SINGLE-CONFIGURATION EVALUATION CAN MISLEAD LEARNED MILP CUT SELECTION: SEED FRAGILITY, MEASUREMENT-ERA ARTIFACTS, AND THE LIMITS OF SCORE BLENDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned components inside mixed-integer linear programming (MILP) solvers are often evaluated under a single experimental configuration, leaving the reliability of reported gains under solver and system variability unclear. We study this problem in the setting of learned cut selection in SCIP. Re-evaluating a frozen supervised cut scorer across six permutation seeds, we find substantial but sign-unstable effects on the search trajectory: only 2 of 31 test instances with complete six-seed data retain one benefit sign across all seeds, and pairwise sign agreement is 0.52. Two additional LightGBM cut scorers, trained with imitation and multiple-instance-learning targets, exhibit nearly identical sign instability (pairwise agreement 0.49 and 0.48, with no instance unanimous across all six seeds), ruling out our attribution-label recipe as the sole cause. Under a symmetric noise model, this disagreement places an approximately 60% upper bound on the fresh-seed sign accuracy of any seed-agnostic instance gate; the gates we test fail consistently with this ceiling. We uncover a second, orthogonal source of unreliability: an earlier 31/0 win–loss result against default SCIP was an artifact of a baseline measured under machine load. CPU-time accounting alone does not remove this confound, because memory pressure can inflate CPU cost per solver iteration; load logging and per-iteration rate probes expose it. After clean re-measurement, score blending is consistently less fragile than replacing SCIP's native score, but it does not produce robust downstream improvement: on benchmark-suite instances the intervention perturbs trajectories while about half of runs reach the same final primal and dual bounds as default, and on harder held-out instances the remaining positive structure is concentrated at intermediate difficulty and remains seed-dependent. These results motivate a two-axis evaluation protocol for learned solver components: vary solver symmetry-breaking conditions and verify the machine state under which paired comparisons are measured.

1 INTRODUCTION

A growing body of learning-for-optimization research trains MILP solvers to choose cutting planes. Yet many reported gains are measured under a single benchmark distribution, solver, time limit, random seed, and—although rarely stated—machine state. Because seemingly irrelevant perturbations can substantially redraw a MILP search tree (Lodi and Tramontani, 2014; Koch et al., 2011), a single run may be a poor basis for a comparative claim. We ask three questions. (RQ1) Are single-configuration evaluations of learned solver components reliable? (RQ2) If not, where does the unreliability come from? (RQ3) Can a simple intervention preserve whatever benefit is genuine?

Our starting point is an incident from our own work. We trained a supervised cut scorer on efficacy-weighted LP-gain attribution labels and embedded it in SCIP through a callback that replaces the native hybrid scores inside a top-30% prefilter at the root node. Under the usual discipline the result looked decisive: at 600s a gated learned policy beat default SCIP on 31 of 35 test instances with no losses ($p = 9.3 \times 10^{-10}$). We froze the configuration and re-ran the test set under three additional permutation seeds; the record collapsed to 14/17, 9/22, and 27/4. We later discovered that even

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.

this collapse understated the problem: after re-measuring the baseline in a load-instrumented clean window, the gated policy no longer outperformed default under any tested seed (10/21 even on its home seed), and the original 31/0 result traced to the baseline arm having been recorded under machine load.

The investigation yields three contributions. *First, we show that single-configuration evaluation can fail along two distinct axes.* The intervention has large per-instance trajectory effects (median $|\Delta| \approx 0.10$–0.14), but their signs are close to a coin flip: pairwise sign agreement is 0.52 across six seeds, and only 2 of the 31 complete-case instances (6.5%) retain one sign across all seeds. Under a symmetric noise model, even an oracle seed-agnostic instance gate has sign accuracy at most about 60% (Proposition 1), and every instance-level gate or rule we test fails consistently with this ceiling. The same sign instability recurs under two additional supervised scorer recipes, imitation and multiple-instance learning, showing that the phenomenon is not specific to our attribution target. Score blending reduces fragility but still fails to recover robust downstream benefit: gating fails for lack of identifiability, whereas blending fails for lack of downstream signal.

Second, we identify and instrument a separate measurement axis. CPU-time accounting protects against CPU preemption but not against per-iteration CPU inflation under memory pressure. On runs with matching node and LP-iteration counts, a baseline recorded under load has CPU times inflated by a median factor of $1.85\times$, creating spurious headline-sized improvements. A per-iteration rate probe combined with load logging detects this contamination at low cost.

Third, we map where any positive structure remains after clean re-measurement. On the benchmark-suite test stratum, the intervention changes trajectories without changing endpoints: at the time limit, 46–51% of (instance, seed) pairs (horizon-dependent) end with gap, dual bound, and primal bound identical to default (relative tolerance 10^{-6}). On harder held-out instances, a limited positive pattern remains: 11 of 19 active instances have positive mean effects, concentrated around intermediate difficulty, but two of six seeds reverse the aggregate sign: a narrow, low-magnitude regime rather than a robust scaling law.

2 Related Work

Learning to select cuts. Tang et al. (2020) frame cut selection as a contextual bandit trained with policy gradients. Paulus et al. (2022) imitate a strong lookahead expert by supervised scoring. Huang et al. (2022) learn a purely neural cut score by multiple-instance learning and take the top 30%, the canonical "learned score replaces native score" design, corresponding to our $\alpha = 1$ ablation. Wang et al. (2023) (HEM) model candidate ordering with a hierarchical sequence model trained by RL on homogeneous instance classes. Turner et al. (2023) (ACS) learn, per instance, convex weights over the four components of SCIP's native score, the closest relative of our blend, except that the learning happens *inside* the native score's components rather than as a convex bridge between a learned score and the native one. Balcan et al. (2021) give sample-complexity bounds for learning such weights. Adjacent tasks include stopping cut generation (Ling, Wang, and Wang, 2024), configuring separators (Wang et al., 2025), tree-global selection (Zeng et al., 2025), graph-based scorers (Zhang et al., 2024), and collaborative learning of interdependent solver modules (Li et al., 2025a).

