Evaluating LLM-Generated Solver Configurations under Controlled Search Budgets

Anonymous authors
Paper under double-blind review

Abstract

Large language model (LLM) workflows can generate solver configurations that improve optimization performance. A strong configuration establishes artifact quality, while attribution to the proposal mechanism requires comparison with alternative search under a common evaluation procedure and budget. We introduce PACER (**P**roposer **A**blation for **C**onfiguration **E**valuation under Matched **R**esources), a protocol that fixes scoring, validation selection, held-out testing, and solver-call accounting while varying the configuration-search method and validation budget. On the easy generalized independent set problem (GISP-easy), a remeasured historical LLM-generated IBM ILOG CPLEX configuration improves the held-out primal-dual integral (PDI) by 34.3%. An independent eight-trial snapshot yields 2 of 15 conventional-search endpoints above this reference. Along the nested searches, the count increases from 5 of 15 after 25 trials to 11 of 15 after 60 trials. Replays of six configurations retained from the historical workflow across three problem families improve both PDI and wall-clock time in every point estimate. An ablation of the easy combinatorial-auction artifact shows that fixed CPLEX settings account for nearly all of its measured PDI gain. These results show how search opportunity changes the interpretation of an LLM-generated solver configuration.

1 Introduction

Mixed-integer linear programs (MILPs) arise in robustness verification, discrete black-box optimization, and decision-focused learning (Tjeng et al., 2019; Papalexopoulos et al., 2022; Jeong et al., 2022). Modern MILP solvers expose many parameters controlling presolve, cutting planes, branching, and primal heuristics (Bixby, 2012; Achterberg et al., 2020). For a recurring workload, these parameters can be fixed as a global solver configuration and selected from validation performance under a finite evaluation budget (Hutter et al., 2009; 2011; Snoek et al., 2012; Bergstra & Bengio, 2012; Schede et al., 2022). Large language model (LLM) agents can generate such configurations from problem descriptions, code, and solver feedback (Romera-Paredes et al., 2024; Liu et al., 2024; Yang et al., 2024; Lawless et al., 2025; Luo et al., 2026). Improvement over the solver default establishes the quality of the selected configuration. Evaluating the proposal mechanism also requires measuring how readily alternative search methods reach comparable performance under the same budget.

This distinction raises the central question of the paper: *how should an LLM-generated solver configuration be evaluated relative to conventional configuration search under a controlled budget?* We introduce PACER (Proposer Ablation for Configuration Evaluation under Matched Resources), a protocol that operationalizes the artifact-versus-proposer distinction for LLM-generated optimization artifacts. PACER fixes the workload, scoring rule, validation procedure, held-out test set, and solver settings while varying the configuration-search method and its evaluation budget. Each run selects its best configuration on validation data and evaluates the selected configuration once on the shared held-out set. PACER counts the resulting solver calls directly, making search effort comparable across methods (Hutter et al., 2010b; Eggensperger et al., 2019). We write ρ for the fixed evaluator, π

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

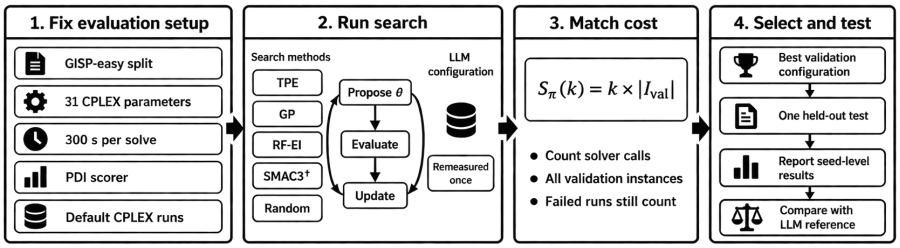

Figure 1: Overview of PACER. PACER fixes the evaluator and paired primal-dual integral score, runs conventional search methods under the same evaluator and solver-call budgets, and evaluates the best completed validation configuration once on the held-out test set. The historical LLM-generated configuration is remeasured under the common evaluator and used as the reference.

for the configuration-search method, and S for the billed CPLEX calls. Figure 1 summarizes this comparison.

