | 15.00±2.88 | 52.88±12.46 | 1.38s | 676.18±15.42 | 1.17±1.90 | 105.74±34.62 | 10.39s | 682.82±7.48 | 0.19±0.24 | 174.29±94.17 | 97.91s |
| ConPaS | ND | COPT | 234.98±37.50 | 65.64±5.51(95) | 100.00±0.00 | 1.25s | 593.35±23.40 | 13.28±3.09 | 155.97±42.14 | 10.19s | 662.67±9.24 | 3.14±0.98 | 411.46±151.69 | 88.53s |
| ConPaS | PMVB | COPT | 370.14±37.19 | 45.89±5.42 | 73.98±4.37 | 1.38s | 390.60±41.82 | 42.90±6.11 | 469.02±56.77 | 2.04s | 390.60±41.82 | 42.90±6.11 | 4331.76±606.25 | 2.14s |
| ConPaS | Apollo | COPT | 578.90±20.72 | 15.38±3.10 | 43.02±5.06 | 1.30s | 675.81±15.89 | 1.22±1.98 | 107.45±34.63 | 5.78s | 681.42±9.95 | 0.40±0.85 | 166.01±110.43 | 50.60s |
| OPTFM | PS | COPT | 568.29±24.65 | 16.94±3.47 | 67.95±22.55 | 1.72s | 670.82±21.42 | 1.95±2.77 | 145.09±51.48 | 10.69s | 682.95±7.51 | 0.18±0.21 | 192.49±110.31 | 96.49s |
| OPTFM | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.53s | 654.90±14.33 | 4.22±1.68(70) | 564.58±340.32 | 10.50s | 659.98±8.28 | 3.48±0.68(88) | 1868.26±3020.54 | 14.75s |
| OPTFM | PMVB | COPT | 675.14±9.90 | 15.94±0.98 | 39.45±7.00 | 1.63s | 674.75±17.26 | 1.38±2.13 | 82.65±20.87 | 10.61s | 683.93±6.78 | 0.03±0.07 | 95.47±29.61 | 94.42s |
| OPTFM | Apollo | COPT | 631.07±15.30 | 7.76±1.90 | 47.03±8.14 | 1.93s | 674.09±16.92 | 1.47±2.07 | 136.70±50.20 | 6.29s | 681.05±11.41 | 0.46±1.06 | 200.35±134.96 | 50.47s |

unreliable predictions. This supports evaluating and designing Learner-Enforcer pairs jointly, using feasibility, full anytime trajectories, and PI rather than an isolated terminal metric.

**Clarification on the Apparent Non-Monotonicity of PIs for ND Enforcer.** The PIs of ND under different *total time budgets* $T$ should not be interpreted as points along a single anytime trajectory. Since $T$ is evenly divided among 50 diving trials, a larger $T$ also gives each trial a proportionally larger time budget. For example, with $T = 1000$s and 50 trials, each trial has a 20-second budget, so by $t = 10$s the first trial may have already found a good solution. In contrast, with $T = 10$s, each trial has only 0.2s, and even the best solution among the 50 trials may be of lower quality. Therefore, the result at $T = 10$s cannot be regarded as the 10-second prefix of the $T = 1000$s run, and the apparent

Table 18: Complete solver-backed enforcer results on CA under different time budgets.

| Learner | Enforcer | Backend | 10s | | | | 100s | | | | 1000s | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Obj.↑ | Gap (%)↓ | PI↓ | Time (s) | Obj.↑ | Gap (%)↓ | PI↓ | Time (s) | Obj.↑ | Gap (%)↓ | PI↓ | Time (s) |
| BKS | – | – | 22272.55* | 0.00±0.00 | – | – | 22272.55* | 0.00±0.00 | – | – | 22272.55* | 0.00±0.00 | – | – |
| Gurobi | – | – | 22122.02 | 0.68±0.55 | 19.31±6.56 | 10.03s | 22253.43 | 0.09±0.17 | 36.90±24.15 | 74.51s | 22272.55 | 0.00±0.00 | 42.99±35.82 | 149.91s |
| Base-SL | PS | Gurobi | 22182.56±578.94 | 0.44±0.36(99) | 13.83±4.56 | 10.00s | 22254.58±569.61 | 0.08±0.15 | 29.40±20.76 | 78.48s | 22272.36±572.21 | 0.00±0.01 | 44.82±62.12 | 221.21s |
| Base-SL | ND | Gurobi | 21839.51±576.55 | 1.94±0.72 | 25.94±7.33 | 10.29s | 22122.28±577.93 | 0.68±0.30 | 92.77±31.58 | 97.73s | 22159.52±572.64 | 0.51±0.24 | 585.49±240.35 | 463.08s |
| Base-SL | PMVB | Gurobi | 21827.82±571.90 | 2.00±0.63(99) | 21.65±6.44 | 1.47s | 21827.82±571.90 | 2.00±0.63(99) | 201.82±63.5 | 1.32s | 21827.82±571.90 | 2.00±0.63(99) | 2003.66±629.66 | 1.65s |
| Base-SL | Apollo | Gurobi | 21967.93±597.39 | 1.37±0.78 | 42.05±15.73 | 5.66s | 22237.16±575.20 | 0.16±0.19 | 40.14±22.47 | 46.47s | 22271.71±572.54 | 0.00±0.03 | 50.60±54.30 | 154.81s |
| PS-SL | PS | Gurobi | 22183.60±574.08 | 0.40±0.33 | 12.92±4.24 | 9.96s | 22257.71±571.21 | 0.07±0.16 | 27.09±21.11 | 79.02s | 22272.55±572.24 | 0.00±0.00 | 35.51±44.45 | 210.25s |
| PS-SL | ND | Gurobi | 21770.47±582.21 | 2.25±0.69 | 32.21±8.54 | 9.98s | 22086.81±568.29 | 0.83±0.36 | 113.47±40.11 | 100.13s | 22134.52±567.58 | 0.62±0.29 | 713.27±300.02 | 419.76s |
| PS-SL | PMVB | Gurobi | 22128.34±589.49 | 0.65±0.53 | 17.47±6.88 | 9.98s | 22246.79±566.82 | 0.12±0.14 | 34.80±19.43 | 63.65s | 22259.58±568.44 | 0.06±0.10 | 91.22±95.40 | 103.23s |
| PS-SL | Apollo | Gurobi | 22009.65±566.03 | 1.18±0.60 | 21.31±5.98 | 5.23s | 22245.16±575.21 | 0.12±0.17 | 55.07±22.84 | 45.86s | 22272.55±572.24 | 0.00±0.00 | 58.42±33.86 | 140.83s |
| CoCo | PS | Gurobi | 22172.76±587.69 | 0.45±0.35 | 14.44±4.78 | 10.03s | 22251.80±575.85 | 0.09±0.21 | 33.74±41.49 | 84.04s | 22272.53±572.25 | 0.00±0.00 | 44.79±59.03 | 228.88s |
| CoCo | ND | Gurobi | 21315.52±514.75 | 4.74±1.02(17) | 846.09±340.21 | 1.26s | 21495.96±504.79 | 3.93±0.88(17) | 8375.48±3589.70 | 1.78s | 21513.93±513.89 | 3.85±0.88(17) | 83663.75±36098.42 | 2.27s |
| CoCo | PMVB | Gurobi | N/A | N/A(0) | 1000.00±0.00 | 0.08s | N/A | N/A(0) | 10000.00±0.00 | 0.11s | N/A | N/A(0) | 100000.00±0.00 | 0.32s |
| CoCo | Apollo | Gurobi | 21963.66±587.22 | 1.39±0.70 | 24.53±7.03 | 5.24s | 22230.23±580.53 | 0.19±0.24 | 43.88±26.26 | 50.78s | 22271.95±571.40 | 0.00±0.02 | 79.54±49.41 | 145.47s |
| DiffILO | PS | Gurobi | 22184.62±583.61 | 0.40±0.30 | 14.69±4.80 | 10.02s | 22258.73±571.82 | 0.06±0.12 | 28.01±17.01 | 77.20s | 22272.27±571.65 | 0.00±0.01 | 32.82±28.19 | 181.83s |
| DiffILO | ND | Gurobi | 21264.67±600.53 | 4.53±0.87 | 52.20±8.89 | 10.09s | 21790.69±569.03 | 2.15±0.53(99) | 256.93±51.71 | 100.11s | 21920.25±564.42 | 1.58±0.38 | 1790.62±397.74 | 646.91s |
| DiffILO | PMVB | Gurobi | 19352.23±615.78 | 13.07±1.56(99) | 132.18±15.52 | 0.42s | 19352.23±615.78 | 13.07±1.56(99) | 1310.02±155.34 | 0.41s | 19352.23±615.78 | 13.07±1.56(99) | 13089.24±1553.95 | 0.39s |
| DiffILO | Apollo | Gurobi | 21895.91±602.22 | 1.69±0.85 | 29.63±8.11 | 5.26s | 22242.43±570.27 | 0.13±0.23 | 43.03±25.66 | 45.26s | 22272.55±572.24 | 0.00±0.00 | 47.64±41.62 | 126.60s |
| DDIM | PS | Gurobi | 22179.13±579.39 | 0.42±0.34 | 13.14±4.41 | 13.04s | 22254.74±571.89 | 0.08±0.19 | 28.94±23.10 | 80.67s | 22272.46±572.32 | 0.00±0.00 | 48.06±80.84 | 250.04s |
| DDIM | ND | Gurobi | 21923.19±590.18 | 1.58±0.70(99) | 21.17±8.12 | 13.06s | 22111.28±570.85 | 0.72±0.34 | 94.01±41.38 | 94.05s | 22137.91±572.36 | 0.60±0.28 | 669.77±314.33 | 315.10s |
| DDIM | PMVB | Gurobi | 21851.15±565.28 | 1.89±0.67 | 19.44±6.75 | 3.59s | 21851.15±565.28 | 1.89±0.67 | 189.55±67.07 | 12.78s | 21851.15±565.28 | 1.89±0.67 | 1890.74±670.33 | 16.98s |
| DDIM | Apollo | Gurobi | 22001.63±587.38 | 1.22±0.60 | 18.31±6.11 | 10.55s | 22235.73±572.09 | 0.17±0.19 | 36.68±22.02 | 71.79s | 22271.80±571.63 | 0.00±0.02 | 47.86±45.68 | 166.51s |
| FMIP | PS | Gurobi | 22177.24±580.91 | 0.43±0.34 | 15.33±4.85 | 12.11s | 22253.81±568.82 | 0.08±0.18 | 29.14±20.86 | 81.72s | 22272.34±572.14 | 0.00±0.01 | 46.83±72.03 | 232.82s |
| FMIP | ND | Gurobi | 21921.17±571.85 | 1.58±0.67 | 21.66±7.55 | 11.96s | 22100.55±573.39 | 0.77±0.34 | 98.23±38.40 | 100.09s | 22127.77±566.44 | 0.65±0.30 | 701.22±303.39 | 285.26s |
| FMIP | PMVB | Gurobi | 21867.16±559.05 | 1.85±0.70(99) | 18.91±7.04 | 2.84s | 21867.16±559.05 | 1.85±0.70(99) | 184.20±70.11 | 2.70s | 21867.16±559.05 | 1.85±0.70(99) | 1837.30±700.81 | 9.25s |
| FMIP | Apollo | Gurobi | 21975.75±571.76 | 1.33±0.75 | 19.65±7.02 | 8.94s | 22233.37±569.08 | 0.18±0.20 | 39.17±24.88 | 49.77s | 22272.26±572.09 | 0.00±0.01 | 44.91±35.16 | 153.89s |
| ConPaS | PS | Gurobi | 22165.12±572.39 | 0.48±0.37 | 17.61±4.98 | 10.03s | 22247.44±573.43 | 0.11±0.16 | 36.80±22.19 | 79.57s | 22264.05±572.95 | 0.04±0.08 | 84.20±88.61 | 269.73s |
| ConPaS | ND | Gurobi | 21296.84±596.31 | 4.38±0.90 | 53.12±8.97 | 10.08s | 21890.62±576.60 | 1.72±0.43 | 214.73±44.20 | 100.10s | 22059.41±569.46 | 0.96±0.26 | 1221.52±263.79 | 994.00s |
| ConPaS | PMVB | Gurobi | 14815.55±470.25 | 33.47±1.83 | 354.07±17.30 | 10.10s | 15231.78±439.50 | 31.61±1.30 | 3251.92±129.99 | 100.19s | 15613.33±431.01 | 29.89±1.32 | 30491.72±1222.80 | 632.86s |
| ConPaS | Apollo | Gurobi | 21934.41±577.13 | 1.52±0.73 | 24.29±7.10 | 5.22s | 22225.17±575.90 | 0.21±0.24 | 47.96±25.92 | 66.82s | 22263.91±572.83 | 0.04±0.08 | 87.51±87.80 | 152.08s |
| OPTFM | PS | Gurobi | 22164.30±581.67 | 0.49±0.34 | 15.20±4.51 | 10.10s | 22251.66±575.17 | 0.09±0.16 | 32.72±20.88 | 80.85s | 22272.55±572.24 | 0.00±0.00 | 46.89±57.19 | 231.10s |
| OPTFM | ND | Gurobi | 21202.32±615.54 | 4.81±1.07 | 67.86±16.04 | 7.90s | 21588.82±597.31 | 3.07±0.86 | 347.74±81.31 | 59.04s | 21684.30±588.40 | 2.64±0.78 | 2753.08±743.55 | 213.76s |
| OPTFM | PMVB | Gurobi | 22115.31±575.54 | 0.71±0.53 | 19.85±6.36 | 10.13s | 22251.61±576.35 | 0.09±0.15 | 38.86±20.34 | 78.65s | 22272.55±572.24 | 0.00±0.00 | 45.36±32.25 | 155.69s |
| OPTFM | Apollo | Gurobi | 21989.88±581.90 | 1.27±0.79 | 19.77±7.35 | 5.34s | 22239.18±572.11 | 0.15±0.22 | 37.54±25.07 | 46.71s | 22272.24±572.26 | 0.00±0.01 | 40.39±34.58 | 143.91s |
| SCIP | – | – | 21891.96 | 1.71±0.79 | 49.63±8.50 | 10.03s | 22266.31 | 0.03±0.08 | 75.05±20.70 | 90.81 | 22272.55 | 0.00±0.00 | 73.09±22.11 | 202.78s |
| Base-SL | PS | SCIP | 21864.71±575.36 | 1.83±0.78 | 54.07±7.59 | 10.29s | 22264.47±570.96 | 0.04±0.10 | 76.35±24.21 | 90.26s | 22272.55±572.24 | 0.00±0.00 | 80.68±26.57 | 217.62s |
| Base-SL | ND | SCIP | 19910.37±761.97 | 10.61±2.42 | 123.05±18.68 | 10.32s | 21833.23±574.20 | 1.97±1.07 | 399.09±211.48 | 100.34s | 22141.66±575.28 | 0.59±0.27 | 800.18±265.58 | 918.88s |
| Base-SL | PMVB | SCIP | 21792.62±557.20 | 2.15±0.69 | 38.65±12.33 | 8.12s | 21826.44±569.23 | 2.00±0.63 | 219.32±63.96 | 10.43s | 21826.44±569.23 | 2.00±0.63 | 2021.04±629.83 | 10.42s |
| Base-SL | Apollo | SCIP | 21745.74±573.38 | 2.36±0.88 | 52.52±8.37 | 5.87s | 22245.19±575.18 | 0.12±0.20 | 78.96±25.80 | 49.39s | 22271.68±571.19 | 0.00±0.04 | 82.08±52.13 | 213.87s |
| PS-SL | PS | SCIP | 21877.36±558.39 | 1.77±0.75 | 53.75±8.81 | 10.10s | 22263.99±572.59 | 0.04±0.12 | 80.69±23.43 | 92.25s | 22272.55±572.24 | 0.00±0.00 | 78.93±37.88 | 213.25s |
| PS-SL | ND | SCIP | 19742.72±682.40 | 11.36±1.85 | 129.24±13.85 | 9.94s | 21846.83±576.94 | 1.91±0.76 | 456.12±148.09 | 97.79s | 22116.98±569.13 | 0.70±0.32 | 955.07±330.44 | 851.45s |
| PS-SL | PMVB | SCIP | 21801.38±571.58 | 2.11±1.08 | 54.38±10.84 | 10.09s | 22252.48±568.10 | 0.09±0.11 | 84.70±25.29 | 81.98s | 22259.58±568.44 | 0.06±0.10 | 135.18±96.29 | 175.33s |
| PS-SL | Apollo | SCIP | 21796.93±579.59 | 2.13±0.74 | 54.61±8.15 | 5.24s | 22242.55±570.58 | 0.13±0.22 | 82.95±28.83 | 49.30s | 22272.26±572.25 | 0.00±0.01 | 80.77±29.17 | 203.49s |
| CoCo | PS | SCIP | 21892.82±591.74 | 1.71±0.75 | 53.95±7.97 | 10.18s | 22262.96±571.86 | 0.04±0.13 | 76.32±23.21 | 90.87s | 22272.55±572.24 | 0.00±0.00 | 77.62±26.68 | 210.23s |
| CoCo | ND | SCIP | 19299.11±738.55 | 13.74±1.12(17) | 863.99±300.96 | 1.84s | 21447.27±489.91 | 4.15±0.78(17) | 8384.50±3569.73 | 2.50s | 21513.93±513.89 | 3.85±0.88(17) | 83669.45±36085.81 | 4.15s |
| CoCo | PMVB | SCIP | N/A | N/A(0) | 1000.00±0.00 | 0.16s | N/A | N/A(0) | 10000.00±0.00 | 0.14s | N/A | N/A(0) | 100000.00±0.00 | 0.15s |
| CoCo | Apollo | SCIP | 21774.74±561.88 | 2.23±0.82 | 52.61±8.08 | 5.28s | 22242.96±577.22 | 0.13±0.17 | 79.27±22.73 | 49.05s | 22272.55±572.24 | 0.00±0.00 | 79.51±24.02 | 211.31s |
| DiffILO | PS | SCIP | 21767.42±629.33 | 2.27±0.98 | 57.07±9.26 | 10.21s | 22261.27±572.75 | 0.05±0.14 | 85.58±27.18 | 92.57s | 22272.55±572.24 | 0.00±0.00 | 90.23±51.93 | 234.77s |
| DiffILO | ND | SCIP | 19415.91±623.32 | 12.83±1.36 | 140.44±13.48 | 10.17s | 20888.65±750.64 | 6.21±2.34 | 795.92±250.67 | 100.16s | 21862.26±569.97 | 1.84±0.47 | 2240.25±457.05 | 989.61s |
| DiffILO | PMVB | SCIP | 19356.91±614.39 | 13.09±1.55 | 142.48±15.22 | 3.08s | 19357.78±615.16 | 13.09±1.55 | 1320.04±154.99 | 3.24s | 19357.78±615.16 | 13.09±1.55 | 13099.71±1553.58 | 3.68s |
| DiffILO | Apollo | SCIP | 21615.36±609.49 | 2.95±1.17 | 59.16±9.95 | 5.37s | 22240.17±575.20 | 0.15±0.25 | 88.93±30.49 | 49.17s | 22272.55±572.24 | 0.00±0.00 | 90.78±57.71 | 217.12s |
| DDIM | PS | SCIP | 21863.37±584.37 | 1.84±0.85 | 53.55±9.60 | 12.01s | 22263.67±571.87 | 0.04±0.11 | 78.22±26.30 | 93.32s | 22272.55±572.24 | 0.00±0.00 | 82.15±33.97 | 231.61s |
| DDIM | ND | SCIP | 20369.22±940.85 | 8.53±3.84 | 108.00±29.97 | 12.32s | 21939.17±574.14 | 1.50±0.63 | 259.43±112.52 | 102.17s | 22129.96±571.82 | 0.64±0.29 | 840.99±354.47 | 796.26s |
| DDIM | PMVB | SCIP | 21834.31±566.59 | 1.97±0.69 | 31.93±7.54 | 10.62s | 21851.19±565.27 | 1.89±0.67 | 202.59±67.36 | 14.14s | 21851.19±565.27 | 1.89±0.67 | 1903.58±670.66 | 13.19s |
| DDIM | Apollo | SCIP | 21755.68±581.12 | 2.32±0.87 | 52.24±7.96 | 8.88s | 22236.86±575.98 | 0.16±0.24 | 81.59±30.51 | 54.63s | 22272.55±572.24 | 0.00±0.00 | 83.18±38.06 | 219.93s |
| FMIP | PS | SCIP | 21841.19±595.27 | 1.94±0.94 | 50.69±8.69 | 25.83s | 22262.79±569.15 | 0.04±0.14 | 82.17±28.05 | 108.50s | 22272.55±572.24 | 0.00±0.00 | 80.48±32.42 | 607.16s |
| FMIP | ND | SCIP | 20340.82±1008.49 | 8.68±3.74 | 107.91±29.50 | 25.80s | 21931.93±559.60 | 1.53±0.70 | 266.63±124.22 | 105.79s | 22119.15±565.76 | 0.69±0.31 | 882.06±341.75 | 804.18s |
| FMIP | PMVB | SCIP | 21844.67±563.42 | 1.92±0.75 | 30.96±8.38 | 24.47s | 21862.83±557.92 | 1.84±0.70 | 196.53±71.03 | 23.19s | 21862.83±557.92 | 1.84±0.70 | 1849.81±701.73 | 13.13s |
| FMIP | Apollo | SCIP | 21782.90±587.75 | 2.20±0.82 | 51.95±8.49 | 36.38s | 22247.85±574.44 | 0.11±0.17 | 75.57±24.28 | 82.90s | 22272.22±572.32 | 0.00±0.01 | 77.34±25.82 | 589.52s |
| ConPaS | PS | SCIP | 21694.78±646.42 | 2.60±1.21 | 62.99±10.52 | 10.44s | 22254.00±574.98 | 0.08±0.13 | 93.72±29.64 | 94.06s | 22264.39±572.61 | 0.04±0.08 | 132.01±86.93 | 263.07s |
| ConPaS | ND | SCIP | 19235.62±640.39 | 13.64±1.63 | 152.25±15.69 | 10.45s | 21525.08±584.95 | 3.35±1.13 | 417.83±122.25 | 100.13s | 22009.67±566.16 | 1.18±0.29 | 1516.20±284.42 | 1000.32s |
| ConPaS | PMVB | SCIP | 14483.39±518.53 | 34.97±1.90 | 373.15±15.85 | 10.40s | 15312.86±417.79 | 31.24±1.25 | 3240.96±121.06 | 100.10s | 15624.08±430.09 | 29.84±1.33 | 30262.87±1242.63 | 912.65s |
| ConPaS | Apollo | SCIP | 21578.38±615.21 | 3.12±1.27 | 62.40±10.82 | 5.88s | 22217.49±578.94 | 0.25±0.27 | 100.59±33.52 | 48.63s | 22272.55±572.61 | 0.04±0.08 | 133.61±88.07 | 239.77s |
| OPTFM | PS | SCIP | 21876.87±558.39 | 1.77±0.76 | 53.3244 ± 8.72 | 10.16s | 22264.60±573.16 | 0.04±0.09 | 76.09±24.13 | 90.18s | 22272.55±572.24 | 0.00±0.00 | 80.06±31.68 | 214.66s |
| OPTFM | ND | SCIP | 19246.44±888.06 | 13.59±3.33 | 165.16±20.54 | 8.06s | 21343.06±563.76 | 4.17±0.97 | 599.34±97.40 | 71.51s | 21660.20±589.05 | 2.75±0.81 | 3022.92±734.71 | 458.98s |
| OPTFM | PMVB | SCIP | 21848.90±568.84 | 1.90±0.85 | 55.42±8.82 | 10.15s | 22263.97±572.14 | 0.04±0.11 | 80.71±23.85 | 90.82s | 22272.55±572.24 | 0.00±0.00 | 84.64±40.68 | 220.64s |
| OPTFM | Apollo | SCIP | 21810.03±568.78 | 2.07±0.77 | 53.98±8.84 | 5.38s | 22243.05±579.70 | 0.13±0.21 | 79.50±27.81 | 49.14s | 22272.55±572.24 | 0.00±0.00 | 78.00±27.35 | 198.18s |
| COPT | – | – | 22032.87 | 1.08±0.80 | 61.62±8.76 | 10.12s | 22271.15 | 0.01±0.03 | 73.37±16.97 | 72.99s | 22272.55 | 0.00±0.00 | 76.62±18.75 | 126.64s |
| Base-SL | PS | COPT | 21911.25±615.44 | 1.62±0.91 | 83.75±10.76 | 10.41s | 22271.22±572.00 | 0.01±0.03 | 99.16±20.75 | 77.63s | 22272.55±572.24 | 0.00±0.00 | 92.31±21.00 | 126.30s |
| Base-SL | ND | COPT | 19096.05±747.18 | 14.27±2.40 | 160.82±23.97 | 10.28s | 22016.67±563.66 | 1.15±0.45 | 178.52±57.82 | 100.65s | 22162.14±574.36 | 0.50±0.24 | 636.30±233.29 | 625.57s |
| Base-SL | PMVB | COPT | 21826.44±569.23 | 2.00±0.63 | 153.48±29.74 | 5.03s | 21826.44±569.23 | 2.00±0.63 | 316.80±99.34 | 4.59s | 21826.44±569.23 | 2.00±0.63 | 2107.77±645.32 | 4.26s |
| Base-SL | Apollo | COPT | 21752.54±594.75 | 2.34±0.79 | 86.29±10.17 | 6.23s | 22259.33±572.09 | 0.06±0.13 | 111.23±24.47 | 49.14s | 22272.55±572.24 | 0.00±0.00 | 96.64±23.24 | 130.51s |
| PS-SL | PS | COPT | 21956.46±621.91 | 1.42±1.04 | 76.30±10.19 | 10.28s | 22270.45±570.90 | 0.01±0.04 | 91.01±22.54 | 75.98s | 22272.55±572.24 | 0.00±0.00 | 91.61±25.71 | 123.32s |
| PS-SL | ND | COPT | 19074.53±660.08 | 14.36±1.77 | 169.40±25.82 | 10.18s | 21988.22±587.81 | 1.28±0.44 | 204.37±60.52 | 98.30s | 22135.24±567.21 | 0.62±0.29 | 762.14±299.83 | 505.51s |
| PS-SL | PMVB | COPT | 22033.94±586.43 | 1.07±0.71 | 75.14±8.66 | 10.27s | 22256.50±566.81 | 0.07±0.12 | 89.97±17.71 | 61.88s | 22259.58±568.44 | 0.06±0.10 | 142.35±97.59 | 91.96s |
| PS-SL | Apollo | COPT | 21756.27±616.28 | 2.32±0.98 | 78.75±10.03 | 5.85s | 22263.14±570.53 | 0.04±0.12 | 91.65±21.82 | 48.63s | 22272.55±572.24 | 0.00±0.00 | 90.71±28.42 | 124.02s |
| CoCo | PS | COPT | 21860.64±612.48 | 1.85±1.02 | 96.09±13.64 | 10.48s | 22270.06±571.27 | 0.01±0.04 | 105.54±24.69 | 78.71s | 22272.55±572.24 | 0.00±0.00 | 95.99±22.59 | 129.59s |
| CoCo | ND | COPT | 18487.00±696.19 | 17.28±2.32(18) | 922.73±189.81 | 10.30s | 21478.00±495.14 | 4.01±0.86(17) | 8497.94±3322.58 | 19.93s | 21513.93±513.89 | 3.85±0.88(17) | 83731.17±35949.25 | 11.43s |
| CoCo | PMVB | COPT | N/A | N/A(0) | 1000.00±0.00 | 0.71s | N/A | N/A(0) | 10000.00±0.00 | 0.42s | N/A | N/A(0) | 100000.00±0.00 | 0.42s |
| CoCo | Apollo | COPT | 21748.24±590.90 | 2.35±0.84 | 95.49±12.24 | 6.11s | 22252.84±568.93 | 0.09±0.20 | 111.87±25.86 | 48.62s | 22272.55±572.24 | 0.00±0.00 | 97.17±21.89 | 130.41s |
| DiffILO | PS | COPT | 21960.49±641.99 | 1.40±1.25 | 80.75±10.63 | 10.25s | 22268.65±569.92 | 0.02±0.08 | 95.06±21.29 | 76.65s | 22272.55±572.24 | 0.00±0.00 | 97.97±26.49 | 129.44s |
| DiffILO | ND | COPT | 18871.16±699.19 | 15.27±2.30 | 168.19±20.60 | 10.18s | 21614.23±593.65 | 2.96±0.70 | 359.17±75.24 | 100.49s | 21923.66±564.40 | 1.57±0.38 | 1839.46±386.97 | 739.13s |
| DiffILO | PMVB | COPT | 19357.78±615.16 | 13.09±1.55 | 171.10±15.21 | 2.87s | 19357.78±615.16 | 13.09±1.55 | 1348.01±154.79 | 2.88s | 19357.78±615.16 | 13.09±1.55 | 13128.39±1553.11 | 2.70s |
| DiffILO | Apollo | COPT | 21569.09±570.98 | 3.16±0.92 | 86.63±10.23 | 6.61s | 22262.04±571.24 | 0.05±0.10 | 93.68±20.91 | 48.30s | 22272.55±572.24 | 0.00±0.00 | 94.35±20.99 | 126.87s |
| DDIM | PS | COPT | 21915.47±644.98 | 1.61±1.13 | 87.43±12.28 | 12.55s | 22271.09±573.26 | 0.01±0.03 | 104.92±23.48 | 81.99s | 22272.55±572.24 | 0.00±0.00 | 98.75±24.90 | 134.38s |
| DDIM | ND | COPT | 19225.34±696.88 | 13.69±2.07 | 153.21±18.41 | 12.67s | 22043.28±577.35 | 1.03±0.48 | 149.19±57.72 | 102.47s | 22138.42±571.83 | 0.60±0.28 | 699.40±308.73 | 403.93s |
| DDIM | PMVB | COPT | 21851.19±565.27 | 1.89±0.67 | 74.90±11.77 | 20.71s | 21851.19±565.27 | 1.89±0.67 | 244.35±67.01 | 6.45s | 21851.19±565.27 | 1.89±0.67 | 1943.68±671.17 | 7.16s |
| DDIM | Apollo | COPT | 21766.12±597.39 | 2.27±0.88 | 88.08±12.37 | 27.18s | 22257.98±572.64 | 0.07±0.15 | 109.62±25.23 | 53.80s | 22272.55±572.24 | 0.00±0.00 | 97.21±23.95 | 144.78s |
| FMIP | PS | COPT | 21963.50±594.78 | 1.39±0.97 | 76.35±8.93 | 19.15s | 22270.10±570.63 | 0.01±0.07 | 87.19±21.29 | 86.66s | 22272.55±572.24 | 0.00±0.00 | 88.66±24.00 | 122.72s |
| FMIP | ND | COPT | 19291.50±659.36 | 13.39±1.86 | 149.68±18.92 | 19.47s | 22041.63±572.62 | 1.04±0.45 | 143.17±54.72 | 111.25s | 22127.80±566.44 | 0.65±0.30 | 731.34±303.15 | 380.23s |
| FMIP | PMVB | COPT | 21862.83±557.92 | 1.84±0.70 | 63.02±8.89 | 17.12s | 21862.83±557.92 | 1.84±0.70 | 237.14±70.72 | 13.01s | 21862.83±557.92 | 1.84±0.70 | 1883.08±700.72 | 5.62s |
| FMIP | Apollo | COPT | 21755.57±577.80 | 2.32±0.81 | 77.26±9.17 | 28.59s | 22261.80±572.81 | 0.05±0.10 | 91.04±20.66 | 52.12s | 22272.55±572.24 | 0.00±0.00 | 89.82±19.53 | 126.02s |
| ConPaS | PS | COPT | 21918.81±643.56 | 1.59±1.02 | 88.02±12.41 | 10.39s | 22261.53±573.25 | 0.05±0.10 | 121.56±30.15 | 63.92s | 22264.39±572.61 | 0.04±0.08 | 153.06±84.96 | 160.23s |
| ConPaS | ND | COPT | 18412.42±758.44 | 17.33±2.64 | 188.27±25.11 | 10.20s | 20038.01±976.29 | 10.05±3.40 | 1139.94±364.72 | 100.45s | 22055.00±571.22 | 0.98±0.26 | 1302.31±283.47 | 998.16s |
| ConPaS | PMVB | COPT | 14460.10±549.25 | 35.07±2.00 | 410.05±16.79 | 10.32s | 15446.16±434.29 | 30.64±1.31 | 3349.33±132.41 | 100.53s | 15624.75±429.98 | 29.84±1.33 | 30258.97±1287.70 | 678.58s |
| ConPaS | Apollo | COPT | 21627.84±631.56 | 2.90±1.13 | 92.02±11.98 | 6.05s | 22249.14±569.00 | 0.10±0.18 | 114.20±28.22 | 49.07s | 22264.39±572.61 | 0.04±0.08 | 159.38±85.61 | 160.43s |
| OPTFM | PS | COPT | 21987.69±615.60 | 1.28±1.04 | 82.14±9.49 | 10.76s | 22269.48±570.96 | 0.01±0.07 | 91.64±21.26 | 75.97s | 22272.55±572.24 | 0.00±0.00 | 92.29±22.68 | 123.63s |
| OPTFM | ND | COPT | 18239.96±706.81 | 18.11±2.26 | 265.20±84.09 | 10.62s | 21499.70±587.27 | 3.47±0.86 | 493.61±120.58 | 68.87s | 21691.82±588.53 | 2.61±0.79 | 2825.55±764.60 | 244.81s |
| OPTFM | PMVB | COPT | 22016.42±597.33 | 1.15±0.93 | 65.37±10.47 | 10.76s | 22270.74±571.12 | 0.01±0.03 | 74.64±22.80 | 75.54s | 22272.55±572.24 | 0.00±0.00 | 76.19±22.77 | 122.24s |
| OPTFM | Apollo | COPT | 21731.03±595.03 | 2.43±0.87 | 85.36±9.44 | 6.63s | 22262.13±572.79 | 0.05±0.12 | 91.73±22.19 | 47.49s | 22272.55±572.24 | 0.00±0.00 | 93.83±22.64 | 123.54s |