Evaluation protocols, seeds, and solver variability. Table 1 summarizes protocols. Most cut-selection studies in Table 1 use a single solver seed, often SCIP's plugin tie-breaking shift (`randomseedshift`) rather than the permutation seed we vary; the few that use several seeds report only aggregate statistics. Nair et al. (2020) evaluate frozen learned diving and branching components across five permutation seeds, but likewise report only aggregates. Among the cut-selection studies we found, none reports per-seed outcomes or tests whether the sign of the benefit is stable. Solver performance variability under perturbation is well documented for *native* components: row/column permutations can substantially alter trajectories (Lodi and Tramontani, 2014), permuted copies are used in benchmarking (Koch et al., 2011), and randomization can be exploited to stabilize search (Fischetti and Monaci, 2014); worst-case analyses further show that tie-breaking-level differences can be exponentially amplified by branch-and-bound recursion (Cheng and Basu, 2026). Deza and Khalil (2023) explicitly caution that a learned agent "will try to learn optimal cut selection parameters for a specific solver environment," a caveat directly relevant to our study. Three closer relatives deserve distinction. Li et al. (2023) learn per-instance separator configurations, showing

Table 1: Evaluation protocols in related work. "Agg." = results reported only as aggregates over instance–seed pairs. "Era-verified" = arms measured inside a load-instrumented window with a documented machine-state check; "n.r." = not reported.

Work	Component	Solver	Seed parameter	#Seeds	Per-seed?	Era-verified?
Tang et al. 2020	RL cut policy	Gurobi env	none	1	—	n.r.
Huang et al. 2022	cut score (SBP)	SCIP env	none	1	—	n.r.
Paulus et al. 2022	imitation scorer	SCIP 7	none	1	—	n.r.
Wang et al. 2023 (HEM)	sequence model	SCIP 8	`scip_seed`	1	—	n.r.
Turner et al. 2023 (ACS)	native-score weights	SCIP 8	`randomseedshift`	3	Agg.	n.r.
Nair et al. 2020	diving + branching	SCIP	permute+seed	5	Agg.	n.r.
this work	cut score + blend	SCIP 10	permutationseed	6	**yes**	**yes**

that *which* configuration helps can be predictable across instances; our negative result concerns a different object—the *sign* of a frozen learned intervention's benefit across solver permutation seeds. Li et al. (2025b) report class-level sign variation, with some families improving and others slowing down; we resolve variability at the finer per-instance, per-seed level. The deep-RL community has documented training-seed variance extensively (Henderson et al., 2018; Agarwal et al., 2021); our seeds instead perturb the solver trajectory of a *frozen* learned model.

3 Setup and a Motivating Incident

We consider a MIP, $\min_x\{c^\top x : Ax \geq b,\ x \in \mathbb{R}^n \times \mathbb{Z}^p\}$, solved by branch-and-cut. A *separation round* offers candidate cuts $\mathcal{C}$, from which the selector commits $\mathcal{S} \subseteq \mathcal{C}$ to the relaxation. SCIP restarts periodically; each attempt is a *run*. We denote the solver's `randomization/permutationseed` by s. Let p_t and d_t be the global primal and dual bounds at CPU time t; the primal-dual integral $\mathrm{PD}_T = \int_0^T (p_t - d_t)\, dt$ summarizes the search trajectory (Bengio, Lodi, and Prouvost, 2021). For policy π against default policy d,

$$\mathrm{impr}_T(i, s, \pi) = \frac{\mathrm{PD}_T(i, s, d) - \mathrm{PD}_T(i, s, \pi)}{\mathrm{PD}_T(i, s, d)}, \tag{1}$$

which is positive when π is better. We compute the integral as a step sum over recorded bounds; missing initial bounds are backfilled with the first finite value. Runs that never find an incumbent are assigned $\mathrm{PD} = \infty$, excluded from finite-mean summaries, and counted separately. We integrate over CPU time by default; Section 8 shows that this is necessary but not sufficient, because memory pressure can inflate per-iteration CPU cost.

The incident. We trained the scorer of Section 6 on the train split (163 MIPLIB 2017 benchmark-suite instances), tuned an instance-level gate on validation (35), and evaluated once on test (35) at seed 42. At 600s the gated policy won 31, lost 0, and tied 4 (mean $+0.271$, $p = 9.3 \times 10^{-10}$). With the configuration frozen, we re-ran at seeds 0, 1, and 2. The 600s records were 31/0, 14/17, 9/22, and 27/4. The immediate lesson is that a single-seed record is not a reliable estimate of comparative performance.

The second autopsy. Months later, suspicious of an unrelated result, we rebuilt the entire comparison in a *clean window*: every arm re-run on the same idle-workstation regime with per-minute memory-pressure logging. In the clean window the gated policy's home-seed record at 600s is 10/21 (mean -0.007), and no seed wins. Tracing the discrepancy, we found the original default arm had been recorded under machine load: matching node and LP-iteration counts were associated with CPU times inflated by a median factor of $1.85\times$. The apparent 31/0 improvement was therefore an artifact of the baseline's measurement era. This reveals a second failure mode, orthogonal to seed choice: an unverified cross-era comparison cannot be assumed valid. Such comparisons may be sound on a dedicated machine under strict load control, but their validity must be demonstrated rather than presumed.

4 The Identifiability of Per-Instance Benefit

Write the normalized improvement of the learned policy on instance i under seed s as

$$\tau(i, s) = \mu(i) + \varepsilon(i, s), \qquad \mathbb{E}[\varepsilon \mid i] = 0,$$

where $\mu(i)$ is the instance's expected benefit and ε collects what the permutation stirs up. A gate predicts $\operatorname{sign}(\mu(i))$ from instance features and intervenes where the sign is positive. How well can any such gate do?

Proposition 1 (Oracle bound for sign prediction)**.** *Suppose that, conditional on i, $\varepsilon(i, s)$ is symmetric about 0 and independent across seeds, with $P(\tau(i, s) = 0 \mid i) = 0$ and $\mu(i) \neq 0$ for almost every instance. For each instance let A_i be the probability that two independent seeds give τ the same sign, and write $A = \mathbb{E}_i[A_i]$ for the population sign agreement. Then the oracle gate, which knows μ exactly, attains per-instance sign accuracy $p_i = (1 + \sqrt{2A_i - 1})/2$, its population accuracy obeys $P_{\text{oracle}} \leq (1 + \sqrt{2A - 1})/2$ with equality iff A_i is constant across instances, and no predictor measurable in instance features alone can exceed P_{oracle}. For $A \leq 0.58$ every such gate misclassifies at least 30% of fresh-seed instance outcomes.*

Proof. For an instance with $\mu(i) \neq 0$, write $p_i = P(\operatorname{sign}\tau(i, s) = \operatorname{sign}\mu(i))$; symmetry of ε gives $P(\text{disagree}) = 1 - p_i$. Two independent draws agree exactly when both match μ or both miss it, so $A_i = p_i^2 + (1-p_i)^2$. Solving and taking the root $p_i \geq 1/2$ gives $p_i = (1+\sqrt{2A_i - 1})/2$. The oracle attains accuracy p_i by using the true sign of $\mu(i)$; any predictor measurable in instance features alone has expected accuracy at most that of the oracle, which has direct access to $\mu(i)$. Averaging over instances, $P_{\text{oracle}} = \frac{1}{2}\big(1+\mathbb{E}_i[\sqrt{2A_i - 1}]\big) \leq \frac{1}{2}\big(1+\sqrt{2A - 1}\big)$ by Jensen's inequality applied to the concave map $a \mapsto \sqrt{2a - 1}$, with equality iff A_i is almost surely constant. The bound is increasing in A: it equals 0.50 at $A = 0.5$, 0.60 at $A = 0.52$, and 0.72 at $A = 0.6$. □

In the clean window (test split, six seeds), the per-instance benefit of the frozen learned policy has pairwise sign agreement $A = 0.52$ (range 0.29–0.74 across seed pairs) and only 6.5% of the 31 complete-case instances keep the same sign under all six seeds; benefit magnitudes are only weakly correlated across seeds (Pearson 0.11). Exact-zero effects occur in only 2 of 186 cells. Treating zero as a third sign yields the reported A; excluding seed pairs affected by an exact zero changes the estimate by less than 0.01. An earlier measurement on era-inflated baselines had suggested stronger magnitude structure ($r \approx 0.7$); that structure does not survive clean re-measurement. What *is* strongly instance-persistent is a different quantity: the root separation yield of an instance reproduces across solver restarts (Spearman 0.71–0.87), yet the yield does not predict the intervention's benefit ($\rho \approx 0$). Under the proposition's assumptions, these observations imply an upper bound of about 0.60 on fresh-seed sign accuracy for any seed-agnostic instance gate, and the observed failures are consistent with that ceiling: a classifier trained on two seeds' labels picks an anti-correlated subset on test; a static-feature classifier with train AUC ≈ 0.9 abstains on 31–33 of 35 instances; a rollback rule that trips on a flat first round confuses an ε-reading with a μ-reading and suspends learning exactly on instances whose later rounds would have helped.

One apparently exploitable structure did not survive clean re-measurement either. The positive-benefit rate appears to rise with instance size on era-inflated reads, but a one-line rule (intervene only when $n_{\text{vars}} \geq 14115$) scores -0.007 in the clean window, indistinguishable from zero; the trend does not extrapolate to larger instances. Under the symmetric-noise assumption, seed-agnostic instance-level predictors face a low accuracy ceiling (Proposition 1), and every signal of this kind we tested, whether features, gates, or rules, fails to carry the sign of this intervention's benefit. This does not rule out predictors that condition on the realized solver state, which observe the seed's trajectory directly; we return to this in Section 9.

5 Score Blending

Gating treats the learned model as an authority at the instance level. A softer alternative treats the learned and native scores as two noisy signals about cut utility. Model the learned score and SCIP's native hybrid score over a candidate pool as noisy estimates of a latent utility $r(c)$:

$$\hat{r}(c) = r(c) + \eta(c), \qquad q(c) = r(c) + \xi(c),$$

with zero-mean error processes η, ξ of variance $\sigma_{\text{ML}}^2, \sigma_{\text{H}}^2$ and correlation ρ. For a blend

$$B_\alpha(c) = \alpha\hat{r}(c) + (1 - \alpha)q(c) \tag{2}$$

after within-pool min-max normalization, the error variance is

$$V(\alpha) = \alpha^2\sigma_{\text{ML}}^2 + (1 - \alpha)^2\sigma_{\text{H}}^2 + 2\alpha(1 - \alpha)\,\rho\,\sigma_{\text{ML}}\sigma_{\text{H}}.$$

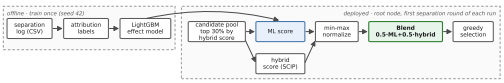


Figure 1: The selector. *Left* (offline, trained once): separation logs from a single incident seed become efficacy-weighted attribution labels for a LightGBM regressor. *Right* (deployed): at the first root separation round of each run, the top-30% pool by hybrid score is scored by the effect model and the hybrid formula, the two vectors are min-max normalized within the pool and blended at equal weight, and SCIP's greedy orthogonal selection commits the top-ranked cuts. All other nodes and rounds use the default selector.

Lemma 1. *(i)* $V(\alpha)$ *is convex. If* $\mathrm{Var}(\eta - \xi) > 0$*, its unconstrained minimizer is* $\alpha^* = (\sigma_{\mathrm{H}}^2 - \rho\sigma_{\mathrm{ML}}\sigma_{\mathrm{H}})/(\sigma_{\mathrm{ML}}^2 + \sigma_{\mathrm{H}}^2 - 2\rho\sigma_{\mathrm{ML}}\sigma_{\mathrm{H}})$*; over* $\alpha \in [0,1]$*, the minimizer is its projection onto* $[0,1]$*. If* $\mathrm{Var}(\eta - \xi) = 0$*,* V *is constant. When* $\rho = 0$*, the interior solution is the inverse-variance weight. (ii) At equal scales* $\sigma_{\mathrm{ML}} = \sigma_{\mathrm{H}} = \sigma$*,* $V(0.5) = \sigma^2(1+\rho)/2 \le \sigma^2$*, with equality iff* $\rho = 1$*. (iii) When* $\rho = 0$*,* $\alpha^* = 0.5$ *iff the two error scales are equal.*

Proof. (i) Differentiate and solve $V'(\alpha) = 0$; convexity follows from $V''(\alpha) = 2(\sigma_{\mathrm{ML}}^2 + \sigma_{\mathrm{H}}^2 - 2\rho\sigma_{\mathrm{ML}}\sigma_{\mathrm{H}}) = 2\,\mathrm{Var}(\eta - \xi) \ge 0$. Restricting α to $[0,1]$ clips the unconstrained minimizer to that interval. (ii) Substitute $\alpha = 0.5$. (iii) Setting $\alpha^* = 1/2$ at $\rho = 0$ gives $\sigma_{\mathrm{H}}^2 = \sigma_{\mathrm{ML}}^2$. □

The lemma therefore characterizes a convex ranking-noise curve: $V(1) = \sigma_{\mathrm{ML}}^2$ and $V(0) = \sigma_{\mathrm{H}}^2$, and an interior blend can reduce ranking noise below both endpoints when the error structure permits. This is only a ranking-level variance argument; it does not imply that minimizing $V(\alpha)$ minimizes downstream solver loss. Section 8 therefore uses the α-sweep to test whether ranking-level variance reduction translates into solver-level benefit.

Implementation. Figure 1 summarizes the selector. At each run's first root separation round, we prefilter candidates to the top 30% by hybrid score, score the survivors with both the learned effect model and the native hybrid formula, min-max normalize the two score vectors within the pool, blend them at $\alpha = 0.5$, and pass the resulting order to SCIP's greedy orthogonality selector. The deployed blend uses no gate, instance filter, or run restriction.

6 The Learned Scorer

The scorer is a LightGBM regressor (Ke et al., 2017) trained on efficacy-weighted LP-gain attribution labels. For each root separation round r with pre- and post-separation LP objectives o_r^{pre} and o_r^{post}, the round's relative improvement $\Delta_r = (o_r^{\mathrm{post}} - o_r^{\mathrm{pre}})/|o_r^{\mathrm{pre}}|$ is apportioned among the applied cuts in proportion to efficacy:

$$y_c = \begin{cases} \Delta_r \cdot \dfrac{\mathrm{efficacy}(c)}{\sum_{c' \in \mathcal{A}_r} \mathrm{efficacy}(c')}, & c \in \mathcal{A}_r, \\ 0, & c \notin \mathcal{A}_r, \end{cases}$$

where $\mathcal{A}_r$ is the set of cuts applied in round r; labels are winsorized at the 0.99 quantile. The target is an attribution signal derived from measured LP improvement, rather than an imitation label or end-to-end reward. Because it depends on LP-objective deltas rather than elapsed time, the training target is unaffected by the machine-state axis of Section 3. The training set contains 964k candidate rows from 114 training instances, collected at the first root separation round of each run. Per-cut features cover solver state, cut structure, coefficient statistics, SCIP built-in scores (efficacy, objective parallelism, cutoff distance, and integer support), and cut origin.

7 Experimental Setup

Instances. From the curated MIPLIB 2017 *benchmark suite* (Gleixner et al., 2021), we use 163 training, 35 validation, and 35 test instances, stratified by size and family. Relative to the collection split, we treat this suite as the easier end of the difficulty spectrum. A separate set of 36 MIPLIB

Policy	Limit	W/L vs. default	mean Δ	min Δ
Ungated top-k	300s	22/9	$+0.048$	-0.20
Ungated top-k	600s	8/23	-0.074	-0.54
Gated (thr=0.7)	300s	6/25	-0.045	-0.26
Gated (thr=0.9)	600s	10/21	-0.007	-0.25

Table 2: Frozen learned-only policies, test split, seed 42, *clean window*. W/L counts $\Delta > 0$ vs. $\Delta < 0$ (ties omitted). On era-inflated baselines the 600s gate had read 31/0 ($+0.271$); in the clean window the gated policy no longer has a positive aggregate result, even on its home seed: the earlier success was an artifact of the baseline's measurement era.

2017 *collection* instances (14k–120k variables, 30 families) was kept entirely outside training and data splitting.

Solver and hardware. We use SCIP 10.0.2 (Hojny et al., 2025) (SoPlex 8.0.2) through PySCIPOpt 6.2.1 (Maher et al., 2016). Each solver run is single-threaded and uses CPU-time accounting, with `permutationseed` $\in \{42, 0, 1, 2, 3, 4\}$ and horizons $\{60, 120, 300, 600\}$s. The 300s and 600s limits are primary; 60s and 120s are used only in the horizon-grid analysis. Experiments run on one workstation (10 cores, 16 GB RAM), with at most four independent solver runs executed concurrently.

Era discipline. All online paired comparisons used to support our current performance claims use arms re-measured inside one load-instrumented clean window, with per-minute memory-pressure logging and a per-iteration rate probe; CPU time alone does not provide this guarantee (Section 8.3).

Policies. We evaluate default hybrid selection; NoCuts; single-rule selectors (random, efficacy, and normalized violation); pure ML ($\alpha = 1$); blends with $\alpha \in \{0.25, 0.5, 0.75\}$; three benefit gates; a rollback circuit breaker; and a size-threshold rule. The gates and rollback are included as attempted repair levers rather than proposed deployment policies.

Protocol and statistics. Configurations are frozen before test evaluation. We report per-seed results, pair the arms by instance, seed, and budget, use CPU time, and report lower-tail summaries (P10, minimum) alongside means. Primary evaluation: test $\times$ 6 seeds $\times$ {300,600}s, over four thousand solver runs. Tests: Wilcoxon signed-rank and sign tests on paired per-instance improvements.

8 Results

We organize the results around the three questions of the Introduction, in order. RQ1: are single-configuration evaluations of learned solver components reliable? RQ2: where does the unreliability come from? RQ3: can a simple intervention preserve whatever benefit is genuine? All test-set numbers pair the policy arm with the default arm on the same instances, seeds, and CPU-time budget; improvement Δ is the ratio of Eq. (1).

8.1 RQ1: Are single-configuration evaluations reliable?

Table 2 evaluates four frozen learned-only configurations from Section 3: the scorer trained on the seed-42 incident, gates with thresholds tuned on validation, all evaluated on the test split at seed 42.

In era-inflated form, this table told a tempting story: gating appeared to convert coin flips into wins, spectacularly so at 600 seconds (then 31/0, Wilcoxon $p < 10^{-9}$). In the clean window the contrast is gone: neither gated operating point has a positive aggregate result, and the ungated policy's 300-second seed-42 win (22/9, $+0.048$) is exactly the kind of reading that does not survive the next seed (Section 8.2).

8.2 RQ2: Where does the unreliability come from?

We rerun the frozen policies on the test split under six seeds $\{42, 0, 1, 2, 3, 4\}$ and audit four repair levers.

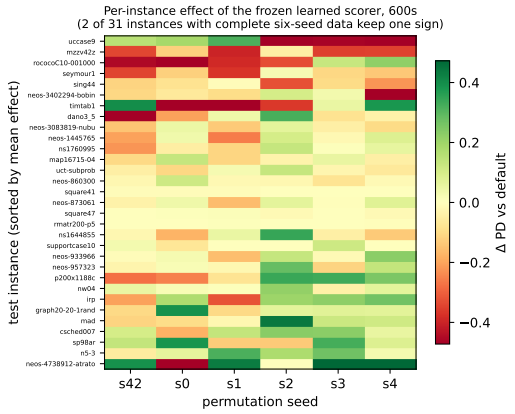


Figure 2: Per-instance effect of the frozen learned scorer at 600s in the clean window (test split, six seeds; color: Δ PD vs. default; rows sorted by mean effect). Only $2/31$ instances keep one sign across all six seeds: the sign of the benefit is close to a coin flip per (instance, seed) cell.

The effect does not transfer across seeds. Aggregated over seeds, the learned scorer's per-instance effect changes sign too often to be exploitable: pairwise sign agreement is 0.52 and only 6.5% of the 31 complete-case instances keep the same sign under all six seeds, so a gate fit on one seed's wins carries no transferable sign information to another seed. The trajectories' *yield* is strongly instance-persistent across restarts (Spearman 0.71–0.87 on first-round root LP gains), but the *attribution* of improvement to our intervention is not: the default arm's path shifts by a comparable amount. (Era-inflated baselines had shown the same sign fragility at 0.52–0.58.)

Not one bad scorer. The fragility is not unique to our label design. In the same clean window, pure deployments of two further scorer recipes (imitation of applied-cut indicators; multiple-instance learning) show the same pattern at 600s: pairwise sign agreement 0.49 and 0.48 (ours 0.52), and not a single instance keeping one sign across all six seeds (ours 6.5%). Thus the observed fragility is not specific to our attribution-label scorer: it recurs across all three supervised LightGBM cut scorers tested under the same SCIP deployment protocol (Table 3).

Lever 1: coarse instance rules. The best threshold rule found on the training split (intervene iff $\text{vars} \geq 14{,}115$) scores mean $\Delta = -0.007$ at 600 seconds in the clean window, indistinguishable from zero. Coarse rules inherit the same fragility at lower resolution.

Lever 2: narrower intervention. Restricting the intervention to the first separation round of the first run (rather than the first round of every run) does not recover positive performance: per-seed mean Δ at 600 seconds ranges from -0.02 to -0.12 across the six seeds. Narrowing the intervention reduces variance but does not create downstream signal.

Lever 3: shorter horizons. Mean Δ of the ungated policy across the four registered time limits $\{60, 120, 300, 600\}$s is $\{-0.052, -0.016, +0.025, -0.023\}$ (six seeds): no time limit turns the learned-only intervention reliably positive. The only positive cell (300s, $+0.025$) is small, is not corroborated by gap or dual-bound metrics, and vanishes at 600s.

Lever 4: blend with the native score. Blending (Eq. (2), α=0.5) is the only lever that survived the seed audit on era-inflated baselines, which is why the next subsection subjects it to the clean window. There, the blend is the *least fragile* configuration, and on the benchmark stratum that is all it is.

8.3 RQ3: Can a simple intervention preserve the benefit?

The blend does not survive clean re-evaluation on the benchmark stratum. Against era-inflated baselines, blending at α=0.5 had been the only uniformly positive configuration (pooled

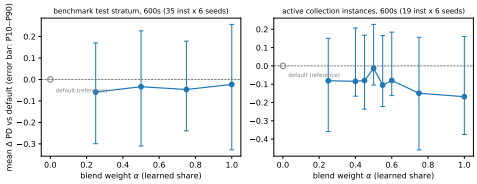


Figure 3: Dose–response of the blend weight α at the 600s horizon in the clean window (mean Δ PD vs. default; error bars P10–P90). The open circle at α=0 is the default arm against itself, the zero reference. *Left*: benchmark test stratum. *Right*: the 19 active collection instances. Every nonzero α has a negative mean effect relative to default; α=0.5 has the smallest degradation and α=1 the largest.

$+0.33$/$+0.31$, winning on every seed). In the clean window, the same frozen configuration yields $+0.028$ at 300s and -0.034 at 600s, neither statistically distinguishable from zero; head-to-head against the pure learned scorer, its win rate is 0.46. The empirical α-sweep is shallow: across $\alpha \in \{0.25, 0.5, 0.75, 1.0\}$, no configuration separates from default on the benchmark stratum (Figure 3). Thus blending reduces fragility, but where default SCIP already closes the problem it has little downstream headroom to recover.

Churn without destination change. The intervention does move the search: the median $|\Delta|$ per (instance, seed) pair is 0.10–0.14 at 600s. But at the time limit, 46–51% of pairs (horizon-dependent) end with gap, dual bound, and primal bound *identical* to default (relative tolerance 10^{-6}). On the benchmark stratum, learned cut selection reshuffles the path, not the destination. This reframes RQ1–RQ2: the large per-instance effects measured on PD are real trajectory effects, but they do not translate into endpoint improvements at the evaluated horizons.

The second axis: measurement-era artifacts. The failure mode behind our invalidated headline is distinct from seed variation. The original default arm had been recorded under heavy machine load. On runs with matching node and LP-iteration counts, its measured CPU times are inflated by a median factor of $1.85\times$ relative to the same runs on an idle machine. Two effects contribute: (i) the oldest runs predate CPU-time accounting, so queueing time entered the measured trajectory directly; (ii) under memory pressure the same work consumes more CPU seconds (paging and cache effects are charged to the process). Because PD integrals and time-to-target metrics integrate over the measured time axis, this throughput difference biases comparisons in favor of policies measured against the slower baseline. We therefore use two inexpensive diagnostics (Appendix A): per-minute memory-pressure logging and a per-iteration rate probe (LP iterations per CPU second). In our data, contaminated windows appear near $0.5\times$ of the clean reference rate, whereas verified clean windows lie at $0.9\times$–$1.0\times$. Every online paired comparison used to support our performance claims comes from such a verified window.

Where the benefit appears, and how it breaks: intermediate difficulty. On the 36-instance held-out collection split, re-measured in the clean window, the blend's effect on the 19 instances with an active intervention surface remains strongly seed-dependent. The first four seeds have positive aggregate means (pooled mean $+0.072$), whereas two additional seeds reverse the sign (-0.27 and -0.10), yielding a six-seed mean of approximately zero (median $+0.0003$); 11 of 19 instances have positive mean effects. A limited pattern remains around moderate difficulty (Spearman -0.22 against baseline PD scale within the set), where the blend is also least fragile. The 19 are exactly the instances where both arms have finite PD integrals. Across all 36 instances under inf-aware accounting, the record is 67 wins, 55 losses, and 87 ties; seven further pairs are unclassifiable (no usable trajectory in either arm, or a zero baseline integral). Among runs without a finite PD integral, the blend improves the dual bound by $+0.085$ on average. At the hardest end the root separation loop often never finishes, leaving no cut-selection opportunity (44% of the collection split produces no candidate cuts to select). We therefore find no regime with robust destination-level improvement: the apparent sweet spot is a region of reduced harm rather than reliable gain.

9 DISCUSSION

9.1 A CHECKLIST FOR EVALUATING LEARNED SOLVER COMPONENTS

The incident of Section 3 was caused not by an implementation bug but by an evaluation protocol that stopped one seed too early. We distill the lessons into five low-cost requirements that would have exposed the problem before submission:

1. **Freeze before test evaluation.** Fix the scorer, gate, thresholds, and blend weight using training and validation data before any test runs; ours (0.7/0.9, $\alpha = 0.5$) were never retuned on the test split.
2. **Vary the symmetry-breaking seed; report per-seed outcomes.** A component trained on one seed's trajectory has no reason to be seed-robust, and aggregate statistics can hide sign reversals. Our apparent seed-42 success did not survive the six-seed audit.
3. **Pair the arms on instance, seed, budget, and era.** Unpaired comparisons confound the effect with solver variability (Lodi and Tramontani, 2014); unverified cross-era comparisons confound it with machine state, and CPU-time accounting alone does not prevent the latter. Measure paired arms under the same verified load regime, using load-instrumented windows and a per-iteration rate probe (Appendix A).
4. **Report tails, not only means.** Alongside mean effects, report per-seed win/loss and lower-tail summaries such as minima or quantiles; we use minima and P10/P90 where applicable.
5. **Preserve the audit trail.** Failed gates, invalidated baselines, and the diagnostics that exposed them are part of the scientific record.

9.2 LIMITATIONS

Instance distribution. The main splits are the MIPLIB 2017 benchmark suite (35 test instances), where we find no robust destination-level improvement; the limited positive structure we observe is confined to collection instances of intermediate difficulty (Section 8.3), with the sweet spot's edges resting on 19 active instances.

Scope. All experiments use SCIP 10 on one workstation, so the magnitudes of seed fragility and machine-state effects may be solver- and hardware-dependent. Proposition 1 assumes symmetric conditional noise and independent seeds; Lemma 1 is only a ranking-level variance argument. Robustness claims are further scoped to permutation-seed perturbations. Three scorer recipes (attribution, imitation, and multiple-instance learning) all exhibit the fragility (Section 8.2), but all three are LightGBM scorers deployed in SCIP; we do not extrapolate beyond that family. Finally, α is a single global scalar. Choosing α from static instance features would reintroduce the identifiability problem of Section 4; adaptation conditioned on the realized solver state remains an open direction.

10 CONCLUSION

We set out to improve learned cut selection in MILP and instead found two ways in which its evaluation could mislead. First, the per-instance benefit of a frozen scorer changes sign across solver permutation seeds often enough that static instance-level gates cannot reliably recover it; narrowing the intervention or changing the time horizon does not restore robust benefit. The same instability appears across three supervised cut-scoring recipes, so the effect is not attributable to a single label design. Blending with SCIP's native score reduces fragility but likewise fails to produce robust destination-level improvement on the benchmark stratum. Second, our original headline result was an artifact of a baseline measured under machine load, exposed only after load-instrumented remeasurement and a per-iteration rate probe. On harder held-out instances the blend is positive on 11 of 19 active cases but remains strongly seed-dependent. Learned components inside exact solvers inherit the solver's symmetry and the machine's state; credible evaluation varies the former and verifies the latter.

AI USE STATEMENT

Generative-AI coding assistants (Kimi Code; CodeBuddy) were used in this work to implement the SCIP callbacks and the benchmark pipeline, to create and edit software code, to give feedback on experimental methodology (including the multi-seed protocol), to orchestrate benchmark batches, to interpret result tables jointly with the authors, and to draft and edit parts of this manuscript. The research questions, the identifiability and blending arguments, and all design decisions were developed in author-led discussion with these tools. No synthetic data was generated, and no reported number was produced by a generative model: every aggregate in this paper is recomputed from the logged per-run artifacts by deterministic scripts, and all AI-assisted code and text was reviewed by the authors. The authors take full responsibility for the content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All experiments use SCIP 10.0.2 with CPU-time accounting, fixed permutation seeds, and configurations frozen before test evaluation and never re-tuned on the test split; all online paired comparisons used to support our performance claims come from arms measured inside a single load-instrumented clean window (per-minute memory-pressure logs and per-iteration rate probes, Appendix A). The instance lists, splits, training and evaluation pipeline, and the blending implementation are described in Sections 6–8 and the appendices, where every aggregate can be traced to per-run logged artifacts. The callback implementation, benchmark driver, trained scorers, evaluation scripts, and the audit-trail instruments are available to reviewers as a supplementary code archive and through an anonymized repository[1] and will be released publicly upon publication.

REFERENCES

Agarwal, R.; Schwarzer, M.; Castro, P. S.; Courville, A.; and Bellemare, M. 2021. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*.

Balcan, M.-F.; Prasad, S.; Sandholm, T.; and Vitercik, E. 2021. Sample complexity of tree search configuration: cutting planes and beyond. In *Advances in Neural Information Processing Systems 34*, 4015–4027.

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

Cheng, H.; and Basu, A. 2026. Theoretical challenges in learning for branch-and-cut. *arXiv:2601.23249*.

Deza, A.; and Khalil, E. 2023. Machine learning for cutting planes in integer programming: a survey. In *Proc. 32nd Int. Joint Conf. Artificial Intelligence*, 6592–6600.

Fischetti, M.; and Monaci, M. 2014. Exploiting erraticism in search. *Operations Research*, 62(1): 114–122.

Gleixner, A.; Hendel, G.; Gamrath, G.; Achterberg, T.; Bastubbe, M.; Berthold, T.; Christophel, P.; Jarck, K.; Koch, T.; Linderoth, J.; Lübbecke, M.; Mittelmann, H.; Ozyurt, D.; Ralphs, T.; Salvagnin, D.; and Shinano, Y. 2021. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3): 443–490.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep reinforcement learning that matters. In *AAAI*.

Hojny, C.; et al. 2025. The SCIP Optimization Suite 10.0. *Optimization Online*, November 2025.

Huang, Z.; Wang, K.; Liu, F.; Zhen, H.-L.; Zhang, W.; Yuan, M.; Hao, J.; Yu, Y.; and Wang, J. 2022. Learning to select cuts for efficient mixed-integer programming. *Pattern Recognition*, 123: 108353.

[1] https://anonymous.4open.science/r/scip-blend-code-06E6/

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. LightGBM: a highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, 3146–3154.

Koch, T.; Achterberg, T.; Andersen, E.; Bastert, O.; Berthold, T.; Bixby, R.; Danna, E.; Gamrath, G.; Gleixner, A.; Heinz, S.; Lodi, A.; Mittelmann, H.; Ralphs, T.; Salvagnin, D.; Steffy, D.; and Wolter, K. 2011. MIPLIB 2010. *Mathematical Programming Computation*, 3(2): 103–163.

Li, S.; Ouyang, W.; Paulus, M. B.; and Wu, C. 2023. Learning to configure separators in branch-and-cut. In *NeurIPS*.

Li, S.; Yu, Y.; Zhang, Z.; Chen, M.; Zhu, F.; Zhong, T.; Liu, P.; and Hao, J. 2025a. Collab-Solver: collaborative policy learning for mixed-integer linear programming. *arXiv:2508.03030*.

Li, S.; Kulkarni, J.; Menache, I.; Wu, C.; and Li, B. 2025b. Towards foundation models for mixed integer linear programming. In *ICLR*.

Ling, H.; Wang, Z.; and Wang, J. 2024. Learning to stop cut generation for efficient mixed-integer linear programming. *Proc. AAAI Conf. Artificial Intelligence*, 38(18): 20759–20767.

Lodi, A.; and Tramontani, A. 2014. Performance variability in mixed-integer programming. In *Theory Driven by Influential Applications*, INFORMS TutORials in Operations Research, 1–12.

Maher, S.; Miltenberger, M.; Pedroso, J. P.; Rehfeldt, D.; Schwarz, R.; and Serrano, F. 2016. PySCIPOpt: mathematical programming in Python with the SCIP Optimization Suite. In *Mathematical Software – ICMS 2016*, LNCS 9725, 301–307.

Nair, V.; Bartunov, S.; Gimeno, F.; von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. 2020. Solving mixed integer programs using neural networks. *arXiv:2012.13349*.

Paulus, M. B.; Zarpellon, G.; Krause, A.; Charlin, L.; and Maddison, C. 2022. Learning to cut by looking ahead: cutting plane selection via imitation learning. In *Proc. 39th Int. Conf. Machine Learning*, PMLR 162, 17584–17600.

Tang, Y.; Agrawal, S.; and Faenza, Y. 2020. Reinforcement learning for integer programming: learning to cut. In *Proc. 37th Int. Conf. Machine Learning*, PMLR 119, 9367–9376.

Turner, M.; Koch, T.; Serrano, F.; and Winkler, M. 2023. Adaptive cut selection in mixed-integer linear programming. *Open Journal of Mathematical Optimization*, 4: 5.

Wang, Z.; Li, X.; Wang, J.; Kuang, Y.; Yuan, M.; Zeng, J.; Zhang, Y.; and Wu, F. 2023. Learning cut selection for mixed-integer linear programming via hierarchical sequence model. In *The Eleventh International Conference on Learning Representations*.

Wang, J.; Ye, M.; Kuang, Y.; et al. 2025. Dynamic configuration of cutting plane separators via reinforcement learning on incremental graph. In *Advances in Neural Information Processing Systems 38*.

Zeng, S.; Zhang, S.; Li, S.; Wu, F.; and Li, X.-Y. 2025. Beyond local selection: global cut selection for enhanced mixed-integer programming. *arXiv:2503.15847*.

Zhang, X.; Chen, L.; Yang, Z.; and Zeng, Z. 2024. Learning to select cutting planes in mixed integer linear programming solving. *arXiv:2410.03112*.

A Measurement hygiene: instruments and audit trail

Load logging. During every benchmark window we log the workstation's memory pressure once per minute (macOS `memory_pressure`). A window is accepted when free memory stays above 25% throughout; our clean windows read 31–32% minimum.

Rate probe. With solver configuration, instance, and permutation seed fixed, the runs used by our rate probe match in node and LP-iteration counts. We therefore compare *LP iterations per*

CPU second across windows on the same instance; a window whose ratio median is $\approx 0.5\times$ a clean window's is contaminated (our July baseline arm reads $0.50\times$ [0.42, 0.64] against the clean window), while clean arms read $0.85\times$–$1.0\times$. The probe needs no re-runs: it reads the existing summaries.

Audit trail. The era artifact was caught when a set of unrelated interventions showed suspiciously uniform gains against the old baseline arm; the same nodes-and-iterations trajectory read 75.3s versus 18.9s across windows. Every online paired comparison used in our performance claims was then re-measured with both arms in the same verified window.

B ADDITIONAL ABLATION DETAILS

Normalization. Offline ranking quality is insensitive to how the two score vectors are normalized within the pool: min-max (deployed), rank-uniform, and z-score variants differ by at most 0.017 in mean NDCG@30% and 0.003 in precision@30% against realized effect on validation pools. Thus, normalization choice has little effect on these offline ranking metrics.

Archived ablations. Two ablation blocks from an earlier draft were measured against the era-inflated baseline arm of Section 8.3: a label ablation (attribution vs. imitation vs. multiple-instance-learning scorers in the blending slot) and 1800s time-to-optimality experiments on the collection split. Because those comparisons used the contaminated baseline, we exclude them from the present claims. The normalization ablation above is retained because it is an offline, time-invariant measurement; the mechanism measurements below and the offline columns of Table 3 are likewise offline and time-invariant, while the online sign columns of Table 3 and the external evaluation in Section 8.3 use clean-window runs.

Scorer	Training target	Offline NDCG@30%	Sign agreement	Unanimous
Attribution (ours)	LP-gain attribution	0.72	0.52	6.5%
Imitation	applied-cut indicator	0.67	0.49	0%
MIL	multiple-instance target	0.53	0.48	0%
Native hybrid (reference)	—	0.78	—	—

Table 3: Three supervised LightGBM cut scorers under the same deployment protocol. Online sign statistics use pure deployment (α=1) at 600s in the clean window; offline NDCG@30% is measured against realized attribution labels on 1,018 validation pools. Because the reference labels are attribution-based, NDCG is not a target-neutral measure of scorer quality. The sign columns follow Section 8.2 (pairwise agreement across six seeds; share of instances unanimous across all six). Offline ranking quality varies substantially across the three learned scorers, whereas online sign agreement remains near 0.5.

Relation to published numbers. The protocols of prior work are not commensurable with ours, so we do not tabulate against them in the main text: HEM reports 30–41% over NoCuts on homogeneous classes with ∼10k training instances each; its SBP baseline (our $\alpha = 1$ family) reports 7.7% on hard MIPLIB; ACS reports 1.75% over SCIP default on heterogeneous MIPLIB with `randomseedshift` variation. On the benchmark stratum we measure no destination effect at all; on held-out collection instances of intermediate difficulty we observe a limited positive pattern (11/19 instances positive, mean ≈ 0 across six seeds).

Mechanism. The mechanism measurements are offline and time-invariant, so they stand regardless of the era issue: the learned ranking disagrees with the native one (Spearman 0.27 at 300s) yet agrees on the head (34% top-10 overlap), and the attribution labels are dual-dominated (85%/15% at 300s; 81% dual share at 600s). The ranking-level variance argument of Section 5 therefore survives, but the benchmark stratum provides little downstream utility for that variance reduction to recover.