We evaluate PACER with IBM ILOG CPLEX (Bixby, 2012; IBM, 2026) on the easy generalized independent set problem benchmark (GISP-easy) (Colombi et al., 2017). A historical LLM workflow selected the reference configuration before the controlled study. PACER remeasures this artifact with the evaluator used for the conventional-search endpoints. The resulting comparison directly measures the artifact's held-out quality and the finite-budget performance of conventional search. The reference improves the held-out primal-dual integral (PDI) by 34.3%. PDI measures solution and bound quality over time (Berthold, 2013; Scavuzzo et al., 2024). The independent eight-trial runs provide a small-budget snapshot. Along the nested search paths, reference exceedance increases from 5 of 15 endpoints at 25 trials to 11 of 15 at 60 trials. We also replay six configurations retained from the historical workflow across three problem families. In an ablation of the easy combinatorial-auction artifact, removing its instance-dependent mixed-integer programming start changes the PDI ratio by approximately 0.02%. Fixed CPLEX settings account for nearly all of the measured PDI gain in this artifact. Our contributions are:

C1) **Controlled evaluation protocol for LLM-generated solver configurations.** PACER remeasures a fixed LLM-generated reference and evaluates conventional configuration search with a common scoring rule, validation procedure, held-out test set, solver settings, and solver-call accounting.

C2) **Budget-controlled comparison with conventional search.** On GISP-easy, we remeasure a historical LLM-generated CPLEX configuration and compare it with five conventional search methods across multiple evaluation budgets. The comparison quantifies how the standing of the reference changes as alternative methods receive more search evaluations.

C3) **Analysis of configuration quality and search opportunity.** Replays across three problem families measure the held-out quality of retained configurations. A component ablation identifies fixed solver settings as the source of nearly all measured PDI gain in the evaluated combinatorial-auction artifact. Proposition 1 relates the probability of reaching a target configuration quality to the available search budget.

2 Related Work and Preliminaries

2.1 Related work

Automated algorithm configuration and Bayesian optimization choose solver settings from feedback collected under a finite evaluation budget (Hutter et al., 2009; 2011; Bergstra et al., 2011; Snoek et al., 2012; Souza et al., 2021). Their empirical comparison requires common benchmarks, repeated runs, controlled budgets, and accounting for tuning effort (Hutter et al., 2010b; Eggensperger et al., 2019). PACER uses these established practices to make

the artifact-versus-proposer distinction explicit for LLM-generated optimization configurations. It remeasures a fixed artifact and evaluates how readily conventional proposers reach comparable held-out performance under a common evaluator and solver-call budgets. LLM-based optimization methods generate programs, heuristics, policies, and solver configurations (Romera-Paredes et al., 2024; Liu et al., 2024; Ye et al., 2024; Yang et al., 2024; Lawless et al., 2025; Luo et al., 2026). PACER evaluates the resulting configuration as a fixed reference and measures how often validation-selected alternatives exceed it. Appendix E reviews solver ecosystems, configuration packages, LLM-based optimization, and trajectory-aware scoring in greater detail.

2.2 Global solver configuration

We consider a recurring workload of mixed-integer linear programs (MILPs). Using a minimization convention, a canonical instance takes the form

$$\min_{x\in\mathbb{R}^n} c^\top x \quad \text{subject to} \quad Ax \le b, \quad x_j \in \mathbb{Z} \quad \text{for all } j \in \mathcal{K}. \tag{1}$$

Here, $x \in \mathbb{R}^n$ is the decision vector and $c \in \mathbb{R}^n$ gives its costs. For m constraints, $A \in \mathbb{R}^{m\times n}$ is the constraint matrix and $b \in \mathbb{R}^m$ gives the corresponding bounds. The set $\mathcal{K} \subseteq \{1, \dots, n\}$ indexes the integer-valued variables.

The linear-programming relaxation of (1) omits integrality. Its optimum gives a dual bound, while any integer-feasible solution provides a primal incumbent. Branch-and-cut repeatedly solves relaxations, adds valid inequalities, and branches on discrete choices. Primal heuristics search for feasible incumbents (Bixby, 2012; Achterberg et al., 2020). CPLEX (IBM, 2026) exposes global parameters that control these operations. Let θ denote one assignment of these solver parameters. CPLEX holds θ fixed while optimizing the decision vector x for each MILP instance in (1). Let $\mathcal{P}$ denote the finite set of configurable parameter axes and let V_a be the allowed domain for axis $a \in \mathcal{P}$. The global configuration space, a candidate configuration, and the CPLEX default are

$$\Theta = \prod_{a\in\mathcal{P}} V_a, \quad \theta = (\theta_a)_{a\in\mathcal{P}} \in \Theta, \quad \theta^\circ = (\theta_a^\circ)_{a\in\mathcal{P}} \in \Theta. \tag{2}$$

A global configuration applies the parameter assignment to every instance in a workload. Configuration search operates at a different level from solving a MILP. Each CPLEX execution optimizes x for one instance, while the outer search uses workload-level performance to choose θ. One *solver call* executes CPLEX on instance i with configuration θ and horizon T. Evaluating θ on an instance set requires one solver call per instance.

2.3 Trajectory-aware solve quality

Throughout, i indexes a workload instance, $\theta \in \Theta$ denotes a global configuration, θ° denotes the CPLEX default, and $t \in [0, T]$ is elapsed solve time. For a minimization instance, let $p_{i,\theta}(t)$ be the primal-incumbent value and let $d_{i,\theta}(t)$ be the dual-bound value. Write $p(t) = p_{i,\theta}(t)$ and $d(t) = d_{i,\theta}(t)$ when the instance and configuration are clear. The implemented normalized primal-dual gap is

$$\gamma_{i,\theta}(t) = \begin{cases} 0, & p(t), d(t) \in \mathbb{R} \text{ and } p(t) = d(t), \\ 1, & p(t) \text{ or } d(t) \text{ is unavailable, or } p(t)d(t) < 0, \\ \min\left\{1, \dfrac{|p(t) - d(t)|}{\max\{|p(t)|, |d(t)|\}}\right\}, & \text{otherwise.} \end{cases} \tag{3}$$

The gap equals zero when the incumbent and bound meet, and it takes its worst value of one when either trajectory is unavailable or their signs differ. The primal-dual integral (PDI) summarizes this gap over the solve horizon (Berthold, 2013; Scavuzzo et al., 2024):

$$\mathrm{PDI}_i(\theta, T) = \int_0^T \gamma_{i,\theta}(t)\,\mathrm{d}t, \quad 0 \le \mathrm{PDI}_i(\theta, T) \le T. \tag{4}$$

The implementation evaluates (4) with a left-Riemann, step-hold sum. The interval before the first solver event receives gap one. A solve that ends before finding a feasible incumbent

has $\mathrm{PDI}_i(\theta, T) = T$, while the interval after an optimality proof receives gap zero. Lower PDI therefore indicates earlier feasible solutions, tighter proof bounds, or both. This trajectory-based score rewards a configuration that finds a useful incumbent early and continues to strengthen its proof bound during the solve horizon. Section 3 uses paired PDI ratios to compare configurations. Appendix A lists the recurring notation.

3 PACER: A Controlled Evaluation Protocol

PACER first evaluates a fixed reference configuration under a common evaluator. It then uses the same evaluator to select and test configurations proposed by conventional search. Scoring, validation selection, held-out testing, and solver-call accounting remain fixed while the search method and number of validation trials k vary. Figure 1 summarizes the comparison, and Algorithm 1 gives the procedure.

3.1 Configuration score and frozen evaluator

Each workload instance is a MILP of the form in (1), and each candidate belongs to the global configuration space Θ in (2). Let $\mathcal{I}_{\mathrm{val}}$ and $\mathcal{I}_{\mathrm{test}}$ be disjoint validation and held-out test sets. The same configuration applies to every instance in both sets. The frozen evaluator ρ fixes the split, search space, CPLEX interface, solver settings, scoring rule, stored per-instance measurements of the default θ°, parallel execution, and result checks. This common evaluator gives every proposer the same mapping from a candidate configuration to validation feedback. Its checks verify each solve and reserve the held-out data for final evaluation. Section 4.1 gives the GISP-easy setting.

For split $D \in \{\mathrm{val}, \mathrm{test}\}$, write $\mathcal{I}_D$ for the corresponding validation or test set. Let $\mathcal{U}_D(\theta) \subseteq \mathcal{I}_D$ contain the instances whose configured and stored-default records both pass the validity checks. The score is the geometric mean of paired PDI ratios. Percentage improvement measures its reduction from the default ratio of one:

$$\mathrm{PDI}_D(\theta) = \left(\prod_{i \in \mathcal{U}_D(\theta)} \frac{\mathrm{PDI}_i(\theta, T)}{\mathrm{PDI}_i(\theta^\circ, T)} \right)^{1/|\mathcal{U}_D(\theta)|}, \quad \mathrm{Imp}_D(\theta) = 100\big(1 - \mathrm{PDI}_D(\theta)\big). \tag{5}$$

Lower PDI_D is better, and positive Imp_D indicates improvement. Pairing each configured solve with the stored default measurement for the same instance controls instance difficulty before aggregation. The shared stored denominator also keeps validation feedback on a common scale across every proposer and trial. Search minimizes $\mathrm{PDI}_{\mathrm{val}}$, and $\mathrm{PDI}_{\mathrm{test}}$ evaluates the selected configuration. Validity checks cover default-measurement availability, score construction, and agreement with archived objectives. Time-limited solves use (3). Audit details appear in Appendix G.

3.2 Finite-budget search

A proposer π chooses candidate configurations using previous evaluation feedback. Each additional proposal creates another opportunity to reach a target validation score. One *full-validation trial* evaluates a configuration on every validation instance and costs $|\mathcal{I}_{\mathrm{val}}|$ solver calls. For a target validation ratio τ, define the set of satisfactory configurations

$$\mathcal{A}_\tau = \{\theta \in \Theta : \mathrm{PDI}_{\mathrm{val}}(\theta) \le \tau\}. \tag{6}$$

For uniform random sampling, the probability of entering this set in one proposal round is $|\mathcal{A}_\tau|/|\Theta|$. Other proposers reduce the expected search cost when they assign greater probability to the same set. A proposal round produces one candidate configuration. Let $Q_{\pi,\tau}$ be the index of the first proposal that belongs to $\mathcal{A}_\tau$.

Proposition 1 (Reaching a satisfactory configuration)**.** *Suppose that, after every search history containing no previous hit, proposer π assigns conditional probability at least $p_\pi(\tau) > 0$ to $\mathcal{A}_\tau$. After q proposal rounds,*

$$\Pr(Q_{\pi,\tau} \le q) \ge 1 - \big(1 - p_\pi(\tau)\big)^q \ge 1 - \exp\big(-q\, p_\pi(\tau)\big). \tag{7}$$

Consequently, for any $\delta \in (0,1)$, $q \geq \log(1/\delta)/p_\pi(\tau)$ suffices for probability at least $1-\delta$, and the expected number of proposals is at most $1/p_\pi(\tau)$.

Appendix B gives the proof. Proposition 1 connects finite-budget performance to the probability assigned to satisfactory configurations. Concentrating more probability on $\mathcal{A}_\tau$ reduces the expected number of proposals needed to reach it. Prior-guided Bayesian optimization encodes beliefs about promising configurations in its search model (Souza et al., 2021). LLM-based optimization methods construct proposals from textual problem descriptions, code, and evaluation feedback (Romera-Paredes et al., 2024; Liu et al., 2024; Lawless et al., 2025; Luo et al., 2026). Both mechanisms alter the proposal distribution over Θ. Within each solver call, CPLEX optimizes x. PACER fixes the evaluator and budgets used to compare the selected configurations. Setting τ to the reference configuration's validation score gives a bound on the probability of matching or improving that score within q proposals. Under the proposition's assumption, the expected additional cost is at most $|\mathcal{I}_{\text{val}}|/p_\pi(\tau)$ CPLEX calls. We then evaluate the validation-selected configuration on the held-out set.

3.3 Proposer interface and PACER procedure

An independent random seed, indexed by s, initializes one search run. A proposer π uses `ask` to return a candidate $\theta^{(r)}_{\pi,s}$ at trial index r from previous evaluation feedback (Lindauer et al., 2022). A completed evaluation calls `tell` with its validation score. A failed evaluation calls `fail` with the recorded failure state, and all of its solver calls remain billed. We compare the tree-structured Parzen estimator (TPE), Gaussian-process search (GP), Sequential Model-based Algorithm Configuration version 3 (SMAC3), random-forest expected improvement (RF-EI), and Random search (Bergstra et al., 2011; Snoek et al., 2012; Lindauer et al., 2022; Hutter et al., 2011; Breiman, 2001; Bergstra & Bengio, 2012). TPE models promising and less promising parameter values with separate densities. GP uses a probabilistic surrogate over observed configurations. SMAC3 and RF-EI use random-forest predictions to guide candidate choice, while Random samples directly from the categorical space. These proposer families test distinct ways of using the same validation feedback. Implementation details are provided in Appendix C.

In the initialized runs, PACER applies *trial zero* to evaluate the CPLEX default θ° and records its validation score against the stored default measurements. GP, TPE, RF-EI, and Random include this trial in their histories. The archived SMAC3 searches begin from generated configurations and are marked SMAC3†. