LEARNING EARLY-TO-FINAL SOLUTION CONSISTENCY FOR MILP ACCELERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixed-Integer Linear Programming (MILP) is a fundamental problem class in operations research and combinatorial optimization, with broad applications to industrial decision-making. Owing to their NP-hardness, however, modern solvers may struggle to find high-quality solutions for challenging MILP instances within practical time limits. Recent learning-based approaches seek to accelerate MILP solving by directly predicting high-quality solutions from static instance-level features, such as variable-constraint bipartite graphs. Yet accurate solution prediction from instance features alone is difficult, and these methods largely overlook the information revealed during the solver's search process. In this paper, we find that solutions produced at the early search stage of MILP solvers, which are computationally cheap to obtain, are often structurally close to the solutions found after full-budget search. Motivated by this observation, we propose a new solver-informed paradigm that shifts the learning target from variable assignment to *early-to-final consistency*: for each variable, we predict whether its early-stage assignment should persist in full-budget solutions. The predicted consistency naturally guides downstream search, for instance by fixing the assignments deemed consistent. At inference time, we further ensemble consistency predictions across multiple early-stage solutions to improve robustness. Experiments across four MILP benchmarks show our method improves prediction-guided search across diverse downstream pipelines. With Gurobi, our proposed method reduces the primal gap by 56.9% on average and closes it completely on combinatorial auction instances. Besides, we transferred the Gurobi-trained model zero-shot to SCIP without adaptation, achieving a 36.4% average gap reduction across benchmarks.

1 INTRODUCTION

Mixed-Integer Linear Programming (MILP) is a fundamental modeling language for combinatorial optimization and decision making, with broad applications in scheduling (Floudas & Lin, 2005), routing (Laporte, 1992), network design (Crainic, 2000) and production planning (Nam & Logendran, 1992). Prevailing MILP solvers employ branch-and-bound (Land & Doig, 1960) frameworks to systematically explore the combinatorial search space while guaranteeing optimality. Built upon this, the MILP community has developed a variety of sophisticated techniques (Padberg & Rinaldi, 1991; Achterberg, 2009) over the past decades, including presolve, cutting planes, primal heuristics, and branching strategies, which further reduce the search space and accelerate the solving process. However, MILP remains computationally expensive for modern solvers due to its inherent NP-hardness (Karp, 1972).

In real-world applications, most MILP instances are derived from similar problem classes with varying parameters, exhibiting shared structural patterns that can be naturally exploited by machine learning methods. As investigated by Bengio et al. (2021), learning models can capture these patterns from instance characteristics and historical solving experiences, and infer effective optimization strategies. Such learned knowledge can be integrated into MILP solvers to accelerate the optimization process (Gasse et al., 2019; Li et al., 2023; Nair et al., 2020). Existing learning-based methods for MILP can be broadly classified into three types: (1) *learning solver decisions*, where machine learning models are employed to guide individual solver components, such as branching, node selection, and cutting plane selection (Khalil et al., 2016; Gasse et al., 2019; Gupta et al., 2020; Tang et al., 2020; Ling et al., 2024; Strang et al., 2026); (2) *learning solver configurations*, where

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.

models automatically tune algorithmic parameters or select suitable configurations for different instances (Hutter et al., 2009; Li et al., 2023); and (3) *learning solution predictions*, where models directly predict complete or partial solutions to provide high-quality initial solutions or reduce the search space (Nair et al., 2020).

Among the solution prediction methods, recent advances have focused on two directions: improving learning-guided search mechanisms and enhancing prediction capacity. On the search side, early studies such as neural diving (Nair et al., 2020) fixed high-confidence variable assignments predicted by neural models, producing reduced MILP instances for efficient downstream optimization. Building upon this idea, Han et al. (2023) replaced hard variable fixing with a trust-region search strategy, reducing the risk of being trapped by inaccurate assignments. More recently, Liu et al. (2025) proposed to alternate prediction with trust-region correction, further improving robustness against prediction errors. On the prediction side, Wang et al. (2026) represented MILPs as geometric feature sequences and used multi-scale attention to overcome expressiveness limitations of graph-based models. Meanwhile, Li et al. (2026) introduced a flow-matching generator that jointly models integer and continuous variables to enhance solution generation quality, while Pu et al. (2026) exploited variable interactions through inter-variable contrastive learning and an intra-constraint competitive graph network. However, these methods still follow the conventional paradigm of predicting a complete solution directly from the MILP formulation. Under this paradigm, the model is required to determine all variable assignments, leading to a prediction task that approaches the complexity of solving the original MILP.

In this paper, we begin by analyzing how primal solutions evolve during the search of modern MILP solvers. Across all evaluated benchmarks, we observe that the solution quality improves rapidly in the early stage of the search and then progresses much more slowly with only local corrections, indicating that early incumbents are highly informative about the final, full-budget assignments. On workload apportionment, for instance, an early incumbent already agrees with the final solution on 95.63% of the binary variables (as shown in Figure 1(b)). Given such an informative starting point, the optimal solution need not be reconstructed from scratch. Conditioned on what the solver has already produced, the learning task reduces to deciding which of its assignments to trust.

Based on this insight, we propose a new *solver-informed* paradigm that reformulates solution prediction as an *early-to-final consistency estimation* problem. Given an MILP graph G, an early solution X^{ES}, and a reference solution X^* obtained from a full-budget run, conventional solution prediction learns $P(X^* \mid G)$ and thus requires the model to infer all variable assignments from static instance features alone. In contrast, we estimate $P(x_i^* = x_i^{ES} \mid G, X^{ES})$, namely the probability that each early assignment persists in the final solution. This target can substantially ease the learning task. Conditioned on the rich anchor X^{ES}, the model only needs to identify a relatively small fraction of assignments that have yet to stabilize. Theoretically, we prove that conditioning on the early solution can strictly increase the best achievable accuracy of predicting the full-budget solution, under a mild posterior-crossing condition, and that this information gain provably persists under finite-sample model selection. Moreover, since the predicted consistent assignments are drawn from a feasible early solution, fixing them preserves feasibility and concentrates the subsequent search on assignments that the solver has left unresolved.

We evaluate our method on multiple MILP benchmarks with Gurobi (Gurobi Optimization, LLC, 2026) as the target solver and integrate it into existing prediction-based search frameworks. Experimental results show that our proposed refiner consistently reduces the primal gap in most evaluated settings. Specifically, it reduces the primal gap of Predict-and-Search (Han et al., 2023) by 56.9% on average and closes the gap on combinatorial auction instances. Furthermore, without retraining, the model trained with Gurobi-generated data can be directly transferred to SCIP (Achterberg, 2009). These results demonstrate that the proposed early-to-final consistency prediction paradigm is clearly effective and showcases transferability across different solvers.

2 PRELIMINARIES

2.1 MIXED INTEGER LINEAR PROGRAMMING

A mixed-integer linear program (MILP) is an optimization problem with a linear objective and linear constraints, where a subset of decision variables is restricted to integer values. We formulate a MILP

instance as

$$\min_{x \in \mathbf{R}^n} \; \boldsymbol{c}^\top \boldsymbol{x} \quad \text{s.t.} \quad \mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}, \quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \quad x_i \in \mathbf{Z} \; \forall i \in \mathcal{I}, \tag{1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$ is the vector of decision variables, $\boldsymbol{c} \in \mathbf{R}^n$ is the objective coefficient vector, $\mathbf{A} \in \mathbf{R}^{m \times n}$ is the constraint matrix, and $\boldsymbol{b} \in \mathbf{R}^m$ is the right-hand-side vector. The bounds $\boldsymbol{l}, \boldsymbol{u} \in (\mathbf{R} \cup \{\pm\infty\})^n$ define the variable domains, and $\mathcal{I} \subseteq \{1, \ldots, n\}$ is the index set of variables required to take integer values. An important special case is that of binary variables, for which $x_i \in \{0, 1\}$. General integer variables can be transformed into binary variables using standard preprocessing techniques.

2.2 Graph Representations for MILP

Learning-based MILP methods commonly encode an instance as a bipartite graph (Salvagnin, 2016; Gasse et al., 2019). For the MILP formulated in Eq. (1), the associated graph is defined as $G = (\mathcal{V}, \mathcal{C}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{C} = \{q_1, \ldots, q_m\}$ denote the sets of variable nodes and constraint nodes, corresponding to decision variables and constraints, respectively. An edge $(v_i, q_j) \in \mathcal{E}$ is introduced when the coefficient $\mathbf{A}_{ji}$ is non-zero.

Each node is initialized with features describing its role in the MILP (Han et al., 2023). A variable node v_i contains the objective coefficient c_i, the lower and upper bounds l_i and u_i, and an indicator of whether x_i is continuous or integer. A constraint node q_j contains the right-hand-side value b_j and an encoding of its constraint sense, such as $\leq$, $=$, or $\geq$. In addition, each edge carries the matrix coefficient A_{ji} as its edge feature. Collectively, these node and edge attributes provide a description of the local algebraic structure of the MILP instance, which serves as the input representation for graph neural networks (GNNs).

2.3 Solution Predictions for MILP

Solution prediction methods learn a mapping from an MILP instance to variable assignments, which can be exploited to reduce the search space for acceleration. Most existing predictors take the static bipartite graph G as input and employ a graph convolutional network (GCN) or a related message-passing GNN to compute variable embeddings, from which the final predictions are decoded.

Let $h_i^{(\ell)}$ be the embedding of variable node v_i at layer ℓ, and let $g_j^{(\ell)}$ be the embedding of constraint node q_j. A common bipartite message-passing layer first aggregates messages from variables to constraints and then sends messages back to variables:

$$\begin{aligned} g_j^{(\ell+1)} &= \phi_C^{(\ell)} \left(g_j^{(\ell)}, \textstyle\sum_{i:(v_i,q_j)\in\mathcal{E}} \psi_C^{(\ell)}(h_i^{(\ell)}, A_{ji}) \right), \\ h_i^{(\ell+1)} &= \phi_V^{(\ell)} \left(h_i^{(\ell)}, \textstyle\sum_{j:(v_i,q_j)\in\mathcal{E}} \psi_V^{(\ell)}(g_j^{(\ell+1)}, A_{ji}) \right), \end{aligned}$$

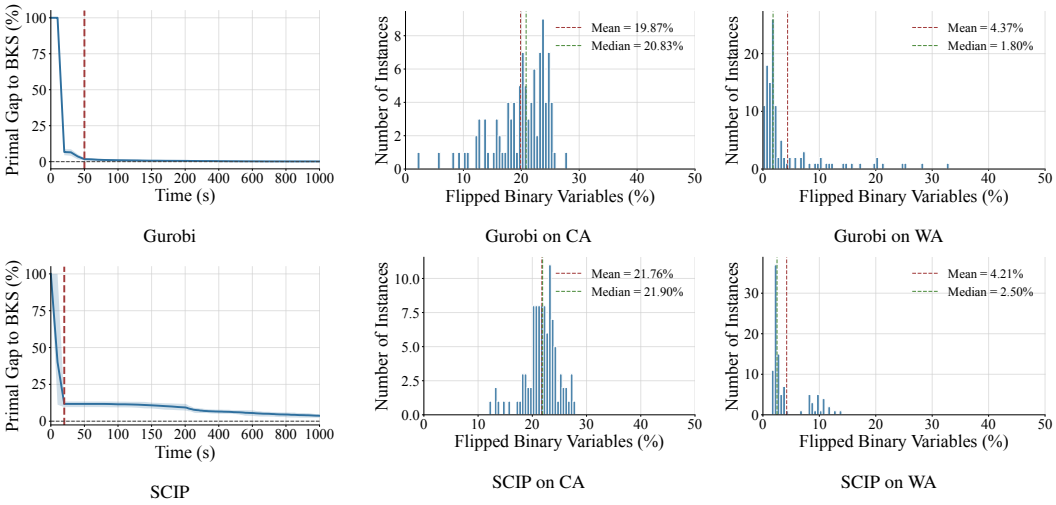
where $\psi_C^{(\ell)}$ and $\psi_V^{(\ell)}$ are message functions, and $\phi_C^{(\ell)}$ and $\phi_V^{(\ell)}$ are node-update functions. These functions are usually implemented as multilayer perceptrons with nonlinear activations. Each update depends on both neighboring node embeddings and the corresponding edge weight A_{ji}.

After L message-passing layers, the predictor applies a variable-level readout to each final variable embedding. For binary integer variables, a common formulation trains a predictor f_θ on the graph G to estimate a target solution X^* as $\hat{p}_i = \sigma\Big(r_\theta(h_i^{(L)})\Big) = f_\theta(G)_i \approx P(x_i^* = 1 \mid G)$, where r_θ is a learnable readout function and σ is the sigmoid function. The output $\hat{p}_i$ is the predicted probability that variable i takes value one in the reference solution. The predictions can then be used to construct a partial assignment, define a neighborhood around a candidate solution, or fix variables before invoking the solver on a reduced MILP.

3 Method

3.1 Motivation

We begin with an observation about the dynamics of modern MILP solvers: the primal and dual gaps typically decrease sharply at the beginning of the solve, after which progress becomes much



(a) Primal gap evolution.

(b) Flipped-variable distributions.

Figure 1: Early-stage evidence motivating EnCore. (a) Primal gap evolution with Gurobi (top) and SCIP (bottom) on set covering, with the time axis scaled to highlight the early phase. (b) Distributions of flipped integer variables between early and full-budget solutions on combinatorial auction (CA) and workload apportionment (WA). Other classes in App. A.

slower and is spread over a long period of search. Figure 1(a) illustrates this behavior with the evolution of the primal gap on example instances; the same qualitative pattern holds across solvers and problem classes (App. A). This behavior suggests a two-stage view of MILP solving. In the *fast descent stage*, the solver rapidly discovers useful early solutions and tightens bounds through presolve, heuristics, cutting planes, and early branch-and-bound decisions. In the *long exploration stage*, the solver spends most of its time searching for further improvements, proving bounds, and resolving difficult variables.

This two-stage view motivates a new prediction target for learning-based MILP solving. In contrast to the prevailing paradigm, which constructs a full solution directly from the static MILP graph, we propose an **E**arly-to-final solutio**n** **Co**nsistency p**re**diction paradigm, termed **EnCore**. Specifically, given an early solution X^{ES} collected after the fast-descent stage, we estimate the probability that the early assignment of each variable persists in the full-budget solution X^*. This target distinguishes assignments that have likely stabilized from those that merit further exploration, thereby directing both learning and search effort toward the decisions that remain open after the fast descent.

The proposed paradigm rests on the premise that the early solution is informative of the final one. We now examine this premise empirically. Figure 1(b) compares each early solution X^{ES} with the full-budget solution X^* by counting the flipped integer variables. Across benchmark classes, the early solutions exhibit agreement with the final ones, and the discrepancies tend to concentrate on a limited subset of variables rather than spreading arbitrarily across the instance. These observations indicate that the early solution carries rich information about the final solution, making it a valuable reference worth exploiting.

3.2 Consistency Prediction

Given the MILP graph G and the collected early solution X^{ES}, we design a consistency predictor that estimates, for each integer variable, the probability that its early assignment agrees with the full-budget solution X^*. The predictor must fuse two heterogeneous sources of information: the structural description of the instance, encoded in the bipartite variable–constraint graph, and the variable assignments in the early solution. We achieve this fusion at the input level, where each integer-variable node is augmented with its early assignment x_i^{ES} as an additional feature. Through rounds of variable-to-constraint and constraint-to-variable message passing, the network can assess each assignment jointly with the constraints it appears in and the assignments of neighboring

variables. The backbone is a bipartite graph neural network with half-convolutions, a standard architecture for MILP representation learning (Han et al., 2023; Liu et al., 2025). A variable-level readout then produces the consistency logit $\hat{z}_i$ for each integer variable, from which the consistency probability is obtained as $\hat{p}_i = \sigma(\hat{z}_i)$. Architectural details are provided in Appendix B.

To train the predictor, we fundamentally reshape the learning target, rather than imitating the complete solutions X^*. Since the early solution already encodes substantial information about the final one, the remaining task is not to predict every variable value from scratch, but to assess which early assignments are likely to persist in the final solution. We therefore define the binary consistency label $y_i = \mathbf{1}[x_i^* = x_i^{ES}]$ for each integer variable i, and reformulate the prediction target as

$$\underbrace{\boxed{P\big(x_i^* = 1 \mid G\big)}}_{\text{Solution Prediction}} \quad \longrightarrow \quad \underbrace{\boxed{P\big(x_i^* = x_i^{ES} \mid G, X^{ES}\big)}}_{\text{Consistency Prediction}}$$

For each training instance G, we collect an early solution X^{ES} and pair it with the full-budget solution X^* (the collection procedure is described later in this section), and train f_θ on the consistency labels with the binary cross-entropy loss, $\mathcal{L}(\theta) = -\sum_{i\in\mathcal{I}}\big[y_i \log \hat{p}_i + (1-y_i)\log(1-\hat{p}_i)\big]$, so that $\hat{p}_i = f_\theta(G, X^{ES})_i$ approximates the consistency probability above. This consistency target reshapes the learning task: rather than constructing the full solution from the static graph alone, the model refines an informative reference solution by assessing the stability of each assignment, which is a more focused and better-conditioned problem. Moreover, the conditioning solution X^{ES} provides a rich source of instance information that purely graph-based predictors cannot access.

Early Solution Collection. We collect early solutions at the transition from *fast descent* to *long exploration*. During the solving process, we monitor feasible-solution improvement events and estimate the local dual-gap decay rate over a sliding window, i.e., the gap reduction within the window divided by the elapsed time. The run is terminated once this rate first falls below a threshold, indicating that rapid bound improvement has mostly ended, and the last explored solution is taken as the early solution X^{ES}. In practice, we additionally impose a minimum probing time and a maximum collection time; details are included in App. C.

Inference-Time Augmentation. The early solution available for conditioning is not unique: depending on when the collection run terminates, the solver may return slightly different solutions, and a variable predicted stable under one of them may still be predicted fluctuating under another. To mitigate this sensitivity, we employ the last few improving solutions for inference-time augmentation. Let $\{X_1, \ldots, X_K\}$ denote the last K improving solutions found during the collection run, with $X_K = X^{ES}$. The trained predictor processes each X_k as the conditioning solution. Since a consistency score is always expressed relative to the value taken in its own conditioning solution, the resulting predictions must be aligned to the common reference X^{ES}. To this end, we flip the sign of each logit whose variable takes a different value in X_k than in X^{ES}, as a high consistency score for such a variable argues against the reference value. The aligned logits are averaged, $\bar{p}_i = \sigma\big(\frac{1}{K}\sum_{k=1}^{K} \tau_{k,i}\hat{z}_{k,i}\big)$, where $\hat{z}_{k,i}$ is the consistency logit produced under the conditioning solution X_k, and $\tau_{k,i} = +1$ if $x_{k,i} = x_i^{ES}$ and -1 otherwise. The ensembled probability $\bar{p}_i$ replaces $\hat{p}_i$ as the consistency score of x_i^{ES} used by the downstream search methods.

3.3 Combine EnCore with Search Methods

Our consistency predictor is independent of the downstream search strategy. In most learning-based MILP accelerators, the method must select a subset of variables whose predicted values are trusted, and the search space is then reduced accordingly. The central question is therefore which assignments can be trusted. Existing methods trust an assignment when the model is confident in its own prediction, so both the value and its evidence come from the model alone. In contrast, our *early-to-final consistency* design trusts values that the solver has already realized in a feasible early solution, and scores them by predicting whether they persist after the full solving process. Specifically, we rank variables by their consistency score $\bar{p}_i$ and select the top-ranked ones. Each selected variable i is constrained to its early value x_i^{ES}, while the remaining variables are left to the downstream search mechanism. This consistency-based design offers two main advantages: (1) *Feasibility*. The fixed values are drawn from a feasible solution, so fixing them never destroys feasibility, which model-generated assignments cannot guarantee. (2) *Alignment with the search process*. The consistency

signals originate from the solver's own search trajectory, and thus naturally guide the model toward the variables that the solver has yet to resolve.

4 THEORETICAL ANALYSIS

Our method depends on the premise that conditioning on the early solution X^{ES} makes predicting the full-budget solution X^* fundamentally easier than predicting from the instance graph alone. In this section, we formalize it by answering two questions: (1) Can the early solution improve the best achievable prediction accuracy, and under what condition is the gain strict (**Theorem 1**)? (2) Does this gain survive when the predictor must be selected from finite training instances (**Theorem 2**)? Complete proofs are in Appendix F.

Let $\mathcal{D}$ be the distribution over triples (G, X^{ES}, X^*), and let J be a uniformly weighted variable index within an instance, so that $\Pr(\widehat{x}_J = x^*_J)$ is the expected per-instance variable accuracy, i.e., the population variable-level evaluation criterion used in our analysis. A message-passing predictor at variable J considers only the instance features in its receptive MILP graph $\mathcal{R}_J$, while EnCore additionally attaches early assignments $X^{ES}_{\mathcal{R}_J}$ to construct the augmented input $\mathcal{U}_J = (\mathcal{R}_J, X^{ES}_{\mathcal{R}_J})$.

A solution predictor h maps $\mathcal{R}_J$ to a prediction of x^*_J, matching the conventional paradigm. A consistency predictor a observes $\mathcal{U}_J$ and decides whether to retain the early value, inducing $h_a(\mathcal{U}_J) = x^{ES}_J$ if $a(\mathcal{U}_J) = 1$ and $h_a(\mathcal{U}_J) = 1 - x^{ES}_J$ if $a(\mathcal{U}_J) = 0$. Given x^{ES}_J, consistency decisions and final assignments determine each other (see Appendix F), so the two paradigms can be compared as predictors of x^*_J.

Information gain of the early solution. Define the best population accuracies attainable from each input as $A^*_{\text{sol}} := \sup_h \Pr\big(h(\mathcal{R}_J) = x^*_J\big)$ and $A^*_{\text{con}} := \sup_a \Pr\big(h_a(\mathcal{U}_J) = x^*_J\big)$. If $\mathcal{D}$ were known, these would be achieved by the Bayes-optimal rules on each input (Devroye et al., 1996). Their comparison isolates the informational value of the early solution from any modeling concern.

Theorem 1. *For any fixed early-solution collection procedure,* $A^*_{\text{con}} \geq A^*_{\text{sol}}$. *Let* $\eta_J := \Pr(x^*_J = 1 \mid \mathcal{U}_J)$. *The inequality is strict if* $\Pr\big(\Pr(\eta_J < \frac{1}{2} \mid \mathcal{R}_J) > 0$ *and* $\Pr(\eta_J > \frac{1}{2} \mid \mathcal{R}_J) > 0\big) > 0$.

Proof sketch. Every solution predictor h defines $a_h(\mathcal{U}_J) = \mathbf{1}\{h(\mathcal{R}_J) = x^{ES}_J\}$, for which $h_{a_h}(\mathcal{U}_J) = h(\mathcal{R}_J)$. Hence the augmented input can reproduce every solution predictor. For strictness, conditional Jensen's inequality (Kallenberg, 2021) shows that refining the input strictly increases Bayes accuracy under the stated condition. □

The inequality itself is unsurprising since additional input can never hurt a Bayes-optimal predictor. The substantive content is about the strictness condition, which we call *posterior crossing*: for a non-negligible set of instance inputs, different early assignments compatible with the same instance structure favor different final values. In this regime, no predictor on instance features alone, however expressive, can resolve the ambiguity, while the early solution can. Figure 1(b) provides complementary empirical motivation by showing that early-to-final discrepancies are often concentrated on a small fraction of variables.

A sparse-correction reading of the gain. Let a^* be the Bayes rule on the augmented input. Define $p = \Pr(x^{ES}_J \neq x^*_J)$ as the early-solution error rate and $\rho = \Pr(a^*(\mathcal{U}_J) = 0)$ as its flipping rate. When $\rho > 0$, let q denote the precision on flipped variables; Bayes optimality implies $q > 1/2$. Then we have $1 - A^*_{\text{con}} = p - \rho(2q - 1)$ and

$$A^*_{\text{con}} - A^*_{\text{sol}} = 1 - A^*_{\text{sol}} - p + \rho(2q - 1).$$

The gain is positive exactly when $p < 1 - A^*_{\text{sol}} + \rho(2q-1)$. This identity captures the residual nature of EnCore: corrections with $q > 1/2$ remove more errors than they introduce, and when early-to-final discrepancies are sparse, learning can focus on a targeted subset of unstable assignments. The condition can also hold when $1 - p < A^*_{\text{sol}}$ (see Appendix F), showing that the early solution is valuable not only as a feasible candidate but also as an informative conditioning signal.

Finite-sample guarantee. Theorem 1 optimizes over all rules and thus characterizes only the information available in principle. However, the predictor is selected from finite data in practice, and

one may worry whether the population gain can survive estimation errors. Following the standard finite-class empirical-risk-minimization analysis based on Bernstein's inequality and a union bound (Boucheron et al., 2013), we fix a finite class $\mathcal{A} = \{a_1, \ldots, a_N\}$ of consistency rules before observing m independent training instances, and let $\widehat{a}$ minimize the average proportion of variable errors; only instances need be independent, matching how MILP training data are collected. The class advantage decomposes as $\Delta_{\mathcal{A}} = \Delta_{\mathrm{B}} - \alpha_{\mathcal{A}}$, where $\Delta_{\mathrm{B}} := A^*_{\mathrm{con}} - A^*_{\mathrm{sol}}$ is the gain from Theorem 1 and $\alpha_{\mathcal{A}} := A^*_{\mathrm{con}} - \max_{a\in\mathcal{A}} \mathrm{Acc}(h_a)$ is the approximation gap, with $\mathrm{Acc}(h) = \Pr(h_J = x^*_J)$. Motivated by the sparse discrepancies in Figure 1(b), we assume the expected per-instance flip rate satisfies $\max_{a\in\mathcal{A}} \mathbb{E}[s_a] \le \bar{s}$.

Theorem 2. *Let $\widehat{h}_{ES} = h_{\widehat{a}}$, and let $\widehat{h}_{\mathrm{sol}}$ be any possibly data-dependent solution predictor. For every $\delta \in (0,1)$, with probability at least $1-\delta$, $\mathrm{Acc}(\widehat{h}_{ES}) - \mathrm{Acc}(\widehat{h}_{\mathrm{sol}}) \ge \Delta_{\mathrm{B}} - \alpha_{\mathcal{A}} - \varepsilon_m(\delta)$, where*

$$\varepsilon_m(\delta) = 2\sqrt{\frac{2\bar{s}}{m}\ln\frac{2N}{\delta}} + \frac{8}{3m}\ln\frac{2N}{\delta}.$$

Proof sketch. Use the rule that always keeps the early solution as a common reference. A candidate differs from it only on variables it flips, so its instance-level loss difference has variance at most $\bar{s}$. Bernstein's inequality with a union bound over the N rules (Boucheron et al., 2013) then controls all empirical loss differences simultaneously. Empirical risk minimization converts this into the penalty $\varepsilon_m(\delta)$, while Bayes optimality lower-bounds the risk of every instance-input solution predictor. □

Thus the learned consistency predictor provably outperforms *any* solution predictor, even one trained with unlimited data, whenever $\Delta_{\mathrm{B}} > \alpha_{\mathcal{A}} + \varepsilon_m(\delta)$. The dominant term of $\varepsilon_m(\delta)$ scales as $\sqrt{\bar{s}/m}$ rather than $\sqrt{1/m}$: sparse correction rules are cheaper to select reliably. A numerical illustration in Appendix F shows that under representative parameters, fewer than 200 training instances suffice, similar to typical training-set sizes.

In summary, Theorem 1 establishes that the early solution provides a strict population-level information gain under the posterior-crossing condition, and the sparse-correction identity attributes the gain to repairing a small, better-than-chance set of unstable assignments. Theorem 2 shows that the gain persists under finite-sample model selection, with a sample cost discounted by the very sparsity that motivates our design. Together, they justify shifting the learning target from full solution prediction to early-to-final consistency estimation.

5 Experiments

5.1 Experimental Setup

Benchmarks. Following Liu et al. (2025), we evaluate EnCore on Combinatorial Auctions (CA) (Leyton-Brown et al., 2000), Set Covering (SC) (Balas & Ho, 1980), Workload Apportionment (WA), and Item Placement (IP) (Gasse et al., 2022). These benchmarks cover MILPs with diverse scales and structures. For zero-shot transfer, we additionally evaluate on the eleven-instance MIPLIB IIS benchmark considered by Liu et al. (2025), following the IIS setting introduced by Wang et al. (2024). The benchmark is a structurally related subset of MIPLIB centered on binary set-covering formulations, while exhibiting substantial variation in problem size, constraint structure, and sparsity. Dataset sources and instance statistics are reported in App. D.

Baselines and variants. We compare against the native MILP solver and three prediction-guided search pipelines: Neural Diving (ND) (Nair et al., 2020), Predict-and-Search (PS) (Han et al., 2023), and Apollo-MILP (Apollo) (Liu et al., 2025). For each learning-based pipeline, we replace its original solution predictor with our consistency predictor while keeping other configurations; the resulting methods are denoted as EnCore-ND, EnCore-PS, and EnCore-Apollo. This design isolates the contribution of the proposed consistency prediction mechanism from downstream solver policies.

Evaluation protocol. All methods use a 1,000-second end-to-end budget that includes early-solution collection, inference, and downstream solving, except for the MIPLIB IIS transfer experiment, which uses a 3,600-second budget. For a fair comparison, EnCore does not use the collected early solutions to warm-start the final search. For EnCore-Apollo, early solutions are collected once

Table 1: Main results with Gurobi under a 1,000-second budget for all methods; EnCore's budget also includes early-solution collection and inference. Bold marks the best learning-based result.

	CA ↑ (BKS 98627.99)		SC ↓ (BKS 123.37)		WA ↓ (BKS 706.86)		IP ↓ (BKS 11.72)	
Method	Obj	Gap	Obj	Gap	Obj	Gap	Obj	Gap
Gurobi (3600s, reference run)	98448.84	179.15	123.37	0.00	706.86	0.00	11.72	0.00
Solver and prediction-guided baselines								
Gurobi (1000s)	97311.69	1316.30	123.64	0.27	707.36	0.50	13.77	2.05
ND	94340.63	4287.36	123.62	0.25	707.10	0.24	14.15	2.43
PS	97906.20	721.79	123.60	0.23	707.09	0.23	12.08	0.36
Apollo	98083.79	544.20	123.56	0.19	707.06	0.20	11.97	0.25
Ours: early-to-final consistency predictor								
EnCore-ND	97847.92	780.07	123.58	0.21	707.03	0.17	13.47	1.75
EnCore-PS	**98627.99**	**0.00**	**123.50**	**0.13**	**706.98**	**0.12**	11.95	0.23
EnCore-Apollo	98491.40	136.59	123.57	0.20	707.03	0.17	**11.82**	**0.10**
Best Gap Reduction	721.79	**100.0%**	0.10	**43.5%**	0.11	**47.8%**	0.15	**60.0%**

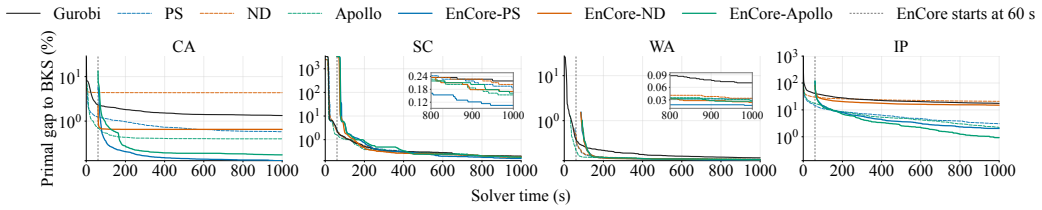


Figure 2: Average primal gap to the BKS versus time under a 1,000-second budget. EnCore starts after early-solution collection; each curve appears only once all test instances are feasible.

and reused across correction rounds. We report the final feasible objective, its absolute gap to the best-known solution (BKS), and the gap reduction relative to the corresponding solution predictor baseline. The BKS is the best mean final objective among all evaluated methods, including the 3,600-second Gurobi reference.

5.2 Main Results

Table 1 reports the final objectives under Gurobi. Substituting EnCore for the original solution predictor reduces the final gap in 11 of the 12 settings (4 benchmarks $\times$ 3 search mechanisms), by 38.7%, 56.9%, and 36.2% on average for ND, PS, and Apollo, respectively. The sole exception is Apollo on SC, where the gap grows marginally to 0.20. On CA, EnCore-PS even finds the best solution among all compared methods, improving on the 3,600-second Gurobi incumbent by 179.15 within a 1,000-second budget, and EnCore-Apollo also exceeds this reference (+42.56). On SC, WA, and IP, where the baselines have already been close to the BKS, EnCore still removes 43.5%, 47.8%, and 60.0% of the best baseline gap. Remarkably, EnCore remains effective even under weak early-to-final consistency: on CA, where about 20% of the variables flip between the early and final solutions, the consistency predictor still captures the learnable patterns underlying these flips, allowing EnCore to achieve a 100% average gap reduction.

Figure 2 traces the primal gap over time: although the best EnCore variant on each benchmark begins prediction-guided search only after the 60-second collection stage, it catches up with its corresponding baseline at 71, 241, 164, and 93 seconds on CA, SC, WA, and IP, respectively, and remains ahead thereafter. EnCore-PS overtakes all competitors early on CA and attains the lowest gap on SC and WA, with the late-stage separation shown in the insets. On IP, EnCore-Apollo exhibits the strongest late-stage anytime performance.

Zero-shot cross-solver transfer to SCIP. We directly apply the Gurobi-trained checkpoint to SCIP-generated trajectories without retraining or model selection. As shown in Table 2, EnCore-ND consistently improves all four benchmarks, reducing the absolute gap by 53.6%, 22.0%, 19.1%, and 50.7% on CA, SC, WA, and IP. Moreover, EnCore-PS also improves performance on CA, WA, and IP. The learned consistency representation thus remains informative under solver shift.

Table 2: Zero-shot transfer from Gurobi to SCIP under a 1,000-second budget. Bold: best result.

	CA ↑ (BKS 98627.99)		SC ↓ (BKS 123.37)		WA ↓ (BKS 706.86)		IP ↓ (BKS 11.72)	
Method	Obj	Gap	Obj	Gap	Obj	Gap	Obj	Gap
SCIP (1000s)	94701.91	3926.08	127.99	4.62	709.06	2.20	23.49	11.77
ND	94342.12	4285.87	125.14	1.77	707.33	0.47	18.19	6.47
PS	97097.46	1530.53	125.03	1.66	708.49	1.63	17.02	5.30
Apollo	97335.92	1292.07	124.97	1.60	708.43	1.57	16.21	4.49
EnCore-ND	96641.43	1986.56	**124.75**	**1.38**	**707.24**	**0.38**	**14.91**	**3.19**
EnCore-PS	**97707.64**	**920.35**	125.04	1.67	708.22	1.36	16.13	4.41
EnCore-Apollo	97293.51	1334.48	125.38	2.01	707.91	1.05	17.39	5.67
Best Improvement	2299.31	**53.6%**	0.39	**22.0%**	0.52	**33.1%**	3.28	**50.7%**

Zero-shot cross-family transfer to MIPLIB IIS. We train our EnCore models on the mixed SC, CA, WA, and IP instances, and evaluate them on the eleven unseen MIPLIB instances used by Liu et al. (2025) under a 3,600-second budget for both the baselines and our variants. The clearest advantage of EnCore is about feasibility. With ND, the original predictor finds feasible solutions on only 3 of 11 instances, while EnCore-ND succeeds on all 11. With PS and Apollo, where the baselines already achieve full feasibility, EnCore still yields consistent improvements in mean objectives. These results demonstrate that our proposed EnCore exhibits greater reliability and robustness than existing methods when faced with substantial distribution shifts. Details are provided in Appendix D.

5.3 Ablation Study

Table 3 presents a cumulative ablation under the Predict-and-Search pipeline. (1) *+ES feature*: add the early solution to variable-node features and keep the original target; (2) *+Consistency target*: switch the target to consistency labels with early solution as feature; (3) *+ES ensemble*: full EnCore setting with ensemble-augmented inference. The ES feature improves CA, SC, and WA but slightly degrades IP, while the consistency target further improves performance on all four benchmarks and eliminates this adverse effect on IP. The ensemble delivers additional gains on CA, SC, and WA.

Table 3: Ablation under the Predict-and-Search pipeline.

Variant	CA ↑	SC ↓	WA ↓	IP ↓
Predict-and-Search	97906.20	123.60	707.09	12.08
+ES feature	98318.22	123.55	707.04	12.17
+Consistency target	98616.65	123.53	706.99	**11.90**
+ES ensemble	**98627.99**	**123.50**	**706.98**	11.95

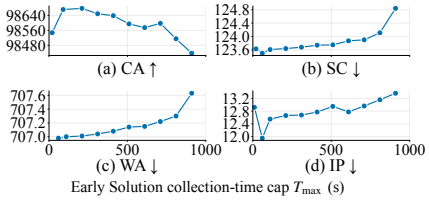


Figure 3: Objective vs. $T_{\max}$.

Trade-off Analysis of Early-Solution Collection. Figure 3 varies the collection budget $T_{\max}$ under a fixed 1,000-second budget. Across all benchmarks, performance peaks at a small $T_{\max}$: a short collection suffices to obtain informative early solutions, whereas larger $T_{\max}$ leads to diminishing improvements in early-solution quality while consuming time budget for downstream search.

6 Conclusion

We presented EnCore, a solver-informed paradigm that accelerates MILP solving via early-to-final solution consistency prediction. Rather than constructing complete solutions from the static formulation alone, EnCore conditions on feasible solutions collected during the early search stage and identifies the assignments likely to persist under a full budget. The resulting consistency scores integrate with diverse prediction-guided search frameworks, and ensembling multiple early solutions further improves robustness. Theoretically, conditioning on early solutions yields a strict gain in achievable prediction accuracy that survives finite-sample model selection, discounted by correction sparsity. Empirically, EnCore improves prediction-guided solving across four benchmarks and transfers zero-shot to unseen solvers and problem families.

REPRODUCIBILITY STATEMENT

The model architecture, early-solution collection procedure, experimental protocols, and hyperparameter settings are fully specified in Appendices B, C, D, and E. Complete proofs of the theoretical results are provided in Appendix F.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing of the manuscript. We have not used generative AI tools for research ideation, experimental design, or data analysis. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

Egon Balas and Andrew Ho. *Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization: A Computational Study*, volume 12, pp. 37–60. Springer Berlin Heidelberg, 1980.

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.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, Oxford, 2013.

Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Stochastic Modelling and Applied Probability*. Springer, New York, 1996.

Christodoulos A. Floudas and Xiaoxia Lin. Mixed integer linear programming in process scheduling: Modeling, algorithms, and applications. *Annals of Operations Research*, 139(1):131–162, 2005.

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*, volume 32, Vancouver, BC, Canada, 2019.

Maxime Gasse, Simon Bowly, Quentin Cappart, Jonas Charfreitag, Laurent Charlin, Didier Chételat, Antonia Chmiela, Justin Dumouchelle, Ambros M. Gleixner, Aleksandr M. Kazachkov, Elias B. Khalil, Pawel Lichocki, Andrea Lodi, Miles Lubin, Chris J. Maddison, Christopher Morris, Dimitri J. Papageorgiou, Augustin Parjadis, Sebastian Pokutta, Antoine Prouvost, Lara Scavuzzo, Giulia Zarpellon, et al. The machine learning for combinatorial optimization competition (ML4CO): Results and insights. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, pp. 220–231, 2022.

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, 2021.

Prateek Gupta, Maxime Gasse, Elias B. Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In *Advances in Neural Information Processing Systems 33*, volume 33, pp. 18087–18097, Virtual, 2020.

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

Qingyu Han, Linxin Yang, Qian Chen, Xiang Zhou, Dong Zhang, Akang Wang, Ruoyu Sun, and Xiaodong Luo. A GNN-Guided Predict-and-Search Framework for Mixed-Integer Linear Programming. In *The 11th International Conference on Learning Representations*, Kigali, Rwanda, 2023.

Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

Olav Kallenberg. *Foundations of Modern Probability*. Springer, Cham, 3rd edition, 2021.

Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of the Symposium on the Complexity of Computer Computations*, pp. 85–103, Yorktown Heights, New York, USA, 1972.

Elias B. Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume 30, pp. 724–731, Phoenix, AZ, USA, 2016.

Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992. ISSN 0377-2217.

Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pp. 66–76, Minneapolis, MN, USA, 2000.

Hongpei Li, Hui Yuan, Han Zhang, Jianghao Lin, Dongdong Ge, Mengdi Wang, and Yinyu Ye. FMIP: Joint continuous-integer flow for mixed-integer linear programming. In *The 14th International Conference on Learning Representations*, Rio de Janeiro, Brazil, 2026.

Sirui Li, Wenbin Ouyang, Max B. Paulus, and Cathy Wu. Learning to configure separators in branch-and-cut. In *Advances in Neural Information Processing Systems*, volume 36, pp. 60021–60034, New Orleans, LA, USA, 2023.

Haotian Ling, Zhihai Wang, and Jie Wang. Learning to stop cut generation for efficient mixed-integer linear programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20759–20767, Vancouver, BC, Canada, 2024.

Haoyang Liu, Jie Wang, Zijie Geng, Xijun Li, Yuxuan Zong, Fangzhou Zhu, Jianye Hao, and Feng Wu. Apollo-MILP: An alternating prediction-correction neural solving framework for mixed-integer linear programming. In *The 13th International Conference on Learning Representations*, Singapore, 2025.

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:2012.13349*, 2020.

Sang-jin Nam and Rasaratnam Logendran. Aggregate production planning — a survey of models and methodologies. *European Journal of Operational Research*, 61(3):255–272, 1992.

Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

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*, pp. 24882–24890, Singapore, 2026.

Domenico Salvagnin. Detecting semantic groups in MIP models. In *Integration of AI and OR Techniques in Constraint Programming*, pp. 329–341, 2016.

Paul Strang, Zacharie Alès, Côme Bissuel, Olivier Juan, Safia Kedad-Sidhoum, and Emmanuel Rachelson. Planning in branch-and-bound: Model-based reinforcement learning for exact combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25627–25635, 2026.

Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 9367–9376, Vienna, Austria, 2020.

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

Ruobing Wang, Xin Li, and Mingzhong Wang. MILPnet: A multi-scale architecture with geometric feature sequence representations for advancing MILP problems. In *The 14th International Conference on Learning Representations*, 2026.

A SOLVER BEHAVIOR

Figures 4 and 5 show, for Gurobi and SCIP on the four benchmarks, the primal gap versus solving time (top row) and the distributions of integer variables flipped between the early and full-budget solutions (bottom row). Figure 6 reports the same analyses on MIPLIB.

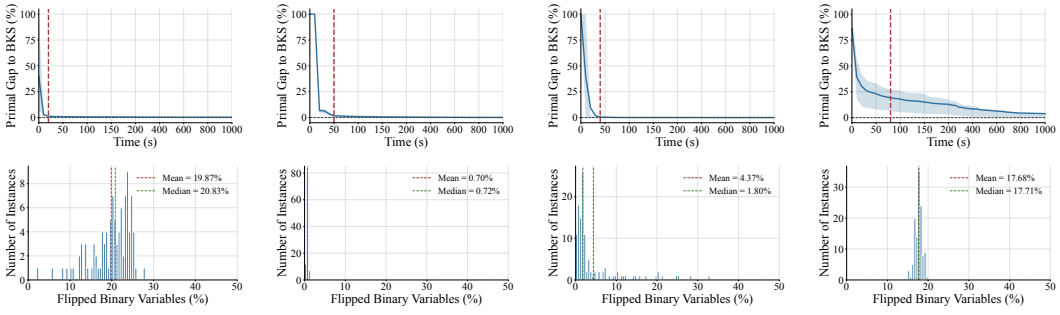


Figure 4: Solver behavior with Gurobi on the four benchmarks. Top row: primal gap versus solving time. Bottom row: distributions of integer variables flipped between the early and full-budget solutions; red and green dashed lines denote the mean and median.

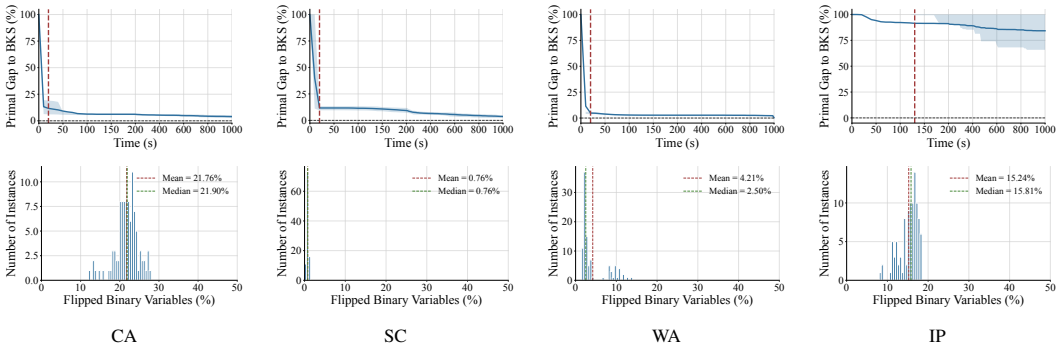
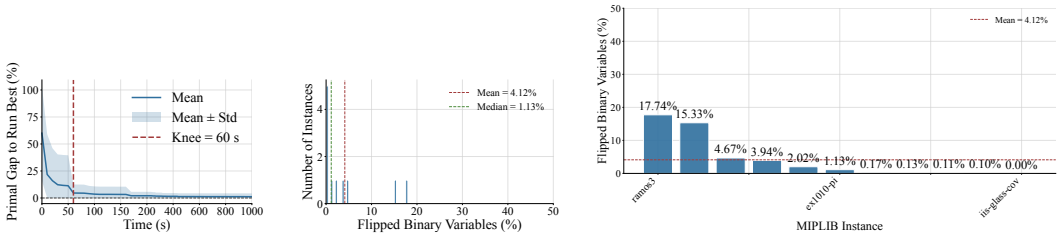


Figure 5: Solver behavior with SCIP on the four benchmarks. Top row: primal gap versus solving time. Bottom row: distributions of integer variables flipped between the early and full-budget solutions; red and green dashed lines denote the mean and median.



(a) Primal gap over time. (b) Aggregate flip distribution. (c) Per-instance flip percentages.

Figure 6: Solver behavior on the MIPLIB IIS dataset with Gurobi.

B MODEL STRUCTURE

Graph Representation. We follow the MILP representation of Han et al. (2023), adding only one dimension to each variable node to encode its early-solution value. Detailed feature definitions are provided in Table 4.

Table 4: Graph features for input.

Index	Feature	Description
Variable-node features		
0	Objective	Normalized objective coefficient.
1	Variable coefficient	Average variable coefficient across all constraints.
2	Variable degree	Degree of the variable node in the bipartite graph.
3	Maximum coefficient	Maximum variable coefficient across all constraints.
4	Minimum coefficient	Minimum variable coefficient across all constraints.
5	Variable type	Indicator of whether the variable is integer.
6–17	Position embedding	Binary encoding of the variable's order among all variables.
18	Early-solution value	Value of the variable in the early solution collected as described in Appendix C.
Constraint-node features		
0	Constraint coefficient	Average of the nonzero coefficients in the constraint.
1	Constraint degree	Degree of the constraint node in the bipartite graph.
2	Bias	Normalized right-hand side of the constraint.
3	Sense	Sense of the constraint.
Edge features		
0	Coefficient	Coefficient connecting the constraint and variable nodes.

Network architecture. We strictly follow the network architecture proposed by Han et al. (2023), modifying only the variable encoder to accommodate the additional early-solution feature. Each variable node is represented by 19 input features, consisting of the original 18 features and the early-solution value introduced by our method. Constraint nodes and edges have 4 and 1 input features, respectively. All node representations are embedded into a 64-dimensional latent space.

For an input feature vector of dimension d, the node encoder is defined as

$$\mathrm{Enc}_d(x) = \mathrm{ReLU}(W_2\ \mathrm{ReLU}(W_1\,\mathrm{LN}(x)))\,,$$

where $W_1 \colon \mathbb{R}^d \to \mathbb{R}^{64}$ and $W_2 \colon \mathbb{R}^{64} \to \mathbb{R}^{64}$. Accordingly, the variable and constraint encoders use Enc_{19} and Enc_4, respectively, while the scalar edge feature is normalized by $\mathrm{LayerNorm}(1)$.

The model performs two rounds of bidirectional message passing between variable and constraint nodes:

$$V \to C \to V \to C \to V.$$

For a directed edge $j \to i$, the message is computed as

$$m_{ij} = W_M\ \mathrm{ReLU}[\mathrm{LN}(W_L h_i + W_E e_{ij} + W_R h_j)]\,,$$

where h_i and h_j denote the target and source node representations, and e_{ij} is the encoded edge feature. Incoming messages are summed and normalized:

$$\bar{m}_i = \mathrm{LN}\left(\sum_{j \in \mathcal{N}(i)} m_{ij}\right).$$

The target node representation is then updated by

$$h_i' = W_{U,2}\ \mathrm{ReLU}(W_{U,1}[h_i \,\|\, \bar{m}_i])\,,$$

where $\|$ denotes concatenation, $W_{U,1} \colon \mathbb{R}^{128} \to \mathbb{R}^{64}$, and $W_{U,2} \colon \mathbb{R}^{64} \to \mathbb{R}^{64}$.

Finally, each variable representation is mapped to a scalar logit:

$$\ell_v = w_{\mathrm{out}}^{\top}\,\mathrm{ReLU}(W_{\mathrm{out}} h_v)\,,$$

where the final output layer has no bias. The predicted probability $\sigma(\ell_v)$ indicates whether the early-solution value of variable v is consistent with its value in the full-budget reference solution.

C Early Solution Collection Details

We first quantify the progress of the dual bound using the average dual-gap decay rate, and then describe the early-solution collection procedure in Algorithm 1. Specifically, for the k-th monitoring interval, the decay rate is defined as

$$r_k = \frac{|D_{k_a} - D_{k_b}|}{t_{k_b} - t_{k_a}}, \tag{2}$$

where k_a and k_b denote the observations at the beginning and end of the interval, respectively; D_{k_a} and D_{k_b} are the corresponding dual gaps; and t_{k_a} and t_{k_b} are their wall-clock times. Thus, r_k measures the average absolute reduction in the dual gap per unit time during the k-th monitoring interval.

Algorithm 1 Early solution collection

Require: MILP instance G, minimum probing time $T_{\min}$, maximum probing time $T_{\max}$, window size w, decay threshold ϵ
Ensure: Early solution X^{ES} and probing time t_{probe}

```
Initialize an empty callback record list R.
Start a probing solver run on G.
while the solver is running do
    if a MIPSOL callback occurs at time t_k then
        Record the explored solution X_k and relative dual gap D_k.
        Append (t_k, X_k, D_k) to R.
        if |R| ≥ w then
            Let k_a and k_b be the first and last records in the latest w-record window.
            Compute r_k by Eq. (2).
            if t_k ≥ T_min and r_k < ε then
                Stop the probing run.
            end if
        end if
    end if
    if the elapsed probing time reaches T_max then
        Stop the probing run.
    end if
end while
Set X^ES to the last explored solution during probing.
Set t_probe to the elapsed probing time.
return X^ES and t_probe.
```

D Experimental Details

Benchmark Details

We evaluate on the four benchmark families used by Apollo-MILP (Liu et al., 2025). Combinatorial Auctions (CA) instances follow the CATS generator (Leyton-Brown et al., 2000), Set Covering (SC) follows the classical construction of Balas & Ho (1980), and Workload Apportionment (WA) and Item Placement (IP) come from ML4CO (Gasse et al., 2022). CA is a maximization problem, whereas SC, WA, and IP are minimization problems. Table 5 reports average instance sizes. We use 240 training, 60 validation, and 100 testing instances, following the settings in Han et al. (2023) and Liu et al. (2025). We train all models with 500 epochs and a learning rate of 0.001.

Protocols for Combining EnCore with Downstream Search Methods

This section details how EnCore is integrated with the three downstream methods evaluated in Section 5. Following the evaluation protocol proposed by Han et al. (2023) and Liu et al. (2025), every run is given a total wall-clock budget of 1000s, and all overhead introduced by EnCore—including

Table 5: Statistical information of the benchmark instances.

	CA	SC	IP	WA
Constraint Number	2590.33	3000	195	64306
Variable Number	1500	5000	1083	61000
Binary Variables Number	1500	5000	1050	1000
Continuous Variables Number	0	0	33	60000
Integer Variables Number	0	0	0	0

the time spent collecting the early solution—is counted against this budget. Let t_{ES} denote the time used to obtain the early solution x^{ES} on a given instance.

Predict-and-Search and Neural Diving. Both methods consume the predictor output once, before the final solve, so the integration requires no change to their search procedures. Given an instance, we run the solver on the original MILP for t_{ES} seconds and record the early solution x^{ES} together with the associated solving-process features. The consistency predictor then assigns each binary variable a score $\bar{p}_i$, the estimated probability that x_i^{ES} agrees with the reference solution. Since both methods specify their fixing budgets separately for the two binary values, selection is value-conditioned: we rank the variables with $x_i^{ES} = 0$ by $\bar{p}_i$ and retain the top k_0, and independently retain the top k_1 among those with $x_i^{ES} = 1$. On CA under Predict-and-Search, for example, the setting $(k_0, k_1, \Delta) = (600, 0, 20)$ selects the 600 zero-valued variables with the highest consistency scores and no one-valued ones, and the resulting partial assignment $\{x_i = 0\}$ defines the center of a search neighborhood of radius $\Delta = 20$. Neural Diving consumes the selection in the same way except that the assignments are imposed as hard fixings and the reduced MILP is solved. In both cases the solver is restarted from scratch with a budget of $1000 - t_{ES}$ seconds, so that the end-to-end wall-clock time matches that of the baselines.

Apollo-MILP. Apollo-MILP performs four prediction–correction rounds with per-round budgets of 100s, 100s, 200s, and 600s. Extending the protocol above directly would require collecting a fresh early solution at the start of every round, which is too costly: the collection phase alone would consume a non-negligible fraction of the 100s rounds. We instead adopt the following assumption. Since Apollo-MILP realizes its fixing step by imposing bound constraints on the selected variables rather than by restructuring the problem, we assume that the early search behavior of the bounded problem stays close to that of the original instance, so that the early values realized on the original instance remain informative anchors throughout the iterative process. Under this assumption, we collect the early solution only once, on the original instance and in exactly the same way as above, and reuse x^{ES} and its consistency scores in the prediction step of every round. Each round applies the same value-conditioned selection, restricted to the variables not yet fixed in earlier rounds. To keep the total budget unchanged, the collection cost t_{ES} is charged to the final round, whose budget becomes $600 - t_{ES}$ seconds, while the first three rounds are left untouched; the overall wall-clock time thus remains 1000s.

Evaluation Protocol

Hardware. All experiments are conducted on a Linux server (Ubuntu 20.04.6 LTS) equipped with an Intel Xeon Platinum 8124M CPU (72 logical cores, 3.50GHz), 256GB of RAM, and an NVIDIA GeForce RTX 3090 GPU (24GB). Neural network training and inference run on the GPU, while all solver calls run on the CPU. The solver configuration (e.g., thread count) is kept identical across all methods on each benchmark.

Metrics. We report the final feasible objective and its absolute primal gap

$$\mathrm{gap}_{\mathrm{abs}} = |z - z_{\mathrm{BKS}}|, \tag{3}$$

where z is the mean final objective of a method and z_{BKS} is the best-known solution (BKS) on the corresponding benchmark. The BKS is defined as the best mean final objective observed among all evaluated methods, including a 3,600-second Gurobi run: Gurobi provides this reference on

SC, WA, and IP, while EnCore-PS provides it on CA. The BKS is thus an in-study best-observed reference rather than an independently certified optimum. To quantify the benefit of our method over a downstream baseline, we additionally report the relative gap reduction

$$\text{Improvement} = \frac{\text{gap}_{\text{base}} - \text{gap}_{\text{ours}}}{\text{gap}_{\text{base}}} \times 100\%. \tag{4}$$

ZERO-SHOT MIPLIB IIS TRANSFER PROTOCOL

The IIS setting traces to Wang et al. (2024), which formed a two-instance dataset from MIPLIB: `iis-glass-cov` was used for training and `iis-hc-cov` for testing. Apollo-MILP later expanded this setting into an eleven-instance subset of MIPLIB 2017 (Gleixner et al., 2021; Liu et al., 2025). Specifically, it selected instances according to similarities measured with 100 human-designed structural features, then discarded instances whose presolving exceeded 300 seconds or whose inference exceeded GPU memory. Apollo-MILP used eight of the resulting instances for training and three—`ramos3`, `scpj4scip`, and `scpl4`—for testing. In contrast, we use all eleven instances exclusively as a zero-shot test set. The instances range from 214 to 200,000 variables and from 32,805 to 2,000,000 nonzero coefficients.

In this experiment, both predictors are trained on the union of the CA, SC, WA, and IP training sets. They are validated on the union of the corresponding validation sets, and checkpoint selection uses only this combined validation set. The selected checkpoints are frozen before evaluation on IIS. No MIPLIB instance is used for training, validation, fine-tuning, or checkpoint selection. All methods, including the baselines, are evaluated under a 3,600-second end-to-end budget in this experiment, with the downstream solving budget adjusted to $3600 - t_{ES}$ seconds accordingly.

Table 6 gives the overall performance while Table 7 gives the instance-level results. The PS improvement is mainly attributable to `ramos3` and `ex1010-pi`, partially offset by `scpk4`. Apollo+Ours improves `ex1010-pi`, degrades on `ramos3`, and ties on the remaining instances.

Table 6: Zero-shot cross-family transfer to the MIPLIB IIS subset. Mean Obj is the mean of the per-instance final objectives. Feas. reports the number of instances with a finite feasible objective. † means computed only over its three feasible instances.

Predictor	Downstream	Mean Obj ↓	Feas.
Origin	Neural Diving	243.00†	3/11
EnCore	Neural Diving	173.45	**11/11**
Origin	Predict-and-Search	172.00	11/11
EnCore	Predict-and-Search	**171.73**	11/11
Origin	Apollo-MILP	172.91	11/11
EnCore	Apollo-MILP	172.82	11/11

Table 7: Final objectives on the eleven MIPLIB IIS instances. All instances are minimization problems. Bold marks the lowest objective in each row; a dash indicates that no finite feasible solution was found.

	ND		PS		Apollo-MILP	
Instance	GCN	Ours	GCN	Ours	GCN	Ours
`ex1010-pi`	–	242.00	237.00	**236.00**	241.00	238.00
`fast0507`	–	**174.00**	**174.00**	**174.00**	**174.00**	**174.00**
`glass-sc`	–	**23.00**	**23.00**	**23.00**	**23.00**	**23.00**
`iis-glass-cov`	–	**21.00**	**21.00**	**21.00**	**21.00**	**21.00**
`iis-hc-cov`	–	**17.00**	**17.00**	**17.00**	**17.00**	**17.00**
`ramos3`	–	235.00	229.00	**226.00**	231.00	233.00
`scpj4scip`	**132.00**	**132.00**	**132.00**	**132.00**	**132.00**	**132.00**
`scpk4`	328.00	330.00	**326.00**	327.00	330.00	330.00
`scpl4`	**269.00**	**269.00**	**269.00**	**269.00**	**269.00**	**269.00**
`seymour`	–	**423.00**	**423.00**	**423.00**	**423.00**	**423.00**
`v150d30-2hopcds`	–	42.00	**41.00**	**41.00**	**41.00**	**41.00**

E HYPERPARAMETER SETTINGS

EARLY SOLUTION COLLECTION

For early-solution collection, we set the sliding-window size to $w = 5$, the dual-gap decay threshold to $\epsilon = 0.01\%$/s, the minimum probing time to $T_{\min} = 20$ seconds, and the maximum probing time to $T_{\max} = 60$ seconds. We use an ensemble size of $K = 3$ for SC, CA and WA and use $K = 2$ for IP.

NEIGHBORHOOD SEARCH PARAMETERS

We report the hyper-parameters we used in our study. EnCore and baselines use the same hyper-parameters. Table 8 shows the hyperparameters of Predict-and-Search and EnCore-PS, and the Neural Diving and EnCore-ND uses the same (k_0, k_1). Table 9 shows the Apollo hyperparameters for baseline Apollo-MILP and EnCore-apollo.

Table 8: The partial solution size parameters (k_0, k_1) and neighborhood parameter Δ.

Benchmark	CA	SC	IP	WA
PS+Gurobi	$(600, 0, 20)$	$(2000, 0, 100)$	$(400, 5, 10)$	$(0, 500, 10)$
PS+SCIP	$(400, 0, 20)$	$(2000, 0, 100)$	$(400, 5, 1)$	$(0, 600, 5)$

Table 9: Hyperparameters $(k_0^{(i)}, k_1^{(i)}, \Delta^{(i)})$ for Apollo-MILP.

	CA	SC	IP	WA
Iteration 1	$(400, 0, 60)$	$(1000, 0, 200)$	$(100, 20, 50)$	$(20, 200, 100)$
Iteration 2	$(200, 0, 30)$	$(500, 0, 100)$	$(40, 15, 20)$	$(10, 100, 50)$
Iteration 3	$(100, 0, 15)$	$(250, 0, 50)$	$(20, 15, 10)$	$(10, 5, 5)$
Iteration 4	$(50, 0, 10)$	$(10, 0, 5)$	$(5, 50, 30)$	$(1, 10, 5)$

F DETAILS AND PROOFS FOR THE THEORETICAL ANALYSIS

POPULATION SETTING AND NOTATION

Let $\mathcal{D}$ be the probability distribution over all triples consisting of an MILP graph, the early solution produced by the fixed collection procedure, and the full-budget solution. Draw

$$Z = (G, X^{ES}, X^*) \sim \mathcal{D}.$$

Let $\mathcal{B}(G)$ be the set of binary-variable indices in G. Assume $|\mathcal{B}(G)| \geq 1$ almost surely. To represent the per-instance variable average compactly, conditional on Z, choose J from $\mathcal{B}(G)$, giving every index probability $1/|\mathcal{B}(G)|$. We use J for this random index in population quantities and r for a deterministic summation index.

Fix the message-passing depth L. For $r \in \mathcal{B}(G)$, let $\mathcal{R}_r$ denote the part of the static graph and its features within L message-passing steps of variable r, and let $X^{ES}_{\mathcal{R}_r}$ denote the early assignments attached to the variable nodes in this local input.

For any assignment predictor h, define

$$\begin{aligned} \ell_h(Z) &:= \frac{1}{|\mathcal{B}(G)|} \sum_{r \in \mathcal{B}(G)} \mathbf{1}\{h_r \neq x^*_r\}, \\ R(h) &:= \mathrm{E}[\ell_h(Z)] = \Pr(h_J \neq x^*_J), \\ \mathrm{Acc}(h) &:= 1 - R(h). \end{aligned}$$

Here a direct prediction h_r is a function of $\mathcal{R}_r$, whereas an induced consistency prediction may also use $X^{ES}_{\mathcal{R}_r}$. The corresponding posteriors are

$$\eta_G := \Pr(x^*_J = 1 \mid \mathcal{R}_J),$$
$$\eta_{ES} := \Pr(x^*_J = 1 \mid \mathcal{R}_J, X^{ES}_{\mathcal{R}_J}).$$

Their Bayes errors are

$$b_G := \mathrm{E}[\min\{\eta_G, 1-\eta_G\}],$$
$$b_{ES} := \mathrm{E}[\min\{\eta_{ES}, 1-\eta_{ES}\}].$$

Thus $A^*_{\mathrm{sol}} = 1 - b_G$ is the best population accuracy attainable from the static local input, and $1 - b_{ES}$ is the best assignment accuracy attainable when the early assignments are also observed.

CONSISTENCY–ASSIGNMENT EQUIVALENCE

Lemma 1 (label–assignment equivalence). For $r \in \mathcal{B}(G)$, let $y_r = \mathbf{1}\{x^{ES}_r = x^*_r\}$. Given a binary consistency prediction $\widehat{y}_r$, retain x^{ES}_r when $\widehat{y}_r = 1$ and use its binary complement otherwise. The induced assignment $\widehat{x}_r$ satisfies

$$\mathbf{1}\{\widehat{x}_r \neq x^*_r\} = \mathbf{1}\{\widehat{y}_r \neq y_r\}.$$

Equivalently, for a consistency rule a, define

$$h_a(G, X^{ES})_r := x^{ES}_r \oplus (1 - a(G, X^{ES})_r).$$

Then

$$\ell_{h_a}(Z) = \frac{1}{|\mathcal{B}(G)|} \sum_{r \in \mathcal{B}(G)} \mathbf{1}\{a(G, X^{ES})_r \neq y_r\}. \tag{5}$$

Proof. For binary values, $x^*_r = x^{ES}_r \oplus (1 - y_r)$. XOR by the same value preserves equality, so $h_a(G, X^{ES})_r \neq x^*_r$ exactly when $a(G, X^{ES})_r \neq y_r$. Averaging over $\mathcal{B}(G)$ proves Eq. (5). □

The induced assignment h_a is used only to compare variable-level prediction errors. In the downstream search method, assignments predicted inconsistent are left free rather than forcibly complemented, so these identities characterize the variable-level prediction task, while the downstream objective and runtime effects are evaluated empirically.

Conversely, every direct solution predictor h defines a consistency rule

$$a_h(G, X^{ES})_r := \mathbf{1}\{h(\mathcal{R}_r) = x^{ES}_r\}.$$

The induced predictor satisfies $h_{a_h}(G, X^{ES})_r = h(\mathcal{R}_r)$. Thus consistency prediction with the augmented input can reproduce every direct predictor based on the static local input.

Let $Y_J = \mathbf{1}\{x^{ES}_J = x^*_J\}$ and

$$\xi_J := \Pr(Y_J = 1 \mid \mathcal{R}_J, X^{ES}_{\mathcal{R}_J}).$$

For a fixed variable r, ξ_r denotes the corresponding conditional posterior evaluated at r. Since x^{ES}_J is observed in the augmented local input,

$$\xi_J = x^{ES}_J \eta_{ES} + (1 - x^{ES}_J)(1 - \eta_{ES}).$$

Consequently,

$$\min\{\xi_J, 1 - \xi_J\} = \min\{\eta_{ES}, 1 - \eta_{ES}\},$$

so the Bayes consistency error equals b_{ES}. Under variable-level 0–1 prediction loss, the optimal rule predicts 1 when $\xi_J \geq 1/2$ and 0 otherwise. By Lemma 1, its induced assignment accuracy is $A^*_{\mathrm{con}} = 1 - b_{ES}$. Because the label transformation is bijective after x^{ES}_J is observed, this is also the Bayes accuracy of a direct assignment predictor using the same augmented input. Theorem 1 therefore compares the information in the augmented input with that in the static input, rather than attributing a Bayes advantage to the consistency relabeling itself.

PROOF OF THEOREM 1

The population accuracy difference can be written as

$$A^*_{\mathrm{con}} - A^*_{\mathrm{sol}} = b_G - b_{ES} \geq 0. \tag{6}$$

The inequality is strict if there exists an event $\mathcal{E} \in \sigma(\mathcal{R}_J)$ with $\Pr(\mathcal{E}) > 0$ such that, almost surely on $\mathcal{E}$,

$$\Pr(\eta_{ES} < \tfrac{1}{2} \mid \mathcal{R}_J) > 0,$$
$$\Pr(\eta_{ES} > \tfrac{1}{2} \mid \mathcal{R}_J) > 0.$$

In words, the same static local input can be paired with early assignments that make either final value more likely.

Proof of Theorem 1. The tower property gives

$$\eta_G = \mathrm{E}[\eta_{ES} \mid \mathcal{R}_J]. \tag{7}$$

Using $\min\{t, 1-t\} = \frac{1}{2} - |t - \frac{1}{2}|$, the Bayes accuracy gap is

$$A^*_{\mathrm{con}} - A^*_{\mathrm{sol}} = \mathrm{E}[|\eta_{ES} - \tfrac{1}{2}|] - \mathrm{E}[|\eta_G - \tfrac{1}{2}|].$$

Conditional Jensen's inequality and Eq. (7) imply

$$\mathrm{E}[|\eta_{ES} - \tfrac{1}{2}| \mid \mathcal{R}_J] \geq |\eta_G - \tfrac{1}{2}|.$$

Taking expectations proves Eq. (6). If the refined posterior lies on both sides of $1/2$ with positive conditional probability, the conditional absolute-value inequality is strict. The stated positive-probability condition therefore gives $A^*_{\mathrm{con}} > A^*_{\mathrm{sol}}$. □

Scope of the comparison. The two optima compared in Theorem 1 range over functions of the same receptive field: A^*_{sol} is the Bayes accuracy given the static local input $\mathcal{R}_J$, and A^*_{con} is the Bayes accuracy given the augmented local input $(\mathcal{R}_J, X^{ES}_{\mathcal{R}_J})$, at the same fixed depth L. Statements in the main text about "any solution predictor", or about what "no predictor on instance features alone" can achieve, therefore quantify over predictors whose decision at a variable is a measurable function of its static local input, however large that function class is; they do not compare against predictors with a strictly larger input scope. In particular, if X^* were almost surely determined by the full graph G—for example, a unique optimal solution returned by a deterministic solver—then the whole-graph posterior would be $\{0,1\}$-valued, posterior crossing could not occur at the whole-graph level, and a whole-graph predictor would attain perfect accuracy in principle, leaving no information-theoretic gain for the early solution. The strict-gain content of Theorem 1 is thus per locality level: for every fixed message-passing depth, augmenting the local input strictly improves the best achievable accuracy whenever posterior crossing occurs at that depth. This is the operative comparison for the models considered in this line of work, including ours and the baselines, which are all message-passing GNNs whose receptive field is fixed by the architecture and is typically far smaller than the benchmark graphs. Moreover, on practical benchmarks X^* retains genuine randomness given G: instances frequently admit multiple optimal or near-optimal solutions, and the incumbent returned under a time budget depends on tie-breaking, thread timing, and budget truncation rather than on G alone. Under such residual randomness the whole-graph posterior is not $\{0,1\}$-valued either, so posterior crossing—and hence a strict gain from the early solution—can occur at any input scope, including the full graph.

SPARSE-CORRECTION IDENTITY

The Bayes consistency rule is

$$a^*(G, X^{ES})_r := \mathbf{1}\{\xi_r \geq \tfrac{1}{2}\}.$$

Define the event

$$\mathcal{C}^* := \{\xi_J < \tfrac{1}{2}\} = \{a^*(G, X^{ES})_J = 0\},$$

and define

$$p := \Pr(Y_J = 0), \qquad \rho := \Pr(\mathcal{C}^*).$$

If $\rho > 0$, also define

$$q := \Pr(Y_J = 0 \mid \mathcal{C}^*).$$

Then $q > 1/2$, and a^* satisfies

$$\begin{aligned} b_{ES} &= R(h_{a^*}) = p - \rho(2q-1), \\ \Delta_{\mathrm{B}} &:= A^*_{\mathrm{con}} - A^*_{\mathrm{sol}} = b_G - p + \rho(2q-1). \end{aligned} \tag{8}$$

Thus $\Delta_{\mathrm{B}} > 0$ exactly when

$$p < b_G + \rho(2q-1). \tag{9}$$

For $p \geq b_G$ and $\rho > 0$, this is equivalent to

$$q > \frac{1}{2} + \frac{p - b_G}{2\rho}.$$

If $\rho = 0$, then $b_{ES} = p$ and $\Delta_{\mathrm{B}} = b_G - p$.

Derivation. For any consistency rule a, retaining an assignment has error $1 - Y_J$, whereas using its binary complement has error Y_J. Hence

$$R(h_a) = p + \mathrm{E}[(2Y_J - 1)(1 - a(G, X^{ES})_J)]. \tag{10}$$

On $\mathcal{C}^*$, the conditional probability of $Y_J = 0$ is q. Substituting a^* into Eq. (10) gives

$$R(h_{a^*}) = p + \rho(1-q) - \rho q = p - \rho(2q-1).$$

Moreover,

$$q = 1 - \mathrm{E}[\xi_J \mid \mathcal{C}^*] > \tfrac{1}{2}.$$

The rule a^* is the pointwise Bayes classifier for Y_J, so Lemma 1 gives $R(h_{a^*}) = b_{ES}$. Subtracting this risk from b_G proves Eq. (8); the remaining statements follow by rearrangement. If $\rho = 0$, a^* retains the early solution almost surely, and its error is p. □

Finite-Sample Analysis

The population result ranges over all possible rules. For the finite-sample result, fix before observing the training instances a finite class $\mathcal{A}$ of $N := |\mathcal{A}| \geq 1$ consistency rules whose variable-level decisions use the augmented local inputs defined above. Let $Z_1, \ldots, Z_m$ be i.i.d. training instances from $\mathcal{D}$. Suppose training and test instances follow $\mathcal{D}$. Define

$$\begin{aligned} A^*_{\mathcal{A}} &:= \max_{a \in \mathcal{A}} \mathrm{Acc}(h_a), \\ \Delta_{\mathcal{A}} &:= A^*_{\mathcal{A}} - A^*_{\mathrm{sol}} = b_G - \min_{a \in \mathcal{A}} R(h_a). \end{aligned}$$

Recall $\Delta_{\mathrm{B}} := A^*_{\mathrm{con}} - A^*_{\mathrm{sol}}$, and define the finite-class approximation gap

$$\alpha_{\mathcal{A}} := A^*_{\mathrm{con}} - A^*_{\mathcal{A}}.$$

Then $\Delta_{\mathcal{A}} = \Delta_{\mathrm{B}} - \alpha_{\mathcal{A}}$, separating the population information gain from the approximation cost of the finite class. For an instance Z, let $s_a(Z)$ be the proportion of binary variables that rule a would flip. Assume every candidate flips at most a proportion $\bar{s} \in [0, 1]$ on average:

$$s_a(Z) := \frac{1}{|\mathcal{B}(G)|} \sum_{r \in \mathcal{B}(G)} (1 - a(G, X^{ES})_r),$$

$$\max_{a \in \mathcal{A}} \mathrm{E}[s_a(Z)] \leq \bar{s}.$$

Thus $\Delta_{\mathcal{A}}$ is the population accuracy difference between the best rule available in $\mathcal{A}$ and the best predictor based only on the static local input. It can be nonpositive if the finite class does not contain a sufficiently accurate consistency rule.

For a predictor that depends on the realized training sample and any training randomness, $R(h)$ and $\mathrm{Acc}(h)$ below denote conditional population quantities evaluated on a fresh test triple independent of that training information.

Let $a_{\mathrm{keep}}(G, X^{ES})_r \equiv 1$. Then $h_{a_{\mathrm{keep}}}(G, X^{ES}) = X^{ES}$ and $R(h_{a_{\mathrm{keep}}}) = p$. Define the loss change relative to this baseline by

$$D_a(Z) := \ell_{h_a}(Z) - \ell_{h_{a_{\mathrm{keep}}}}(Z),$$

$$\widehat{D}_a := \frac{1}{m} \sum_{k=1}^{m} D_a(Z_k).$$

Lemma 2 (uniform concentration). Following the finite-class Bernstein–union-bound argument (Boucheron et al., 2013), we apply concentration to instance-level loss differences relative to the always-retain rule. For every $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$\sup_{a \in \mathcal{A}} |\widehat{D}_a - \mathrm{E}[D_a]| \le \beta_{ES},$$
$$\beta_{ES} := \sqrt{\frac{2\bar{s}}{m} \ln \frac{2N}{\delta}} + \frac{4}{3m} \ln \frac{2N}{\delta}.$$

Proof. A rule differs from the always-retain baseline only on assignments that it marks inconsistent, so

$$|D_a(Z)| \le s_a(Z) \le 1,$$
$$\mathrm{Var}(D_a(Z)) \le \mathrm{E}[D_a(Z)^2] \le \mathrm{E}[s_a(Z)] \le \bar{s}.$$

Thus, when candidate corrections modify only a small proportion of early assignments, their loss differences have lower variance; this is the finite-class sparse-correction effect captured by $\bar{s}$. Also, $|D_a(Z) - \mathrm{E}[D_a]| \le 2$. (The sharper bound $|D_a(Z) - \mathrm{E}[D_a]| \le s_a(Z) + \bar{s} \le 1 + \bar{s}$ is available, since $|D_a(Z)| \le s_a(Z)$ and $|\mathrm{E}[D_a]| \le \bar{s}$; we retain the simpler constant 2, which only inflates the second-order term and keeps the penalty identical to the main-text statement.) To make the additive constant explicit, start from the two-sided Bernstein inequality in exponential form (Boucheron et al., 2013, Theorem 2.10): for a variance proxy σ^2 and centered range c,

$$\Pr\Big(|\widehat{D}_a - \mathrm{E}[D_a]| \ge \epsilon\Big) \le 2 \exp\left(-\frac{m\epsilon^2}{2\sigma^2 + 2c\epsilon/3}\right).$$

Bounding the right-hand side by $2e^{-t}$ gives the quadratic condition $m\epsilon^2 \ge 2\sigma^2 t + 2ct\epsilon/3$, whose positive root satisfies

$$\epsilon = \frac{ct}{3m} + \sqrt{\frac{2\sigma^2 t}{m} + \frac{c^2 t^2}{9m^2}} \le \sqrt{\frac{2\sigma^2 t}{m}} + \frac{2ct}{3m},$$

where the inequality uses $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$. Substituting the variance proxy $\sigma^2 = \bar{s}$ and the range $c = 2$ shows that the deviation $\sqrt{2\bar{s}t/m} + 4t/(3m)$ is sufficient; the additive form is conservative relative to the exponential form because of this square-root relaxation. That is, for every fixed a and $t > 0$,

$$\Pr\left(|\widehat{D}_a - \mathrm{E}[D_a]| > \sqrt{\frac{2\bar{s}t}{m}} + \frac{4t}{3m}\right) \le 2e^{-t}.$$

Setting $t = \ln(2N/\delta)$ and applying a union bound over $\mathcal{A}$ proves the claim. □

Theorem 2 (restated). Let $\widehat{a}$ minimize the empirical 0–1 risk over $\mathcal{A}$, and write $\widehat{h}_{ES} := h_{\widehat{a}}$. Let $\widehat{h}_{\mathrm{sol}}$ be any possibly data-dependent direct predictor determined without access to the fresh test triple and whose test-time decisions use only the static local inputs. Define

$$\varepsilon_m(\delta) := 2\sqrt{\frac{2\bar{s}}{m} \ln \frac{2N}{\delta}} + \frac{8}{3m} \ln \frac{2N}{\delta}.$$

With probability at least $1 - \delta$,

$$\mathrm{Acc}(\widehat{h}_{ES}) - \mathrm{Acc}(\widehat{h}_{\mathrm{sol}}) \ge \Delta_{\mathcal{A}} - \varepsilon_m(\delta) = \Delta_{\mathrm{B}} - \alpha_{\mathcal{A}} - \varepsilon_m(\delta).$$

On the same event,

$$R(\widehat{h}_{ES}) - \min_{a \in \mathcal{A}} R(h_a) \le \varepsilon_m(\delta).$$

Proof. Subtracting the same empirical loss of a_{keep} from every candidate preserves the empirical minimizer. On the event in Lemma 2, choose

$$a^*_{\mathcal{A}} \in \arg\min_{a \in \mathcal{A}} R(h_a).$$

Then

$$\mathrm{E}[D_{\widehat{a}}] \le \widehat{D}_{\widehat{a}} + \beta_{ES} \le \widehat{D}_{a^*_{\mathcal{A}}} + \beta_{ES}$$
$$\le \mathrm{E}[D_{a^*_{\mathcal{A}}}] + 2\beta_{ES}.$$

Because $R(h_a) = p + \mathrm{E}[D_a]$, it follows that

$$\begin{aligned} R(\widehat{h}_{ES}) &\le R(h_{a^*_{\mathcal{A}}}) + 2\beta_{ES} \\ &= 1 - A^*_{\mathcal{A}} + 2\beta_{ES} \\ &= b_G - \Delta_{\mathcal{A}} + 2\beta_{ES}. \end{aligned} \tag{11}$$

Condition on the realized training sample. Any resulting direct predictor whose test decision uses only the static local input has, by Bayes optimality,

$$R(\widehat{h}_{\mathrm{sol}}) \ge b_G. \tag{12}$$

Combining Eqs. (11) and (12) gives

$$\begin{aligned} \mathrm{Acc}(\widehat{h}_{ES}) - \mathrm{Acc}(\widehat{h}_{\mathrm{sol}}) &= R(\widehat{h}_{\mathrm{sol}}) - R(\widehat{h}_{ES}) \\ &\ge \Delta_{\mathcal{A}} - 2\beta_{ES}. \end{aligned}$$

Since $\varepsilon_m(\delta) = 2\beta_{ES}$, the theorem follows. □

A sufficient sample size. If $\Delta_{\mathcal{A}} > 0$ and

$$m > \max\left\{ \frac{32\bar{s}\ln(2N/\delta)}{\Delta^2_{\mathcal{A}}}, \frac{16\ln(2N/\delta)}{3\Delta_{\mathcal{A}}} \right\}, \tag{13}$$

then $\varepsilon_m(\delta) < \Delta_{\mathcal{A}}$. Consequently, with probability at least $1-\delta$, $\mathrm{Acc}(\widehat{h}_{ES}) > \mathrm{Acc}(\widehat{h}_{\mathrm{sol}})$.

Proof. Let $\Lambda := \ln(2N/\delta)$. The two bounds in Eq. (13) imply

$$2\sqrt{\frac{2\bar{s}\Lambda}{m}} < \frac{\Delta_{\mathcal{A}}}{2}, \qquad \frac{8\Lambda}{3m} < \frac{\Delta_{\mathcal{A}}}{2}.$$

Adding the inequalities gives $\varepsilon_m(\delta) < \Delta_{\mathcal{A}}$; Theorem 2 then yields the claim. □

Illustrative Calculations

Population gain. The sparse-correction condition can hold even when the early solution itself is less accurate than direct prediction. For example, take

$$p = 0.21, \qquad A^*_{\mathrm{sol}} = 0.82, \qquad \rho = 0.12, \qquad q = 0.70.$$

These numerical values are purely illustrative. The right-hand side of Eq. (9) is

$$1 - A^*_{\mathrm{sol}} + \rho(2q-1) = 0.18 + 0.12(0.40) = 0.228.$$

Thus $p = 0.21 < 0.228$. The early solution alone has accuracy 0.79, which is below the direct-prediction accuracy 0.82, whereas identifying the inconsistent assignments gives

$$A^*_{\mathrm{con}} = 1 - [0.21 - 0.12(0.40)] = 0.838.$$

The early-solution input therefore raises the accuracy from 0.82 to 0.838, an improvement of 1.8 percentage points.

Finite-sample bound. For a numerical evaluation of the bound, consider a fixed class of $N = 4$ correction rules constructed to satisfy $\mathrm{E}[s_a] \le 0.0180$. The value $\bar{s} = 0.0180$ uses the 1.80% median WA early-to-final difference in Figure 1(b) as an illustrative correction-budget scale, with $\mathrm{E}[s_a] \le \bar{s}$ retained as the stated rule-class assumption. Set the confidence level to $1 - \delta = 0.9$ and take a deliberately conservative illustrative class advantage $\Delta_{\mathcal{A}} = 0.12$. Here

$$\ln\frac{2N}{\delta} = \ln 80 \approx 4.382.$$

The two terms on the right-hand side of Eq. (13) are

$$\frac{32\bar{s}\ln(2N/\delta)}{\Delta^2_{\mathcal{A}}} \approx 175.3,$$
$$\frac{16\ln(2N/\delta)}{3\Delta_{\mathcal{A}}} \approx 194.8.$$

Thus the convenient sufficient condition in Eq. (13) would require $m = 195$. It is conservative because it separately bounds each of the two terms in ε_m by $\Delta_{\mathcal{A}}/2$. Directly solving $\varepsilon_m(0.1) < 0.12$, namely

$$2\sqrt{\frac{2(0.0180)}{m}\ln 80} + \frac{8}{3m}\ln 80 < 0.12,$$

gives $m > 188.19$, so the smallest integer sample size satisfying the exact inequality is $m = 189$.

Consistency requirement on the parameters. The illustrative values above are not jointly free. A rule that flips a proportion $s_a(Z)$ of assignments changes the induced assignment on at most that fraction of variables, so its accuracy exceeds the always-retain accuracy by at most the expected flip rate: $\mathrm{Acc}(h_a) \le (1-p) + \mathrm{E}[s_a] \le (1-p) + \bar{s}$. Consequently,

$$\Delta_{\mathcal{A}} = A^*_{\mathcal{A}} - A^*_{\mathrm{sol}} \le b_G - p + \bar{s},$$

and the choice $\bar{s} = 0.0180$, $\Delta_{\mathcal{A}} = 0.12$ implicitly requires $b_G - p \ge 0.102$: the Bayes error of static local prediction must exceed the early-solution error rate by roughly ten percentage points or more. On WA, where the 1.80% median early-to-final difference suggests $p \approx 0.018$, this amounts to $b_G \gtrsim 0.12$; that is, even the best predictor using only the static local input must mislabel at least about 12% of the binary variables. This is a substantive assumption about the difficulty of static prediction on the benchmark, not a consequence of the theorems, and we state it so that the example is read as a self-consistent operating point rather than a measured one. Smaller, still positive class advantages remain valid and simply rescale the sufficient sample size, which grows like $\Delta_{\mathcal{A}}^{-2}$ in Eq. (13).

Each MILP instance is one independent observation, so labels within an instance may be arbitrarily dependent. Accordingly, the penalty depends on m, the number of instances, rather than the total number of variable labels. The result assumes a fixed finite rule class selected by empirical variable-level 0–1 prediction loss. Its formal scope is finite-rule model selection, providing an abstraction that isolates instance-level estimation and the sparse-correction variance effect. In practice the predictor is a GNN trained by stochastic optimization with validation-based checkpoint selection, a procedure better described as comparing a small, data-dependent set of candidates—the checkpoints and configurations actually evaluated—than as exact empirical risk minimization over a class fixed in advance. Theorem 2 should therefore be read as an idealized account of that selection stage: it isolates the instance-level estimation cost and explains why sparse correction classes are cheap to select among, but it is not a uniform-convergence guarantee for the full GNN function class. Statements in the main text about the gain persisting under finite-sample model selection refer to this abstraction layer.

Relation to the deployed algorithm. Two design choices of the deployed system fall outside the formal statements above and are worth flagging explicitly. First, the analysis scores rules by variable-level 0–1 accuracy—equivalently, thresholding the posterior at $1/2$ over *all* binary variables—whereas the deployed method ranks variables by the predicted consistency probability and fixes only the top-ranked ones under a budget chosen by the downstream method. The theorems thus quantify the information available in the scores; how the scores are consumed by a ranking-based fixing rule is evaluated empirically. Second, the posteriors in this appendix condition on a single early solution, whereas at inference time the system averages, per variable, the logits computed from the last K improving solutions after aligning them to the reference solution X^{ES}. Because conditioning on the tuple of the last K solutions only enlarges the input, Theorem 1 applies verbatim to the ensembled variant, while the specific alignment-and-averaging estimator is a practical variance-reduction heuristic whose effect is evaluated empirically.