non-monotonicity of PIs does not contradict the monotonicity of PI along a fixed anytime trajectory. The complete *anytime behavior* can instead be examined from the corresponding primal-integral curves in Figures 15 and 16.

Table 19: Complete solver-backed enforcer results on CFL under different time budgets.

| Learner | Enforcer | Backend | 1s Obj↓ | 1s Gap↓ | 1s PI↓ | 1s Time | 10s Obj↓ | 10s Gap↓ | 10s PI↓ | 10s Time | 100s Obj↓ | 100s Gap↓ | 100s PI↓ | 100s Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BKS | – | – | 11388.79* | 0.00±0.00 | – | – | 11388.79* | 0.00±0.00 | – | – | 11388.79* | 0.00±0.00 | – | – |
| Gurobi | – | – | 11510.30 | 1.07±1.23 | 16.79±2.28 | 1.10s | 11399.81 | 0.10±0.25 | 19.89±5.19 | 5.81s | 11388.79 | 0.00±0.00 | 20.24±5.77 | 8.07s |
| Base-SL | PS | Gurobi | 16602.78±113.55 | 45.78±8.24(95) | 55.74±1.58 | 0.72s | 16587.66±115.68 | 45.65±8.25(95) | 486.06±131.40 | 1.14s | 16587.66±115.68 | 45.65±8.25(95) | 4790.16±133.21 | 1.06s |
| Base-SL | ND | Gurobi | 45889.07±2045.64 | 303.13±13.88 | 100.00±0.00 | 1.42s | 11541.73±494.01 | 1.39±1.04(99) | 27.55±11.34 | 10.43s | 11464.12±485.03 | 0.66±0.68 | 90.43±68.67 | 48.54s |
| Base-SL | PMVB | Gurobi | 12004.43±649.97 | 5.42±3.63(99) | 12.33±11.07 | 0.92s | 11999.77±647.61 | 5.38±3.61(99) | 69.30±100.58 | 1.09s | 11999.77±647.61 | 5.38±3.61(99) | 639.07±1007.21 | 1.11s |
| Base-SL | Apollo | Gurobi | 19225.96±4585.67 | 68.98±41.10(75) | 83.17±16.69 | 1.47s | 18735.95±4168.04 | 64.82±37.85(99) | 602.87±204.63 | 1.71s | 18735.95±4168.04 | 64.82±37.85(99) | 5865.59±2124.29 | 1.84s |
| PS-SL | PS | Gurobi | 16147.07±927.49 | 41.99±5.98(90) | 55.92±16.02 | 0.76s | 16118.66±922.16 | 41.74±5.86(90) | 484.07±180.71 | 1.04s | 16118.66±922.16 | 41.74±5.86(90) | 4764.77±1831.38 | 1.04s |
| PS-SL | ND | Gurobi | 46086.72±2222.72 | 304.76±12.80 | 100.00±0.00 | 1.14s | 11545.41±496.63 | 1.37±0.89 | 28.35±10.17 | 10.13s | 11460.10±491.58 | 0.62±0.55 | 93.73±57.47 | 47.03s |
| PS-SL | PMVB | Gurobi | 11676.94±537.14 | 2.54±2.45 | 9.54±3.47 | 0.94s | 11635.14±534.09 | 2.18±2.49 | 29.51±24.35 | 1.97s | 11634.96±534.08 | 2.17±2.49 | 225.20±248.60 | 2.39s |
| PS-SL | Apollo | Gurobi | 16154.14±928.96 | 41.99±5.66(89) | 57.98±15.51 | 0.87s | 16124.25±924.72 | 41.78±5.86(90) | 486.52±179.94 | 1.20s | 16124.25±924.72 | 41.78±5.86(90) | 4771.13±1829.29 | 1.22s |
| CoCo | PS | Gurobi | 29515.64±2029.03 | 159.09±14.18(98) | 100.00±0.00 | 0.35s | 29515.64±2029.03 | 159.09±14.18(98) | 1000.00±0.00 | 0.34s | 29515.64±2029.03 | 159.09±14.18(98) | 10000.00±0.00 | 0.35s |
| CoCo | ND | Gurobi | 50779.37±2085.37 | 346.14±14.88 | 100.00±0.00 | 1.15s | 16810.02±2520.52 | 47.81±22.84 | 549.00±209.38 | 8.61s | 16899.23±2461.67 | 48.63±22.64 | 4902.66±2167.27 | 14.17s |
| CoCo | PMVB | Gurobi | 40575.07±1721.87 | 257.05±11.21(92) | 100.00±0.00 | 0.25s | 40575.07±1721.87 | 257.05±11.21(92) | 1000.00±0.00 | 0.26s | 40575.07±1721.87 | 257.05±11.21(92) | 10000.00±0.00 | 0.31s |
| CoCo | Apollo | Gurobi | 28074.15±2000.06 | 146.50±14.09 | 100.00±0.00 | 0.71s | 28073.68±2000.25 | 146.50±14.09 | 1000.00±0.00 | 0.72s | 28073.68±2000.25 | 146.50±14.09 | 10000.00±0.00 | 0.82s |
| DiffILO | PS | Gurobi | 19607.72±2040.71 | 71.79±15.15(97) | 78.98±11.26 | 0.49s | 19607.72±2040.71 | 71.79±15.15(97) | 755.60±132.18 | 0.45s | 19607.72±2040.71 | 71.79±15.15(97) | 7522.38±1341.09 | 0.66s |
| DiffILO | ND | Gurobi | 46842.79±2020.77 | 311.58±15.56 | 100.00±0.00 | 1.20s | 12120.43±620.21 | 6.51±3.27(97) | 93.00±36.42 | 10.20s | 11885.66±546.91 | 4.40±2.16(98) | 488.25±218.99 | 18.32s |
| DiffILO | PMVB | Gurobi | 16516.70±2125.52 | 44.80±15.59 | 46.86±15.62 | 0.38s | 16516.20±2126.68 | 44.79±15.60 | 450.01±156.02 | 0.36s | 16516.20±2126.68 | 44.79±15.60 | 4481.28±1559.62 | 0.41s |
| DiffILO | Apollo | Gurobi | 20126.09±2193.27 | 76.40±17.39(97) | 82.81±11.12 | 0.87s | 20126.09±2193.27 | 76.40±17.39(97) | 790.72±135.77 | 1.28s | 20126.09±2193.27 | 76.40±17.39(97) | 7854.11±1392.61 | 1.12s |
| DDIM | PS | Gurobi | 13494.75±1170.21 | 19.77±10.61(39) | 96.71±12.12 | 4.80s | 13488.56±1150.87 | 19.72±10.46(39) | 959.26±149.16 | 4.20s | 13488.56±1150.87 | 19.72±10.46(39) | 9584.73±1519.82 | 3.44s |
| DDIM | ND | Gurobi | 46071.82±1997.94 | 304.74±13.61 | 100.00±0.00 | 4.08s | 11573.30±482.28 | 1.63±1.02 | 30.99±11.27 | 12.74s | 11491.12±479.21 | 0.90±0.64 | 115.76±66.69 | 40.23s |
| DDIM | PMVB | Gurobi | 12608.14±641.40 | 10.77±4.96 | 12.36±4.87 | 3.20s | 12608.14±641.40 | 10.77±4.96 | 109.38±48.90 | 2.90s | 12608.14±641.40 | 10.77±4.96 | 1078.74±495.44 | 4.11s |
| DDIM | Apollo | Gurobi | 13737.48±1504.95 | 22.29±12.05(33) | 95.73±13.62 | 6.88s | 14328.00±2254.31 | 27.60±19.83(34) | 930.54±182.43 | 6.66s | 14328.00±2254.31 | 27.60±19.83(34) | 9239.78±1974.00 | 8.78s |
| FMIP | PS | Gurobi | 17029.25±2911.77 | 49.84±24.08(93) | 70.28±19.27 | 3.53s | 16948.76±2929.55 | 49.13±24.24(93) | 613.45±246.84 | 4.58s | 16944.83±2931.31 | 49.10±24.26(93) | 6043.84±2528.68 | 5.28s |
| FMIP | ND | Gurobi | 46300.27±2224.31 | 306.70±14.81 | 100.00±0.00 | 3.68s | 11547.13±491.10 | 1.34±0.85(99) | 26.95±9.14 | 12.10s | 11471.50±485.56 | 0.73±0.63 | 95.08±63.90 | 34.38s |
| FMIP | PMVB | Gurobi | 12275.72±712.60 | 7.80±4.53 | 9.50±4.01 | 2.78s | 12275.16±713.06 | 7.79±4.54 | 79.47±44.79 | 2.32s | 12275.16±713.06 | 7.79±4.54 | 780.78±453.06 | 2.28s |
| FMIP | Apollo | Gurobi | 17004.43±3185.91 | 49.92±26.18(90) | 70.30±20.24 | 5.90s | 17288.73±3060.14 | 51.71±24.37(95) | 577.87±221.21 | 6.02s | 17287.69±3079.21 | 51.70±24.53(94) | 5586.59±2246.61 | 6.43s |
| ConPaS | PS | Gurobi | N/A | N/A(0) | 100.00±0.00 | 0.24s | N/A | N/A(0) | 1000.00±0.00 | 0.34s | N/A | N/A(0) | 10000.00±0.00 | 0.24s |
| ConPaS | ND | Gurobi | 46216.94±2123.29 | 306.04±15.41 | 100.00±0.00 | 1.17s | 11715.37±487.82 | 2.88±1.22 | 49.47±12.91 | 10.18s | 11509.46±480.88 | 1.06±0.47 | 169.45±51.32 | 99.75s |
| ConPaS | PMVB | Gurobi | N/A | N/A(0) | 100.00±0.00 | 0.30s | N/A | N/A(0) | 1000.00±0.00 | 0.26s | N/A | N/A(0) | 10000.00±0.00 | 0.26s |
| ConPaS | Apollo | Gurobi | N/A | N/A(0) | 100.00±0.00 | 0.40s | N/A | N/A(0) | 1000.00±0.00 | 0.42s | N/A | N/A(0) | 10000.00±0.00 | 0.40s |
| OPTFM | PS | Gurobi | 27390.63±2845.80 | 140.82±26.06 | 99.87±0.75 | 0.65s | 27390.05±2845.38 | 140.82±26.06 | 998.44±9.20 | 0.85s | 27389.96±2845.54 | 140.82±26.06 | 9983.99±94.29 | 1.70s |
| OPTFM | ND | Gurobi | 48248.23±2200.51 | 323.98±18.15 | 100.00±0.00 | 1.24s | 12605.20±724.44 | 10.85±7.80(98) | 154.56±83.97 | 9.14s | 12528.29±812.27 | 10.21±8.97 | 1081.95±892.40 | 17.83s |
| OPTFM | PMVB | Gurobi | 11502.54±498.18 | 1.00±1.12 | 18.49±2.44 | 1.23s | 11402.50±482.44 | 0.12±0.27 | 22.11±6.18 | 6.15s | 11387.20±482.36 | 0.00±0.00(98) | 22.77±7.65 | 10.03s |
| OPTFM | Apollo | Gurobi | 27403.31±2849.54 | 140.93±26.07 | 99.86±0.80 | 0.98s | 27391.28±2846.34 | 140.83±26.05 | 998.41±9.35 | 1.06s | 27391.19±2846.49 | 140.82±26.05 | 9983.78±95.36 | 1.43s |
| SCIP | – | – | 19901.27 | 74.57±76.86 | 91.22±9.93 | 1.09s | 11830.92 | 3.89±2.84 | 251.09±62.50 | 10.10s | 11529.09 | 1.22±1.46 | 475.65±177.54 | 87.76s |
| Base-SL | PS | SCIP | 17033.21±1371.39 | 49.99±10.04(91) | 71.94±12.24 | 1.49s | 16602.72±1101.00 | 45.86±8.06(96) | 511.36±128.32 | 4.16s | 16578.41±1113.51 | 45.65±8.20(96) | 4820.55±1323.70 | 5.65s |
| Base-SL | ND | SCIP | N/A | N/A(0) | 100.00±0.00 | 1.50s | 12212.85±807.32 | 7.18±4.14 | 126.05±73.00 | 10.54s | 11596.46±530.60 | 1.82±1.35 | 349.79±179.21 | 98.98s |
| Base-SL | PMVB | SCIP | 12952.85±1301.27 | 13.70±9.35(97) | 67.55±21.90 | 1.48s | 12059.38±664.47 | 5.90±3.63(99) | 147.62±96.59 | 6.00s | 12012.96±647.46 | 5.50±3.54(99) | 735.29±992.37 | 15.29s |
| Base-SL | Apollo | SCIP | 20891.37±7075.50 | 84.14±62.80(85) | 80.46±16.18 | 1.73s | 18756.65±4156.60 | 65.00±37.76(99) | 609.75±203.39 | 3.60s | 18735.92±4168.02 | 64.82±37.85(99) | 5869.99±2123.81 | 3.67s |
| PS-SL | PS | SCIP | 16813.63±1143.14 | 47.94±8.91(85) | 73.90±13.64 | 1.18s | 16188.89±966.00 | 42.34±6.03(90) | 518.39±173.42 | 5.29s | 16118.62±922.17 | 41.74±5.86(90) | 4807.76±1817.78 | 8.45s |
| PS-SL | ND | SCIP | N/A | N/A(0) | 100.00±0.00 | 1.11s | 12307.83±792.50 | 8.04±4.58 | 136.89±88.95 | 10.51s | 11614.46±529.88 | 2.00±1.37 | 378.98±171.18 | 98.61s |
| PS-SL | PMVB | SCIP | 13279.96±1685.21 | 16.54±13.30(99) | 78.25±18.19 | 1.22s | 11876.30±583.83 | 4.28±2.77 | 163.52±54.26 | 9.41s | 11656.31±534.88 | 2.36±2.46 | 448.99±234.52 | 48.60s |
| PS-SL | Apollo | SCIP | 17483.36±1437.60 | 53.62±11.19(75) | 76.60±15.22 | 1.17s | 16255.90±996.38 | 42.93±6.32(90) | 521.23±172.71 | 4.14s | 16124.81±923.75 | 41.79±5.85(90) | 4811.89±1816.03 | 8.98s |
| CoCo | PS | SCIP | 29545.15±2014.13 | 159.43±13.99 | 100.00±0.00 | 1.31s | 29516.66±2011.63 | 159.18±14.07 | 1000.00±0.00 | 1.48s | 29516.39±2011.73 | 159.18±14.07 | 10000.00±0.00 | 1.49s |
| CoCo | ND | SCIP | N/A | N/A(0) | 100.00±0.00 | 1.83s | 17770.23±2472.11 | 56.26±22.60 | 625.11±198.11 | 10.84s | 16936.28±2479.39 | 48.92±22.47 | 5084.33±2107.38 | 28.25s |
| CoCo | PMVB | SCIP | 40574.96±1721.89 | 257.05±11.21(92) | 100.00±0.00 | 1.24s | 40574.96±1721.89 | 257.05±11.21(92) | 1000.00±0.00 | 1.22s | 40574.96±1721.89 | 257.05±11.21(92) | 10000.00±0.00 | 1.30s |
| CoCo | Apollo | SCIP | 28176.60±2003.99 | 147.39±13.92 | 100.00±0.00 | 2.11s | 28073.57±2000.27 | 146.50±14.09 | 1000.00±0.00 | 2.32s | 28073.57±2000.27 | 146.50±14.09 | 10000.00±0.00 | 2.57s |
| DiffILO | PS | SCIP | 19920.35±2309.53 | 74.49±17.41(97) | 89.33±7.99 | 1.50s | 19607.71±2040.71 | 71.79±15.15(97) | 766.46±126.36 | 1.84s | 19607.71±2040.71 | 71.79±15.15(97) | 7533.82±1334.87 | 1.74s |
| DiffILO | ND | SCIP | 18213.93±2447.09 | 59.57±18.16(94) | 100.00±0.00 | 2.52s | 12949.09±811.53 | 13.71±5.41 | 171.05±55.17 | 10.50s | 12072.42±618.62 | 5.99±2.72 | 844.34±341.50 | 76.37s |
| DiffILO | PMVB | SCIP | 16542.47±2104.62 | 45.04±15.45 | 59.89±11.58 | 1.06s | 16517.27±2124.82 | 44.80±15.58 | 461.91±149.73 | 1.29s | 16516.18±2126.68 | 44.79±15.60 | 4493.91±1553.11 | 1.27s |
| DiffILO | Apollo | SCIP | 20900.08±3008.86 | 83.09±24.27(97) | 89.51±8.87 | 1.83s | 20127.97±2196.85 | 76.41±17.41(97) | 794.99±133.32 | 2.10s | 20126.09±2193.27 | 76.40±17.39(97) | 7859.13±1389.49 | 1.51s |
| DDIM | PS | SCIP | 13348.77±903.19 | 17.76±7.51(36) | 98.97±6.26 | 7.24s | 13491.23±1155.21 | 19.74±10.53(39) | 963.24±135.40 | 5.39s | 13488.56±1150.87 | 19.72±10.46(39) | 9588.56±1506.25 | 7.89s |
| DDIM | ND | SCIP | 13205.91±702.68 | 16.49±5.03(37) | 100.00±0.00 | 7.45s | 12035.52±584.03 | 5.69±2.95 | 100.88±41.87 | 12.95s | 11586.012±501.45 | 1.73±1.22 | 314.02±149.24 | 99.59s |
| DDIM | PMVB | SCIP | 12790.84±638.53 | 12.39±4.62(99) | 51.20±20.52 | 3.52s | 12618.13±637.50 | 10.86±4.87 | 152.28±43.34 | 4.24s | 12608.13±641.41 | 10.77±4.96 | 1126.59±476.88 | 4.67s |
| DDIM | Apollo | SCIP | 13475.99±1855.79 | 19.71±14.71(32) | 99.98±0.22 | 7.45s | 14291.29±2337.81 | 27.26±20.44(34) | 943.69±156.60 | 10.74s | 14328.00±2254.31 | 27.60±19.83(34) | 9242.67±1966.42 | 6.86s |
| FMIP | PS | SCIP | 16859.90±3826.60 | 48.56±32.44(80) | 91.77±11.69 | 17.72s | 17096.77±2876.66 | 50.46±23.89(93) | 680.50±211.66 | 21.27s | 16953.19±2925.62 | 49.17±24.21(93) | 6145.95±2452.01 | 36.18s |
| FMIP | ND | SCIP | 12768.75±727.04 | 12.28±4.70(50) | 100.00±0.00 | 18.36s | 12683.78±5846.35 | 11.21±49.59 | 130.72±114.68 | 26.00s | 11540.63±499.25 | 1.33±0.91 | 250.02±108.09 | 112.23s |
| FMIP | PMVB | SCIP | 12411.60±723.71 | 8.97±4.20 | 24.87±16.81 | 16.57s | 12292.27±700.23 | 7.94±4.40 | 116.55±38.84 | 23.43s | 12282.98±709.40 | 7.86±4.47 | 821.93±435.77 | 6.35s |
| FMIP | Apollo | SCIP | 16032.08±4310.89 | 42.25±37.87(72) | 95.01±10.58 | 34.34s | 17367.72±3070.38 | 52.64±24.55(95) | 657.19±203.56 | 9.39s | 17300.10±3040.28 | 51.90±24.25(97) | 5692.39±2189.50 | 22.07s |
| ConPaS | PS | SCIP | N/A | N/A(0) | 100.00±0.00 | 0.76s | N/A | N/A(0) | 1000.00±0.00 | 0.85s | N/A | N/A(0) | 10000.00±0.00 | 0.46s |
| ConPaS | ND | SCIP | N/A | N/A(0) | 100.00±0.00 | 1.63s | 59032.84±21663.10 | 419.18±190.41 | 887.62±235.35 | 10.57s | 12939.37±761.69 | 13.58±3.77 | 1758.56±379.13 | 100.23s |
| ConPaS | PMVB | SCIP | N/A | N/A(0) | 100.00±0.00 | 0.95s | N/A | N/A(0) | 1000.00±0.00 | 1.02s | N/A | N/A(0) | 10000.00±0.00 | 0.55s |
| ConPaS | Apollo | SCIP | N/A | N/A(0) | 100.00±0.00 | 1.48s | N/A | N/A(0) | 1000.00±0.00 | 0.91s | N/A | N/A(0) | 10000.00±0.00 | 0.82s |
| OPTFM | PS | SCIP | 27919.71±3403.63 | 145.44±31.71(99) | 99.98±0.13 | 1.19s | 27397.19±2842.37 | 140.88±26.05 | 998.86±6.59 | 1.63s | 27389.91±2845.49 | 140.82±26.06 | 9984.95±88.28 | 4.89s |
| OPTFM | ND | SCIP | N/A | N/A(0) | 100.00±0.00 | 1.24s | 13906.35±755.29 | 22.18±6.10 | 291.05±76.73 | 10.26s | 12895.21±805.00 | 13.39±8.29 | 1637.21±780.06 | 65.10s |
| OPTFM | PMVB | SCIP | 20794.34±9917.68 | 82.31±85.82 | 95.81±6.69 | 1.28s | 11828.15±586.26 | 3.87±3.02 | 271.55±62.02 | 10.30s | 11527.20±518.12 | 1.21±1.51 | 509.57±193.17 | 88.12s |
| OPTFM | Apollo | SCIP | 28542.03±3675.24 | 150.36±31.93(95) | 99.99±0.10 | 1.25s | 27411.46±2841.31 | 141.00±26.00 | 998.75±7.24 | 3.40s | 27391.14±2846.49 | 140.82±26.05 | 9984.39±91.46 | 4.96s |
| COPT | – | – | 19646.01 | 72.55±11.06 | 89.32±9.59 | 1.16s | 11846.21 | 3.99±2.19 | 243.99±35.33 | 10.03s | 11392.04 | 0.03±0.16 | 284.68±59.10 | 43.69s |
| Base-SL | PS | COPT | 24505.20±232.18 | 115.73±16.87(21) | 100.00±0.00 | 1.72s | 16587.44±1125.53 | 45.72±8.54(96) | 665.44±103.07 | 6.19s | 16578.41±1113.51 | 45.65±8.20(96) | 4959.21±1296.35 | 6.10s |
| Base-SL | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.54s | 23141.23±4595.62 | 103.62±39.76(7) | 989.51±62.85 | 10.55s | 16152.63±884.54 | 41.81±4.40 | 4453.65±437.13 | 100.65s |
| Base-SL | PMVB | COPT | 16031.80±7426.14 | 41.09±69.94(39) | 97.50±5.07 | 1.82s | 12044.94±662.11 | 5.78±3.66(99) | 264.60±126.17 | 6.44s | 11999.75±647.63 | 5.38±3.61(99) | 828.12±997.22 | 7.29s |
| Base-SL | Apollo | COPT | 27321.77±554.58 | 138.22±41.93(7) | 99.94±0.56 | 2.16s | 18872.17±4206.37 | 66.09±38.17(96) | 720.66±162.27 | 5.61s | 18735.92±4168.02 | 64.82±37.85(99) | 5992.27±2062.87 | 6.30s |
| PS-SL | PS | COPT | 23000.37±1907.22 | 103.05±15.42(24) | 100.00±0.00 | 1.71s | 16136.84±931.67 | 41.90±6.11(90) | 627.66±137.30 | 5.83s | 16118.63±922.16 | 41.74±5.86(90) | 4906.68±1784.22 | 6.07s |
| PS-SL | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.54s | 21978.99±6574.61 | 90.75±57.83(24) | 931.52±165.54 | 10.54s | 15957.14±1261.67 | 40.11±9.27 | 4438.42±473.80 | 100.62s |
| PS-SL | PMVB | COPT | 15760.77±4663.47 | 37.50±43.57(69) | 91.33±8.53 | 1.69s | 11829.58±582.92 | 3.87±2.85 | 207.86±64.40 | 8.94s | 11635.93±534.21 | 2.18±2.50 | 418.04±250.75 | 17.89s |
| PS-SL | Apollo | COPT | 23468.61±2977.20 | 104.74±20.06(5) | 99.92±0.67 | 2.04s | 16343.22±1085.31 | 43.53±7.32(88) | 633.40±148.08 | 5.99s | 16124.25±924.72 | 41.78±5.86(90) | 4903.26±1787.83 | 6.33s |
| CoCo | PS | COPT | 47579.58±2922.19 | 317.94±22.26 | 100.00±0.00 | 1.90s | 29516.52±2011.79 | 159.18±14.07 | 1000.00±0.00 | 5.16s | 29516.52±2011.79 | 159.18±14.07 | 10000.00±0.00 | 4.89s |
| CoCo | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.66s | 32355.97±4351.11 | 183.84±36.78(56) | 1000.00±0.00 | 10.65s | 22388.16±2696.47 | 96.92±25.05 | 9266.69±977.05 | 100.76s |
| CoCo | PMVB | COPT | 56357.29±3577.84 | 392.25±23.69(51) | 100.00±0.00 | 1.90s | 40575.19±1721.85 | 257.05±11.21(92) | 1000.00±0.00 | 4.89s | 40575.19±1721.85 | 257.05±11.21(92) | 10000.00±0.00 | 3.89s |
| CoCo | Apollo | COPT | 49076.79±4764.05 | 328.65±37.02(71) | 100.00±0.00 | 2.44s | 30512.06±6901.81 | 167.77±58.65 | 1000.00±0.00 | 6.09s | 28073.61±2000.23 | 146.50±14.09 | 10000.00±0.00 | 5.48s |
| DiffILO | PS | COPT | 22620.23±3987.08 | 98.10±32.39(95) | 99.15±3.27 | 1.50s | 19607.71±2040.71 | 71.79±15.15(97) | 814.87±100.96 | 3.43s | 19607.71±2040.71 | 71.79±15.15(97) | 7577.68±1310.93 | 3.45s |
| DiffILO | ND | COPT | 18213.93±2447.09 | 59.57±18.16(94) | 100.00±0.00 | 1.62s | 21032.61±6699.43 | 84.54±57.93(95) | 901.08±136.90 | 10.70s | 16610.28±832.76 | 45.88±5.14 | 4958.13±505.80 | 100.65s |
| DiffILO | PMVB | COPT | 18161.49±6568.84 | 59.09±55.39 | 89.77±6.91 | 1.42s | 16516.18±2126.67 | 44.79±15.60 | 502.85±142.40 | 3.32s | 16516.18±2126.67 | 44.79±15.60 | 4531.97±1546.70 | 3.18s |
| DiffILO | Apollo | COPT | 25627.57±8398.91 | 124.54±72.53(95) | 98.88±3.44 | 1.69s | 20126.10±2193.27 | 76.40±17.39(97) | 835.03±108.83 | 4.34s | 20126.10±2193.27 | 76.40±17.39(97) | 7897.47±1365.72 | 4.19s |
| DDIM | PS | COPT | 13225.17±697.45 | 16.65±4.96(36) | 100.00±0.00 | 6.63s | 13451.75±1143.05 | 19.08±9.83(38) | 977.07±92.93 | 8.53s | 13488.58±1150.92 | 19.72±10.46(39) | 9603.79±1452.02 | 12.18s |
| DDIM | ND | COPT | 13205.91±702.68 | 16.49±5.03(37) | 100.00±0.00 | 6.11s | 16427.26±8104.21 | 44.19±69.68(42) | 978.70±94.55 | 15.32s | 16145.44±782.69 | 41.78±3.85 | 4441.81±372.12 | 107.34s |
| DDIM | PMVB | COPT | 15262.68±8794.88 | 34.07±79.13(69) | 95.96±6.39 | 5.56s | 12612.69±635.32 | 10.81±4.91 | 206.98±49.34 | 21.00s | 12608.14±641.41 | 10.77±4.96 | 1182.52±483.69 | 12.74s |
| DDIM | Apollo | COPT | 13171.99±773.11 | 17.23±5.11(31) | 100.00±0.00 | 8.58s | 13689.73±1554.32 | 22.18±13.75(34) | 972.05±101.02 | 28.68s | 14328.00±2254.31 | 27.60±19.83(34) | 9288.09±1852.51 | 24.61s |
| FMIP | PS | COPT | 13666.75±4283.43 | 20.39±35.40(55) | 100.00±0.00 | 4.56s | 17152.72±2921.31 | 50.95±24.27(93) | 782.76±162.79 | 19.14s | 16944.83±2931.31 | 49.10±24.26(93) | 6222.76±2406.13 | 23.77s |
| FMIP | ND | COPT | 12768.75±727.04 | 12.28±4.70(50) | 100.00±0.00 | 12.98s | 19140.17±8811.68 | 67.99±76.55(73) | 842.52±216.23 | 21.51s | 16055.65±842.64 | 40.97±4.14 | 4329.29±428.90 | 111.70s |
| FMIP | PMVB | COPT | 13213.62±4313.54 | 15.70±36.90(90) | 83.75±10.48 | 8.89s | 12279.92±711.18 | 7.83±4.50 | 161.84±37.55 | 14.68s | 12275.17±713.05 | 7.79±4.54 | 904.75±436.19 | 13.65s |
| FMIP | Apollo | COPT | 13212.92±3604.06 | 16.42±30.58(52) | 100.00±0.00 | 26.58s | 17423.68±3875.95 | 53.75±32.54(87) | 817.62±179.34 | 26.45s | 17281.64±3047.49 | 51.74±24.34(97) | 5879.99±2110.61 | 28.41s |
| ConPaS | PS | COPT | N/A | N/A(0) | 100.00±0.00 | 1.46s | N/A | N/A(0) | 1000.00±0.00 | 2.22s | N/A | N/A(0) | 10000.00±0.00 | 2.74s |
| ConPaS | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.27s | 50790.83±0.00 | 326.02±0.00(1) | 1000.00±0.00 | 10.31s | 18170.45±975.76 | 59.60±6.79 | 6229.88±676.66 | 100.56s |
| ConPaS | PMVB | COPT | N/A | N/A(0) | 100.00±0.00 | 1.44s | N/A | N/A(0) | 1000.00±0.00 | 3.03s | N/A | N/A(0) | 10000.00±0.00 | 3.49s |
| ConPaS | Apollo | COPT | N/A | N/A(0) | 100.00±0.00 | 1.88s | N/A | N/A(0) | 1000.00±0.00 | 4.60s | N/A | N/A(0) | 10000.00±0.00 | 5.05s |
| OPTFM | PS | COPT | 41245.92±3727.35 | 261.78±31.85(95) | 100.00±0.00 | 1.81s | 27393.07±2845.71 | 140.84±26.07 | 999.21±4.71 | 5.27s | 27389.93±2845.50 | 140.82±26.06 | 9984.95±88.36 | 5.55s |
| OPTFM | ND | COPT | N/A | N/A(0) | 100.00±0.00 | 1.57s | 25706.86±6767.31 | 127.39±59.09(45) | 972.57±79.14 | 10.56s | 13157.15±1152.23 | 15.78±12.52 | 3821.03±1111.29 | 91.69s |
| OPTFM | PMVB | COPT | 20187.62±1482.42 | 77.39±12.80 | 99.35±1.74 | 1.70s | 11892.48±631.43 | 4.39±2.31 | 316.06±28.13 | 10.68s | 11392.18±480.80 | 0.03±0.11 | 350.20±52.92 | 44.76s |
| OPTFM | Apollo | COPT | 41272.79±3961.70 | 261.90±33.40(89) | 100.00±0.00 | 2.12s | 27432.33±2853.87 | 141.18±26.09 | 999.34±4.50 | 5.53s | 27391.18±2846.51 | 140.82±26.05 | 9984.60±90.37 | 6.21s |

### I.4 Supplementary Results on Learners with Different Refiners

Table 20 reports the complete results for Learners paired with different incumbent Refiners. Figures 18 and 17 provide two complementary views of the corresponding gap-time trajectories, grouping curves by Refiner and by Learner, respectively.

Table 20: Complete results for learners with different incumbent (solver-supported) *refiners*. We report the primal gaps (%) at 10/100/1000 seconds (G@10,$10^2$,$10^3$) and the primal integral over the 1000s refinement horizon. The backend solver for the `LNS` refiner family is fixed as `Gurobi`. Refiner-specific parameters used to obtain this table of results are offered in Table 13.

| Learner | Refiner | **SC** (100 inst.) G@10 | G@$10^2$ | G@$10^3$ | PI↓ | **MIS** (100 inst.) G@10 | G@$10^2$ | G@$10^3$ | PI↓ | **CA** (100 inst.) G@10 | G@$10^2$ | G@$10^3$ | PI↓ | **CFL** (100 inst.) G@10 | G@$10^2$ | G@$10^3$ | PI↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gurobi 1000s | | 6.259 | 0.916 | 0.140 | 973.68 | 0.018 | 0.000 | 0.000 | 15.95 | 0.676 | 0.086 | 0.000 | 42.99 | 0.097 | 0.000 | 0.000 | 20.24 |
| Base-SL | WS(Gurobi) | 65.182 | 0.955 | 0.162 | 1255.95 | 0.025 | 0.000 | 0.000 | 2.18 | 0.595 | 0.071 | 0.000 | 33.77 | 0.125 | 0.000 | 0.000 | 21.06 |
| PS-SL | WS(Gurobi) | 30.583 | 0.899 | 0.162 | 1191.25 | 0.016 | 0.002 | 0.000 | 4.06 | 0.705 | 0.082 | 0.000 | 40.72 | 0.113 | 0.000 | 0.000 | 21.52 |
| ConPaS | WS(Gurobi) | 105.007 | 0.940 | 0.163 | 1209.33 | 0.012 | 0.000 | 0.000 | 2.47 | 0.662 | 0.088 | 0.000 | 41.55 | 0.109 | 0.000 | 0.000 | 21.13 |
| CoCo-MILP | WS(Gurobi) | 51.943 | 0.943 | 0.153 | 1228.89 | 0.016 | 0.000 | 0.000 | 3.05 | 0.726 | 0.063 | 0.000 | 33.96 | 0.104 | 0.000 | 0.000 | 20.71 |
| OPTFM | WS(Gurobi) | 44.906 | 1.016 | 0.142 | 1219.57 | 0.016 | 0.002 | 0.002 | 3.55 | 0.665 | 0.082 | 0.000 | 36.73 | 0.108 | 0.000 | 0.000 | 21.64 |
| DiffILO | WS(Gurobi) | 83.326 | 0.999 | 0.195 | 1307.18 | 0.027 | 0.000 | 0.000 | 1.93 | 0.647 | 0.088 | 0.000 | 36.69 | 0.107 | 0.000 | 0.000 | 21.30 |
| DDIM | WS(Gurobi) | 49.802 | 0.933 | 0.142 | 1207.40 | 0.015 | 0.002 | 0.000 | 1.20 | 0.675 | 0.089 | 0.000 | 40.75 | 0.111 | 0.000 | 0.000 | 21.53 |
| FMIP | WS(Gurobi) | 19.814 | 0.955 | 0.121 | 1157.10 | 0.015 | 0.002 | 0.000 | 1.91 | 0.721 | 0.080 | 0.000 | 38.20 | 0.107 | 0.000 | 0.000 | 21.39 |
| Base-SL | LB(Gurobi) | 45.953 | 0.879 | 0.182 | 1245.11 | 0.002 | 0.002 | 0.000 | 0.65 | 0.742 | 0.122 | 0.023 | 61.23 | 0.104 | 0.000 | 0.000 | 21.23 |
| PS-SL | LB(Gurobi) | 30.898 | 0.993 | 0.210 | 1242.88 | 0.006 | 0.002 | 0.002 | 2.47 | 0.784 | 0.112 | 0.015 | 59.28 | 0.132 | 0.000 | 0.000 | 22.39 |
| ConPaS | LB(Gurobi) | 57.030 | 1.044 | 0.230 | 1276.43 | 0.004 | 0.002 | 0.000 | 0.89 | 0.803 | 0.114 | 0.014 | 56.88 | 0.101 | 0.000 | 0.000 | 21.03 |
| CoCo-MILP | LB(Gurobi) | 37.482 | 0.901 | 0.187 | 1222.50 | 0.003 | 0.000 | 0.000 | 0.83 | 0.871 | 0.108 | 0.006 | 51.42 | 0.112 | 0.000 | 0.000 | 22.45 |
| OPTFM | LB(Gurobi) | 15.601 | 0.900 | 0.278 | 1088.15 | 0.003 | 0.000 | 0.000 | 0.59 | 0.801 | 0.087 | 0.007 | 43.66 | 0.109 | 0.000 | 0.000 | 22.39 |
| DiffILO | LB(Gurobi) | 33.023 | 0.980 | 0.128 | 1203.57 | 0.004 | 0.002 | 0.000 | 0.83 | 0.755 | 0.136 | 0.013 | 52.85 | 0.078 | 0.000 | 0.000 | 21.36 |
| DDIM | LB(Gurobi) | 77.633 | 1.020 | 0.181 | 1302.59 | 0.004 | 0.003 | 0.000 | 1.19 | 0.794 | 0.118 | 0.018 | 62.83 | 0.132 | 0.000 | 0.000 | 21.72 |
| FMIP | LB(Gurobi) | 15.031 | 0.951 | 0.211 | 1198.56 | 0.003 | 0.002 | 0.000 | 1.40 | 0.811 | 0.091 | 0.015 | 49.67 | 0.080 | 0.000 | 0.000 | 21.53 |
| SCIP 1000s | | 8.536 | 1.525 | 0.207 | 997.49 | 3.415 | 0.599 | 0.115 | 389.66 | 1.731 | 0.026 | 0.000 | 78.82 | 3.889 | 1.226 | 0.061 | 653.71 |
| Base-SL | WS(SCIP) | 14.179 | 1.573 | 0.237 | 1122.58 | 0.607 | 0.241 | 0.081 | 167.53 | 1.301 | 0.048 | 0.004 | 49.87 | 3.982 | 1.480 | 0.024 | 737.09 |
| PS-SL | WS(SCIP) | 12.228 | 1.486 | 0.215 | 1081.37 | 1.458 | 0.383 | 0.084 | 241.53 | 1.740 | 0.053 | 0.000 | 55.16 | 4.003 | 1.420 | 0.038 | 749.48 |
| ConPaS | WS(SCIP) | 13.217 | 1.694 | 0.205 | 1038.96 | 0.731 | 0.249 | 0.078 | 163.32 | 1.784 | 0.075 | 0.007 | 75.60 | 4.012 | 1.478 | 0.037 | 775.42 |
| CoCo-MILP | WS(SCIP) | 14.128 | 1.598 | 0.236 | 1059.77 | 0.918 | 0.276 | 0.090 | 200.05 | 1.764 | 0.051 | 0.000 | 53.05 | 4.119 | 1.406 | 0.103 | 823.77 |
| OPTFM | WS(SCIP) | 8.376 | 1.517 | 0.144 | 949.03 | 0.593 | 0.232 | 0.077 | 158.36 | 1.442 | 0.045 | 0.000 | 42.32 | 3.959 | 1.495 | 0.054 | 808.08 |
| DiffILO | WS(SCIP) | 7.963 | 1.435 | 0.206 | 972.50 | 0.357 | 0.250 | 0.100 | 169.30 | 1.324 | 0.030 | 0.000 | 38.10 | 3.940 | 1.415 | 0.059 | 793.28 |
| DDIM | WS(SCIP) | 8.262 | 1.462 | 0.203 | 993.76 | 0.163 | 0.117 | 0.050 | 86.64 | 1.741 | 0.038 | 0.000 | 46.64 | 4.256 | 1.618 | 0.045 | 853.44 |
| FMIP | WS(SCIP) | 8.177 | 1.419 | 0.185 | 952.33 | 0.073 | 0.057 | 0.035 | 46.87 | 1.707 | 0.031 | 0.000 | 46.45 | 4.515 | 1.457 | 0.093 | 844.39 |
| Base-SL | LB(SCIP) | 9.296 | 1.672 | 0.235 | 1060.89 | 0.607 | 0.175 | 0.009 | 72.06 | 1.444 | 0.058 | 0.000 | 44.87 | 3.677 | 1.349 | 0.049 | 726.55 |
| PS-SL | LB(SCIP) | 9.544 | 1.815 | 0.153 | 1061.90 | 1.626 | 0.283 | 0.003 | 122.80 | 2.095 | 0.052 | 0.000 | 55.50 | 3.686 | 1.230 | 0.038 | 700.58 |
| ConPaS | LB(SCIP) | 9.280 | 1.883 | 0.113 | 1071.20 | 0.821 | 0.210 | 0.010 | 83.95 | 2.421 | 0.050 | 0.000 | 64.93 | 3.726 | 1.171 | 0.037 | 700.74 |
| CoCo-MILP | LB(SCIP) | 9.644 | 1.779 | 0.143 | 1045.14 | 1.028 | 0.199 | 0.004 | 94.30 | 2.116 | 0.063 | 0.000 | 58.46 | 3.618 | 1.293 | 0.026 | 709.96 |
| OPTFM | LB(SCIP) | 9.572 | 1.798 | 0.136 | 1021.51 | 0.635 | 0.196 | 0.007 | 81.85 | 1.597 | 0.055 | 0.000 | 50.04 | 3.668 | 1.188 | 0.057 | 702.22 |
| DiffILO | LB(SCIP) | 9.413 | 1.847 | 0.186 | 1088.26 | 0.357 | 0.216 | 0.003 | 71.59 | 1.442 | 0.085 | 0.000 | 53.51 | 3.572 | 1.292 | 0.019 | 723.21 |
| DDIM | LB(SCIP) | 9.647 | 1.821 | 0.214 | 1150.73 | 0.163 | 0.076 | 0.006 | 30.40 | 2.039 | 0.053 | 0.000 | 58.20 | 3.548 | 1.308 | 0.038 | 739.64 |
| FMIP | LB(SCIP) | 9.032 | 2.015 | 0.175 | 1145.78 | 0.073 | 0.066 | 0.010 | 32.59 | 2.293 | 0.041 | 0.000 | 57.60 | 3.461 | 1.290 | 0.087 | 746.42 |
| COPT 1000s | | 3318.282 | 3318.282 | 0.180 | 29676.31 | 10.816 | 0.353 | 0.016 | 588.05 | 1.137 | 0.006 | 0.000 | 76.62 | 3.935 | 0.027 | 0.000 | 288.05 |
| Base-SL | WS(COPT) | 807.625 | 807.625 | 0.264 | 39190.78 | 0.607 | 0.214 | 0.016 | 88.84 | 1.267 | 0.014 | 0.000 | 34.69 | 4.426 | 0.055 | 0.000 | 383.87 |
| PS-SL | WS(COPT) | 777.849 | 777.849 | 0.274 | 39520.83 | 1.692 | 0.291 | 0.016 | 157.19 | 1.581 | 0.016 | 0.000 | 41.50 | 4.475 | 0.066 | 0.000 | 382.74 |
| ConPaS | WS(COPT) | 3318.282 | 3318.282 | 0.264 | 39509.09 | 0.822 | 0.231 | 0.016 | 99.05 | 1.718 | 0.014 | 0.000 | 45.81 | 4.390 | 0.065 | 0.000 | 382.74 |
| CoCo-MILP | WS(COPT) | 789.203 | 789.203 | 0.264 | 40050.43 | 1.028 | 0.244 | 0.016 | 114.03 | 1.661 | 0.016 | 0.000 | 43.59 | 4.395 | 0.052 | 0.000 | 387.88 |
| OPTFM | WS(COPT) | 838.723 | 838.723 | 0.306 | 42402.76 | 0.635 | 0.169 | 0.012 | 81.89 | 1.442 | 0.014 | 0.000 | 37.55 | 4.368 | 0.056 | 0.000 | 382.92 |
| DiffILO | WS(COPT) | 695.869 | 695.869 | 0.296 | 41604.75 | 0.357 | 0.183 | 0.012 | 76.07 | 1.332 | 0.014 | 0.000 | 35.22 | 4.437 | 0.060 | 0.000 | 380.54 |
| DDIM | WS(COPT) | 894.422 | 894.422 | 0.306 | 42308.77 | 0.163 | 0.100 | 0.009 | 40.11 | 1.636 | 0.014 | 0.000 | 42.09 | 4.444 | 0.060 | 0.000 | 389.65 |
| FMIP | WS(COPT) | 882.170 | 882.170 | 0.296 | 41700.14 | 0.073 | 0.053 | 0.006 | 25.54 | 1.687 | 0.014 | 0.000 | 44.48 | 4.386 | 0.055 | 0.000 | 378.46 |
| Base-SL | LB(COPT) | 482.442 | 482.442 | 0.329 | 36484.68 | 0.135 | 0.022 | 0.002 | 13.78 | 1.336 | 0.026 | 0.000 | 38.05 | 4.794 | 0.082 | 0.000 | 571.23 |
| PS-SL | LB(COPT) | 486.198 | 486.198 | 0.240 | 36232.36 | 0.286 | 0.026 | 0.004 | 22.91 | 1.820 | 0.032 | 0.000 | 48.36 | 4.747 | 0.074 | 0.000 | 560.90 |
| ConPaS | LB(COPT) | 1957.645 | 1957.645 | 0.276 | 40148.44 | 0.248 | 0.035 | 0.003 | 20.83 | 2.104 | 0.023 | 0.000 | 52.29 | 4.805 | 0.124 | 0.000 | 561.69 |
| CoCo-MILP | LB(COPT) | 482.380 | 482.380 | 0.218 | 37396.70 | 0.281 | 0.032 | 0.006 | 25.16 | 1.947 | 0.033 | 0.000 | 51.47 | 4.790 | 0.090 | 0.000 | 545.43 |
| OPTFM | LB(COPT) | 546.134 | 546.134 | 0.273 | 38470.33 | 0.172 | 0.034 | 0.004 | 17.40 | 1.502 | 0.019 | 0.000 | 39.38 | 4.803 | 0.057 | 0.000 | 559.32 |
| DiffILO | LB(COPT) | 466.489 | 466.489 | 0.201 | 38222.28 | 0.176 | 0.022 | 0.003 | 17.33 | 1.301 | 0.031 | 0.000 | 36.17 | 4.780 | 0.115 | 0.000 | 559.97 |
| DDIM | LB(COPT) | 515.685 | 515.685 | 0.297 | 37863.80 | 0.082 | 0.026 | 0.003 | 12.11 | 1.843 | 0.034 | 0.000 | 50.22 | 4.778 | 0.059 | 0.000 | 568.28 |
| FMIP | LB(COPT) | 527.565 | 527.565 | 0.315 | 38455.67 | 0.057 | 0.019 | 0.006 | 10.97 | 1.954 | 0.040 | 0.000 | 52.56 | 4.761 | 0.065 | 0.000 | 562.94 |
| Base-SL | LNS-Rand | 5.142 | 2.126 | 1.157 | 1816.74 | 0.075 | 0.028 | 0.015 | 20.89 | 1.268 | 1.161 | 1.015 | 1075.37 | 15.155 | 8.502 | 7.284 | 8041.56 |
| PS-SL | LNS-Rand | 5.091 | 1.912 | 1.143 | 1732.34 | 0.076 | 0.023 | 0.013 | 18.82 | 1.881 | 1.648 | 1.434 | 1524.81 | 15.514 | 7.706 | 6.507 | 7395.04 |
| ConPaS | LNS-Rand | 24.768 | 2.308 | 1.057 | 1984.77 | 0.078 | 0.031 | 0.021 | 26.52 | 2.110 | 1.796 | 1.547 | 1640.15 | 15.803 | 10.332 | 8.898 | 9787.25 |
| CoCo | LNS-Rand | 4.281 | 2.091 | 1.346 | 1820.71 | 0.078 | 0.029 | 0.015 | 21.13 | 1.986 | 1.757 | 1.518 | 1634.55 | 15.600 | 8.488 | 6.973 | 7930.59 |
| OPTFM | LNS-Rand | 4.696 | 1.991 | 1.121 | 1702.71 | 0.069 | 0.020 | 0.016 | 18.98 | 1.459 | 1.319 | 1.162 | 1227.84 | 15.295 | 9.171 | 7.957 | 8664.89 |
| DiffILO | LNS-Rand | 4.730 | 2.147 | 1.075 | 1729.69 | 0.073 | 0.029 | 0.016 | 22.38 | 1.286 | 1.188 | 1.057 | 1104.78 | 15.512 | 8.375 | 7.212 | 8016.68 |
| DDIM | LNS-Rand | 4.713 | 2.131 | 1.037 | 1676.11 | 0.044 | 0.023 | 0.016 | 19.55 | 1.906 | 1.689 | 1.467 | 1574.61 | 14.421 | 7.648 | 6.624 | 7310.73 |
| FMIP | LNS-Rand | 4.390 | 1.922 | 1.071 | 1601.80 | 0.040 | 0.026 | 0.023 | 24.48 | 2.019 | 1.727 | 1.441 | 1568.28 | 13.921 | 8.156 | 7.000 | 7780.43 |
| Base-SL | LNS-Part | 5.064 | 2.202 | 1.023 | 1736.37 | 0.076 | 0.037 | 0.016 | 24.71 | 1.255 | 1.125 | 1.040 | 1079.57 | 14.016 | 7.777 | 6.534 | 7391.62 |
| PS-SL | LNS-Part | 12.174 | 1.980 | 1.159 | 1754.00 | 0.102 | 0.035 | 0.016 | 27.82 | 1.869 | 1.627 | 1.440 | 1524.46 | 14.220 | 7.963 | 6.911 | 7674.64 |
| ConPaS | LNS-Part | 28.150 | 1.965 | 1.237 | 1763.59 | 0.079 | 0.037 | 0.021 | 27.93 | 2.033 | 1.784 | 1.577 | 1657.51 | 14.951 | 10.191 | 8.535 | 9457.84 |
| CoCo-MILP | LNS-Part | 4.662 | 2.110 | 1.026 | 1692.93 | 0.078 | 0.035 | 0.018 | 24.01 | 2.072 | 1.715 | 1.474 | 1584.41 | 15.822 | 9.106 | 7.682 | 8575.22 |
| OPTFM | LNS-Part | 5.275 | 2.397 | 1.092 | 1828.23 | 0.069 | 0.034 | 0.018 | 25.56 | 1.489 | 1.327 | 1.201 | 1280.58 | 15.292 | 9.327 | 7.785 | 8687.31 |
| DiffILO | LNS-Part | 8.444 | 1.849 | 0.955 | 1560.65 | 0.082 | 0.037 | 0.021 | 27.48 | 1.289 | 1.163 | 1.021 | 1080.98 | 14.924 | 9.249 | 7.906 | 8715.93 |
| DDIM | LNS-Part | 4.559 | 1.867 | 1.026 | 1550.66 | 0.051 | 0.029 | 0.021 | 23.72 | 1.877 | 1.616 | 1.415 | 1509.73 | 15.043 | 7.787 | 6.462 | 7322.37 |
| FMIP | LNS-Part | 11.998 | 1.930 | 1.036 | 1634.94 | 0.034 | 0.025 | 0.022 | 23.99 | 2.010 | 1.721 | 1.461 | 1590.28 | 15.215 | 8.201 | 7.010 | 7816.44 |
| Base-SL | LNS-BFS | 6.632 | 2.919 | 1.724 | 2503.86 | 0.086 | 0.026 | 0.016 | 21.76 | 1.391 | 1.321 | 1.261 | 1285.00 | 1.833 | 0.041 | 0.002 | 330.39 |
| PS-SL | LNS-BFS | 6.334 | 2.657 | 1.652 | 2304.39 | 0.117 | 0.031 | 0.015 | 22.89 | 2.226 | 2.017 | 1.890 | 1951.37 | 0.893 | 0.035 | 0.000 | 306.48 |
| ConPaS | LNS-BFS | 11.285 | 2.881 | 1.843 | 2794.02 | 0.110 | 0.031 | 0.022 | 28.16 | 2.511 | 2.318 | 2.157 | 2239.57 | 1.257 | 0.058 | 0.001 | 336.72 |
| CoCo-MILP | LNS-BFS | 7.072 | 2.881 | 1.713 | 2432.85 | 0.102 | 0.034 | 0.019 | 27.25 | 2.459 | 2.249 | 2.123 | 2184.05 | 1.746 | 0.084 | 0.001 | 332.97 |
| OPTFM | LNS-BFS | 8.732 | 2.948 | 2.042 | 2776.35 | 0.103 | 0.034 | 0.015 | 24.08 | 1.663 | 1.570 | 1.476 | 1514.77 | 3.939 | 0.034 | 0.001 | 369.54 |
| DiffILO | LNS-BFS | 8.317 | 2.708 | 1.645 | 2496.26 | 0.110 | 0.044 | 0.018 | 26.73 | 1.464 | 1.379 | 1.286 | 1321.32 | 2.597 | 0.068 | 0.002 | 359.19 |
| DDIM | LNS-BFS | 8.275 | 2.789 | 1.817 | 2544.66 | 0.056 | 0.026 | 0.019 | 21.96 | 2.252 | 2.043 | 1.899 | 1966.35 | 1.985 | 0.080 | 0.000 | 355.59 |
| FMIP | LNS-BFS | 8.593 | 2.656 | 1.393 | 2271.37 | 0.041 | 0.025 | 0.022 | 23.53 | 2.443 | 2.239 | 2.046 | 2135.49 | 2.018 | 0.084 | 0.001 | 344.73 |

**Refiner choice substantially reshapes downstream performance.** Across most settings, changing the Refiner produces larger differences than changing the Learner within a fixed Refiner, particularly after the initial search phase. With the Gurobi backend, WS and LB substantially outperform the LNS variants on SC and CA: at 1000 seconds, the best LNS gaps remain 0.955% and 1.015%, respectively, whereas WS or LB reaches 0.128% on SC and zero or near-zero gaps on CA. More elaborate iterative neighborhood search is therefore not inherently more effective than simpler warm-start or local-branching refinement. The preferred neighborhood is also task dependent. On CFL, LNS-BFS reduces PI to 306.48-369.54, markedly improving over LNS-Rand and LNS-Part, whose PIs exceed $7.3 \times 10^3$, although it remains substantially behind WS and LB. This suggests that

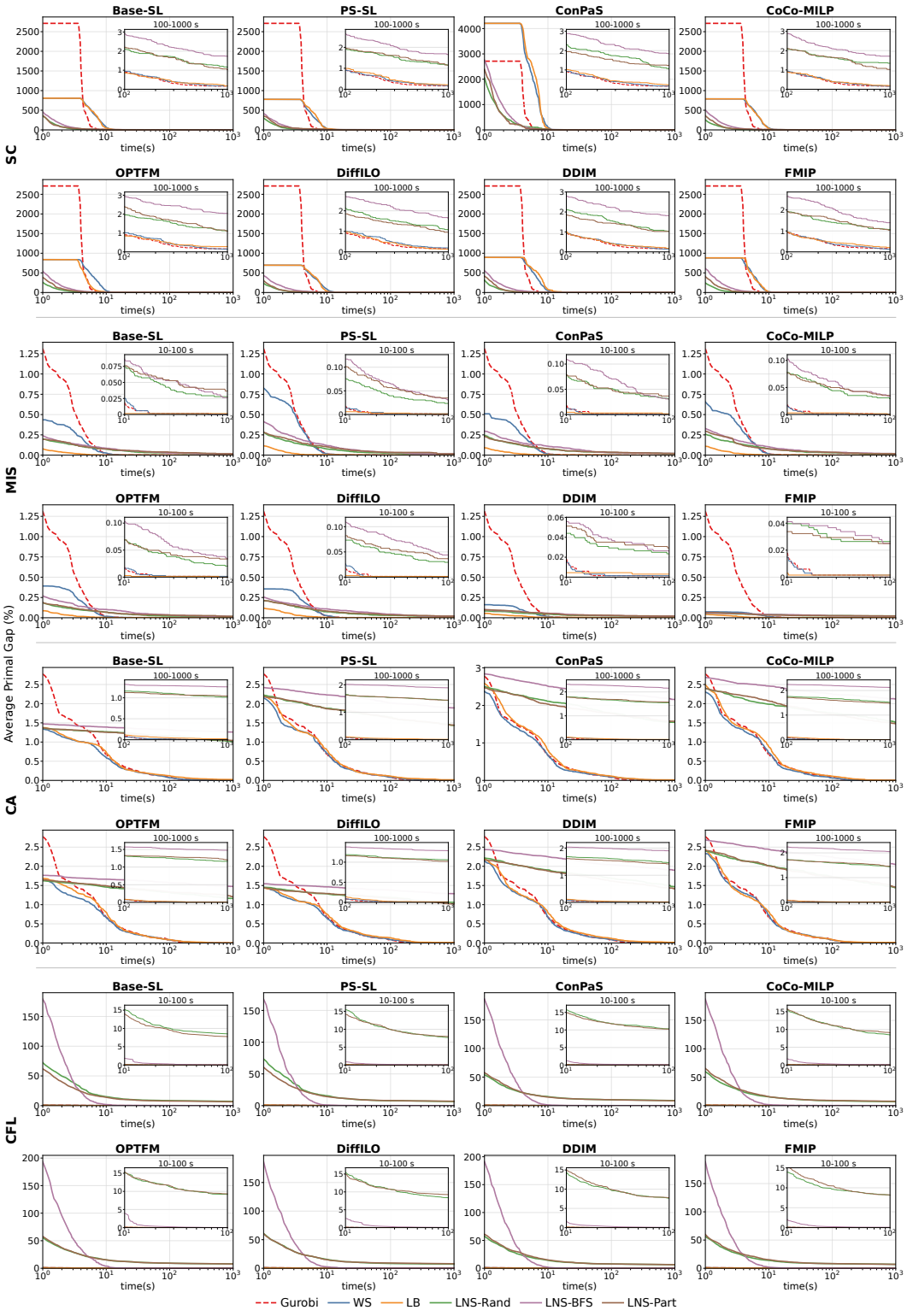



Figure 17: Gap-time curves for fixed Learner with different Refiners. $S_R$: `Gurobi`.

graph-local neighborhoods better match CFL structure, but neighborhood compatibility alone does not compensate for insufficient search strength.

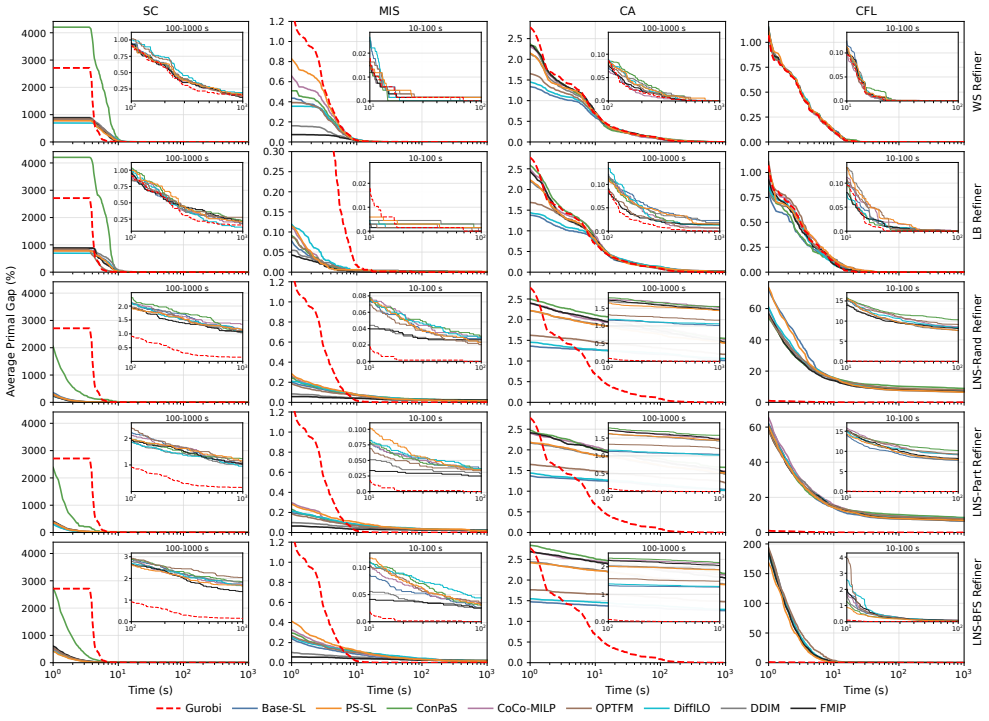



Figure 18: Complete gap-time curves for different Learners with different Refiners. $S_R$: `Gurobi`.

**Learner utility is conditional on the Refiner and solver backend.** The best Learner changes across Refiners, with frequent crossings and rank reversals in the gap-time curves. On MIS, DDIM performs best under Gurobi-backed WS with a PI of 1.20, whereas OPTFM is best under LB with 0.59 and PS-SL is best under LNS-Rand with 18.82. Similarly, on CFL, CoCo-MILP, ConPaS, and PS-SL are preferred under WS, LB, and LNS-BFS, respectively. The solver backend further alters these interactions. Under SCIP, WS and LB reduce the MIS baseline PI from 389.66 to 46.87-241.53 and 30.40-122.80, respectively, but every corresponding pairing on CFL is worse than the SCIP baseline. Thus, a Learner cannot be assigned a backend-independent downstream ranking; its utility depends on how the Refiner exposes, fixes, or repairs the information encoded in its predictions.

**Learner differences are primarily reflected in anytime behavior.** For Gurobi-backed WS on CFL, all Learners reach zero gap by 100 seconds and yield a narrow PI range of 20.71-21.64. Similar compression appears on CA and MIS, where most Learner-specific trajectories converge to identical or nearly identical terminal gaps despite appreciable early-stage differences. Conversely, the larger separation retained by weak LNS configurations should not be interpreted as better exploitation of learned guidance, since it often coincides with substantially worse residual gaps and PIs. These results reinforce the main-text finding that neither terminal solution quality nor preservation of the initial Learner ordering alone establishes effective use of learned information. A desirable Refiner should instead be co-designed with the semantics, calibration, and confidence of Learner scores, while adapting its intervention to the task structure and available budget. Accordingly, Learner-Refiner pairs should be evaluated jointly using full anytime trajectories and primal integrals, rather than selected independently or compared only through terminal gaps.

**Discussion on the Negative Transfer from Incumbent Injection in WS.** Table 20 also reveals that a learned incumbent is not uniformly beneficial to subsequent exact search. Compared with `Gurobi` from scratch, all `WS` configurations increase PI on SC from 973.68 to 1157.10-1307.18 and, more mildly, on CFL from 20.24 to 20.71-21.64. One plausible explanation is that injecting a suboptimal or structurally mismatched incumbent changes incumbent-dependent solver decisions, such as cutoff management, root-heuristic scheduling, and the allocation of early search effort, potentially delaying native construction heuristics or directing exploration toward less productive regions. Since these internal mechanisms cannot be isolated from aggregate trajectories, we regard this explanation as a reasonable speculation rather than a causal conclusion.

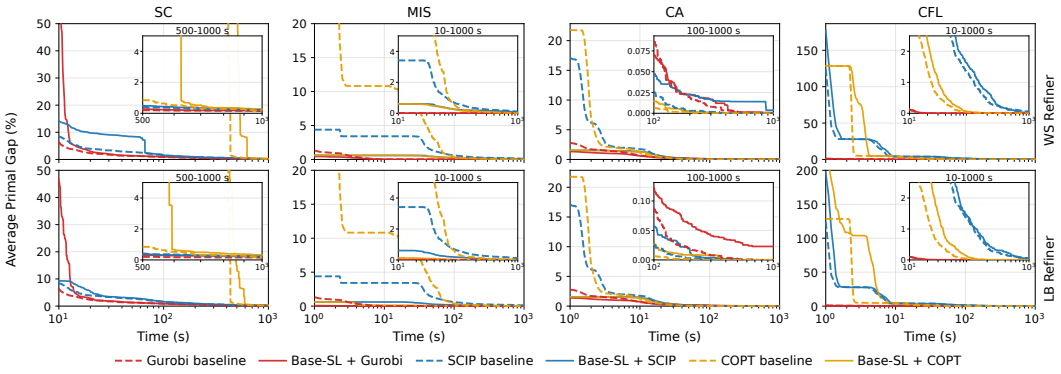



Figure 19: Comparison of different solver backends with the same Refiner (`WS` & `LB`) and Learner (Base-SL) on different tasks.

**Clarification on the Observed Limitations of LNS-based Refiners.** To avoid overgeneralizing our findings, we emphasize that the reported performance deficits apply specifically to generic LNS with learning-free destroy policies. Learned LNS methods (Yuan et al., 2025b; Wu et al., 2021; Sonnerat et al., 2021; ravichandra addanki et al., 2020) were developed precisely to address the difficulty of constructing effective neighborhoods for tightly constrained MILPs. As trainable neighborhood policies fall outside the current scope of our benchmark, our results should not be interpreted as evidence against the broader LNS paradigm. Systematically incorporating and evaluating learned destroy policies within the unified Learner-Enforcer-Refiner framework remains an important direction for future work.

**Discussion on the Effect of Solver Backend Selection.** Table 20 and the primal-gap curves show that the native strength of the backend can outweigh gains from the surrounding refinement pipeline. On MIS, the PI of Gurobi from scratch is 15.95, compared with 389.66 for SCIP and 588.05 for COPT; on CFL, the corresponding values are 20.24, 653.71, and 288.05. The curves locate much of this difference in early search: Gurobi attains substantially lower gaps by 10 seconds, even where final gaps later converge. A learned start can markedly improve a weaker backend without closing this gap: on MIS, Base-SL with `WS` reduces SCIP's PI to 167.53 and COPT's to 88.84, both still above unassisted Gurobi. This pattern is task dependent: Gurobi and SCIP are comparable on SC, while on CA Base-SL with `WS` brings COPT close to Gurobi with the same Refiner (34.69 versus 33.77 PI). Thus, gains measured over a particular solver version need not persist as backend solvers improve. Evaluating learned refinement requires both a strong, current solver baseline and explicit reporting of backend-conditioned gains, rather than relying on marginal improvements over a fixed solver configuration.

## I.5 Additional Results and Discussion for Stage-wise Correlation

### I.5.1 Full Learner-Enforcer Scatterplot and Discussion

Different Enforcers vary substantially in their ability to preserve and leverage raw learner informativeness. As shown in Figure 20, higher raw prediction quality does not translate uniformly into better downstream performance; instead, this relationship is strongly mediated by how the Enforcer interprets and uses the predicted scores. Some Enforcers preserve the relative quality encoded by the Learner, whereas others may underutilize, reinterpret, or overwrite its signal during solver-backed completion, thereby masking meaningful differences among Learners. This finding motivates a key design principle: an effective Enforcer should not merely convert predictions into feasible solutions, but should also retain and exploit their informative structure. Future ML4MILP systems may therefore benefit from tighter Learner-Enforcer co-design, with fixing, prioritization, and search strategies tailored to the semantics and confidence of the predicted scores.

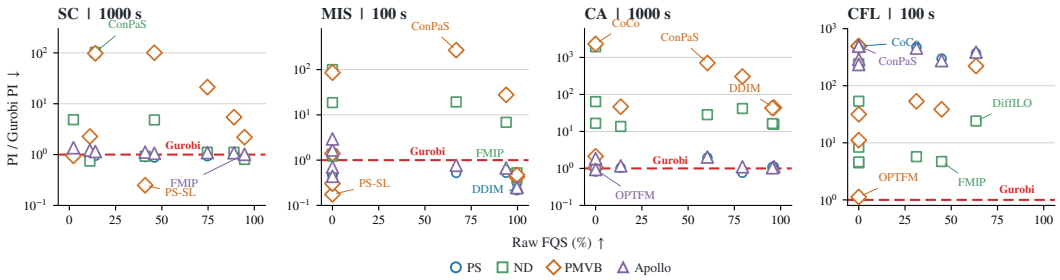



Figure 20: Raw learner quality (FQS↑) vs. Enforcer utility (relative PI↓). Each marker denotes the result of one of eight Learners coupled with the indicated Enforcer on a given task. The horizontal axis reports raw FQS by 1000-sampling from Table 2; the vertical axis reports the primal integral normalized by the matched Gurobi baseline on a logarithmic scale. The selected labels identify representative Learners.

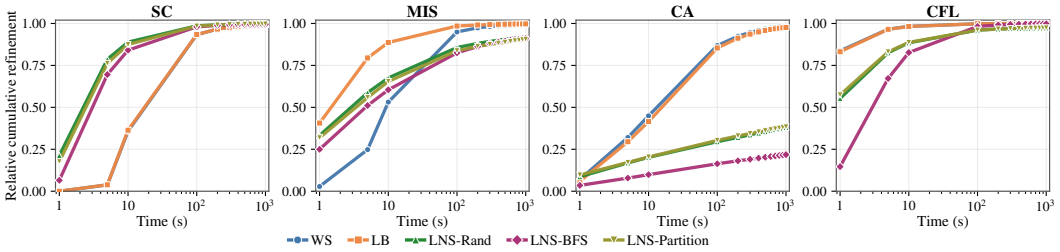



Figure 21: Relative cumulative refinement over time across tasks and Refiners. Each curve reports the normalized anytime reduction in the initial BKS-relative gap, averaged over Learner-conditioned initial incumbents. Higher values indicate earlier and larger improvements; 0 denotes no improvement and 1 denotes immediate closure of the initial gap.

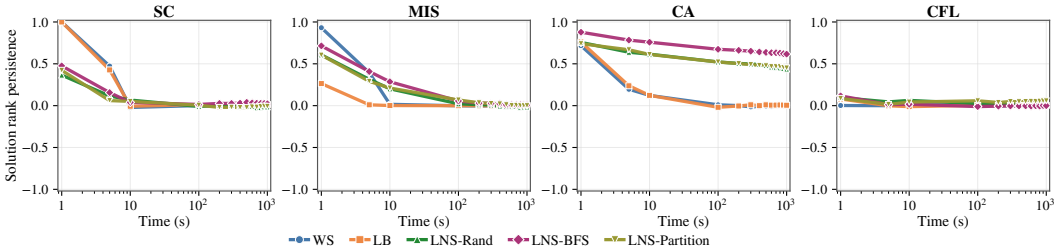



Figure 22: Solution-rank persistence over time across tasks and Refiners. The metric measures whether the pairwise quality ordering of Learner-conditioned initial incumbents is preserved during refinement. Values near 1 indicate preserved ordering, values near 0 indicate washout or ties, and negative values indicate systematic rank reversal.

### I.5.2 Full Results and Discussion for Refiner Behavior Study

These figures decompose the joint trajectories in Fig. 5b) into their temporal components. For each instance-Learner pair, all Refiners receive the same cached initial incumbent, enabling paired comparison of how different search procedures preserve and improve Learner-conditioned solutions.

**Metrics.** Let $g^0_{i,\mathsf{L}}$ denote the nonnegative BKS-relative gap of the initial incumbent produced for instance $i$ by Learner $\mathsf{L}$, and let $g_{i,\mathsf{L},\mathsf{R}}(t)$ denote the best-so-far gap after applying Refiner $\mathsf{R}$ for time $t$. To measure rank persistence, we first construct the set of unordered Learner pairs whose initial qualities are distinguishable:

$$\mathcal{P}_i := \left\{(\mathsf{L},\mathsf{L}') : \mathsf{L} < \mathsf{L}',; g^0_{i,\mathsf{L}} \neq g^0_{i,\mathsf{L}'}\right\}. \tag{27}$$

For each valid pair, its ordering score at horizon $T$ is

$$q_{i,\mathsf{L},\mathsf{L}',\mathsf{R}}(T) = \operatorname{sgn}\left[\left(g^0_{i,\mathsf{L}'} - g^0_{i,\mathsf{L}}\right)\left(g_{i,\mathsf{L}',\mathsf{R}}(T) - g_{i,\mathsf{L},\mathsf{R}}(T)\right)\right]. \tag{28}$$

The score is +1 when the initial ordering is preserved, -1 when it is reversed, and 0 when the two refined solutions are tied. Solution-rank persistence is computed by first averaging across comparable Learner pairs within each instance and then averaging equally across valid instances:

$$P_{\mathsf{R}}(T) := \frac{1}{|\mathcal{I}_T|}\sum_{i\in\mathcal{I}_T}\frac{1}{|\mathcal{P}_i|}\sum_{(\mathsf{L},\mathsf{L}')\in\mathcal{P}_i} q_{i,\mathsf{L},\mathsf{L}',\mathsf{R}}(T), \tag{29}$$

where $\mathcal{I}_T$ contains instances with at least one comparable pair and valid results at horizon $T$. This instance-wise aggregation prevents instances with more distinguishable Learner pairs from receiving disproportionate weight. Importantly, $P_{\mathsf{R}}(T)$ measures retention of the initial ordering, rather than the absolute quality of the refined solutions or causal use of learned guidance.

To quantify improvement, let $\mathcal{A}_T$ contain valid instance-Learner pairs with $g^0_{i,\mathsf{L}} > 0$. Relative cumulative refinement is defined as

$$C_{\mathsf{R}}(T) := \frac{1}{|\mathcal{A}_T|}\sum_{(i,\mathsf{L})\in\mathcal{A}_T}\frac{1}{Tg^0_{i,\mathsf{L}}}\int_0^T \left(g^0_{i,\mathsf{L}} - g_{i,\mathsf{L},\mathsf{R}}(t)\right)\mathrm{d}t. \tag{30}$$

Since, in most cases, $g_{i,\mathsf{L},\mathsf{R}}(t) \geq 0$ holds, the metric lies between 0 and 1. It equals 0 when the initial incumbent is never improved and approaches 1 when its gap is closed immediately. Unlike terminal gap reduction, it rewards improvements that occur earlier and persist throughout the refinement horizon.

**Results.** Figures 21 and 22 reveal a task-dependent tension between persistence and leverage. On SC, the LNS variants improve more rapidly at short horizons, but all Refiners reduce persistence to approximately zero by 10 seconds and achieve cumulative refinement of about 0.93 or higher by 100 seconds. CFL is even more solver dominated: `WS` and `LB` remove approximately 84% of the cumulative initial gap within one second despite almost no rank persistence, and all Refiners approach complete refinement by 100 seconds. In these tasks, sufficiently effective search quickly compresses differences among Learner-conditioned incumbents.

CA exhibits the opposite pattern. At 1000 seconds, `WS` and `LB` achieve cumulative refinement of 0.983 and 0.976 while reducing persistence to approximately zero. By contrast, `LNS-Rand` and `LNS-BFS` retain persistence of 0.438 and 0.616 but achieve only 0.380 and 0.218 cumulative refinement, respectively; `LNS-Partition` closely follows the conservative behavior of `LNS-Rand`. Their higher persistence therefore reflects limited movement from the initial incumbents rather than stronger exploitation of learned information. MIS presents an intermediate regime: `LB` delivers the strongest early improvement and rapidly removes the initial ordering, whereas the LNS variants preserve more ordering at short horizons but improve more gradually. At longer horizons, persistence largely vanishes as cumulative refinement converges.

These results complement the main-text finding that persistence and leverage must be interpreted jointly. High persistence alone may indicate search inertia, while low persistence is not necessarily undesirable when accompanied by substantial paired improvement. A desirable Refiner should instead exploit high-quality initial structure early, preserve only assignments supported by reliable Learner signals, and progressively relax this guidance as the search budget grows. This motivates confidence-, task-, and budget-aware Learner-Refiner co-design rather than treating the Refiner as an independent post-processing module.

## J Minimal Code Example for ML4MILP-Bench Usage

ML4MILP-Bench provides unified interfaces for data generation, reference solution acquisition, configuration for traditional MILP solving, learner training, and modular Learner-Enforcer-Refiner evaluation. The examples below are executable from the repository root after activating the provided environment. We omit nonessential logging and output for compactness.

### J.1 Data Generation and Reference Solutions

`ConsoleBase` provides a common interface and control center for generating instances, exporting standard LP files, and acquiring reference solutions. The following example generates small SC training and validation splits and solves them with Gurobi to construct single-solution supervision.

```
>> from pathlib import Path
>> from ml4milp_bench import ConsoleBase, GurobiSolver, SCGenerator

>> def build_split(name, num_instances, seed):
...     root = Path("Demo") / name
...     generator = SCGenerator(
...         n_constr=200, n_vars=400,
...         density=0.05, seed=seed
...     )
...     solver = GurobiSolver(
...         time_limit=60.0,
...         mip_gap=0.0, threads=1
...     )
...     console = ConsoleBase(
...         task_type="SC",
...         generator=generator,
...         solver=solver
...     )
...     console.generate(num_samples=num_instances, num_threads=1)
...     console.to_lp(root / "instances", write_sol=None, overwrite=True)
...     console.solve(
...         sol_as="ref_sol",
...         sol_dir=root / "ref_sol_gurobi_60s", num_threads=1
...     )
...     return root

>> train_root = build_split("train", 32, seed=0)

[Console] Generation configured:
+------+-------------+-------------+---------+
| Task | Generator   | Num samples | Threads |
+------+-------------+-------------+---------+
| SC   | SCGenerator | 32          | 1       |
+------+-------------+-------------+---------+
Generating instances: 100%|-----| 32/32 [00:01<00:00, 19.27it/s]
[generate] generated 32 instance(s) in 1.66s
Writing LP files: 100%|-----| 32/32 [00:00<00:00, 54.99it/s]
[Console] Saved 32 .lp files to Demo/train/instances, using 0.6s
[Console] Evaluation configured:
+------+-------+------------------+-------+
| Task | Sense | Solver           | Type  |
+------+-------+------------------+-------+
| SC   | min   | GurobiSolver(60.0)| Exact |
+------+-------+------------------+-------+
Solving instances: 100%|-----| 32/32 [00:02<00:00, 12.84it/s]
[Console] 32/32 instances Solved, using 0.0779s/instance

>> val_root = build_split("val", 8, seed=1000)
```

The same interface supports the built-in CA, MIS, and CFL generators, batched generation, LP/MPS serialization, and solution-pool construction. The solving budget determines the strength of the resulting reference labels and does not, by itself, certify optimality.

### J.2 TRADITIONAL MILP SOLVERS

The exact solver wrappers share the same constructor and `ConsoleBase.solve` interface. Switching the backend therefore requires only replacing the solver class.

```
>> from ml4milp_bench import (
...     ConsoleBase, GurobiSolver, ScipSolver,
...     CoptSolver, CplexSolver
... )

>> solver_classes = {
...     "gurobi": GurobiSolver,
...     "scip": ScipSolver,
...     "copt": CoptSolver,
...     "cplex": CplexSolver,
... }
>> backend = "scip"
>> solver = solver_classes[backend](
...     time_limit=60.0, mip_gap=0.0, threads=1, seed=0
... )

>> console = ConsoleBase(task_type="SC", solver=solver)
>> console.from_lp(
...     train_root / "instances",
...     sol_dir=train_root / "ref_sol_gurobi_60s",
...     sol_as="ref_sol"
... )
>> console.solve(
...     sol_as="sol",
...     sol_dir=Path("demo_results") / backend
... )
>> console.evaluate()

Loading LP files: 100%|-----| 32/32 [00:00<00:00, 37.76it/s]
[Console] Loaded 32 .lp files, parsed by Gurobi, using 0.9s
[Console] Evaluation configured:
+------+-------+-----------------+-------+---------+----------+---------+
| Task | Sense | Solver          | Type  | Learner | Enforcer | Refiner |
+------+-------+-----------------+-------+---------+----------+---------+
| SC   | min   |ScipSolver(60.0) | Exact | none    | none     | none    |
+------+-------+-----------------+-------+---------+----------+---------+

Solving instances: 100%|-----| 32/32 [00:26<00:00,  1.19it/s]
[Console] 32/32 instances Solved, using 0.8402s/instance
[Console] Feasibility:
+--------------+-----+--------+---------------+
| IFR          | SFR | Viol   | Viol (infeas) |
+--------------+-----+--------+---------------+
| 32/32 (100%) | N/A | 0 +- 0 | N/A           |
+--------------+-----+--------+---------------+

[Console] Quality:
+-----------------+-----------------+---------+-------------+---------+
| Obj             | Obj(feas.)      | Ref obj | Gap(feas.)  | FQS     |
+-----------------+-----------------+---------+-------------+---------+
| 335.031+-38.073 | 335.031+-38.073 | 335.031 | 0.00%+-0.00% | 100.00% |
+-----------------+-----------------+---------+-------------+---------+
```

The wrappers do not redistribute solver binaries or licenses. Users must install the corresponding official Python API and satisfy its licensing requirements. In particular, Gurobi, COPT, and CPLEX require valid licenses; SCIP requires a compatible PySCIPOpt installation.

### J.3 Learner Construction and Training

Learners are assembled from a unified bipartite backbone, a data environment, and a learning objective. The following example constructs the supervised baseline learner. Following successful practice in Li et al. (2025); Ma et al. (2025b), we wrap the training process with the standard PyTorch-Lightning (Falcon & Team, 2019) interface.

```
>> import torch
>> from ml4milp_bench.learner import (
...     MILPEnv, MILPGNN, MILPVanillaSLModel, Trainer
... )

>> use_cuda = torch.cuda.is_available()
>> env = MILPEnv(
...     task_type="SC", mode="train",
...     train_path=train_root / "instances",
...     val_path=val_root / "instances",
...     train_sol_dir=train_root / "ref_sol_gurobi_60s",
...     val_sol_dir=val_root / "ref_sol_gurobi_60s",
...     data_format="lp",
...     train_batch_size=4, val_batch_size=4,
...     device="cuda" if use_cuda else "cpu",
...     normalize=True
... )

>> backbone = MILPGNN(var_dim=8, con_dim=5, hidden_dim=32, num_layers=6)
>> learner = MILPVanillaSLModel(
...     env=env,
...     backbone=backbone,
...     learning_rate=1e-3,
...     val_decoder="gumbel",
...     val_num_samples=500
... )

>> trainer = Trainer(
...     model=learner,
...     accelerator="gpu" if use_cuda else "cpu",
...     devices=[0] if use_cuda else 1,
...     max_epochs=20,
...     resume_id="sl_sc_demo",
...     ckpt_monitor="val/obj",
...     mode="min",
...     save_top_k=1
... )
>> trainer.model_train()

Loading LP files: 100%|-----| 32/32 [00:00<00:00, 40.28it/s]
[Console] Loaded 32 .lp files, parsed by Gurobi, using 0.8s
Loading LP files: 100%|-----| 32/32 [00:00<00:00, 42.11it/s]
[Console] Loaded 8 .lp files, parsed by Gurobi, using 0.2s

  | Name  | Type    | Params
---------------------------------
0 | model | MILPGNN | 261 K
---------------------------------
261 K     Trainable params
0         Non-trainable params
261 K     Total params
1.045     Total estimated model params size (MB)

Epoch 1: 100%|-----| 8/8 [00:00<00:00,  8.28it/s, v_num=demo,
    ↪ train/loss_step=0.117, val/loss=0.116, val/feas_rate=0.000]
...
```

For reproducible experiments, the same construction is packaged into task-specific entry points; for example, `python train_scripts/sl/train_sl_sc.py` launches the complete SC training configuration.

### J.4 Modular Learner-Enforcer-Refiner Evaluation

A trained learner can be independently paired with an Enforcer and a Refiner. Below, the SL learner predicts variable-wise logits, Predict-and-Search obtains a feasible incumbent using a 10-second Gurobi backend, and a warm-start solver refines that incumbent for 60 seconds.

```
>> from ml4milp_bench.enforcer import PredictSearchEnforcer
>> from ml4milp_bench.refiner import WarmStartSolverRefiner
>> from ml4milp_bench.solver import MILPVanillaSLSolver

>> enforcer = PredictSearchEnforcer(
...     backend="gurobi",
...     center="threshold",
...     k0=0.30, k1=0.04, delta=0.20,
...     time_limit=10.0, threads=1, seed=0
... )

>> refiner = WarmStartSolverRefiner(
...     backend="gurobi",
...     time_limit=60.0, threads=1, seed=0
... )

>> backbone = MILPGNN(var_dim=8, con_dim=5, hidden_dim=32, num_layers=6)
>> solver = MILPVanillaSLSolver.from_checkpoint(
...     ckpt_path="train_ckpts/sl_sc/sc_default/last.ckpt",
...     backbone=backbone,
...     enforcer=enforcer,
...     refiner=refiner,
...     device="cuda" if use_cuda else "cpu"
... )

>> test = ConsoleBase(task_type="SC", solver=solver)
>> test.from_lp(
...     train_root / "instances",
...     sol_dir=train_root / "ref_sol_gurobi_60s",
...     sol_as="ref_sol"
... )
>> test.solve(
...     sol_as="sol",
...     sol_dir="demo_results/sl_ps_warm_start"
... )
>> test.evaluate()

Loading LP files: 100%|-----| 32/32 [00:00<00:00, 36.66it/s]
[Console] Loaded 32 .lp files, parsed by Gurobi, using 0.9s
[Console] Evaluation configured:
+------+-------+----------+--------+---------+------------+------------+
| Task | Sense | Solver   | Type   | Learner | Enforcer   | Refiner    |
+------+-------+----------+--------+---------+------------+------------+
| SC   | min   | SLSolver | Neural | SLModel | PS(Gurobi) | WS(Gurobi) |
+------+-------+----------+--------+---------+------------+------------+

Solving instances: 100%|-----| 32/32 [00:05<00:00,  5.52it/s]
[Console] 32/32 instances Solved, using 0.1813s/instance
[Console] Feasibility:
+--------------+-----+--------+---------------+
```

```
| IFR          | SFR | Viol   | Viol (infeas) |
+--------------+-----+--------+---------------+
| 32/32 (100%) | N/A | 0 +- 0 | N/A           |
+--------------+-----+--------+---------------+

[Console] Quality:
+-----------------+-----------------+---------+--------------+---------+
| Obj             | Obj(feas.)      | Ref obj | Gap(feas.)   | FQS     |
+-----------------+-----------------+---------+--------------+---------+
| 335.031+-38.073 | 335.031+-38.073 | 335.031 | 0.00%+-0.00% | 100.00% |
+-----------------+-----------------+---------+--------------+---------+
```

The learner checkpoint remains unchanged when replacing the Enforcer or Refiner. This enables controlled recombination of learned variable scores, feasibility-enforcement mechanisms, refinement algorithms, and solver backends under a common data and evaluation contract.

## K Limitation and Future Work

ML4MILP is a broad and rapidly evolving field, and the current release of ML4MILP-Bench deliberately concentrates on methods that expose explicit variable-wise primal guidance and can therefore be meaningfully decomposed into the Learner-Enforcer-Refiner pipeline. Consequently, it does not yet exhaustively cover several important research directions, including recent reinforcement-learning-based primal heuristics, the broad family of learned LNS methods with trainable destroy or repair policies, and earlier solver-internal acceleration techniques for branching, cutting, presolving, and related decisions. **This boundary is an intentional delimitation of the present study rather than a methodological or experimental deficiency**: incorporating these paradigms requires additional abstractions, supervision contracts, and evaluation protocols to preserve the same level of controlled component attribution. Within its stated scope, the current benchmark already aligns eight learners, six enforcers, three refiners (with variants), multiple exact-solver backends, diverse problem families, and extensive stage-wise recombinations. Therefore, its empirical coverage is adequately commensurate with that of representative recent studies in top venues and is sufficient to support the conclusions drawn in this work.

Looking forward, we intend to develop ML4MILP-Bench as a continuously maintained *living benchmark*: expanding its modular interfaces to accommodate the above method families, maintaining a versioned and component-aware leaderboard, and providing a practical research base and reusable toolkit for both learning and optimization algorithms. We will also broaden the evaluation suite toward more diverse, larger-scale, and substantially harder instances, with particular emphasis on genuine mixed-integer problems containing both discrete and continuous variables. Together with continuously updated data assets, oracle reference solutions, reproductions, and pretrained checkpoints, these extensions **aim to make ML4MILP-Bench a durable infrastructure for transparent comparison and rapid development in future ML4MILP research**.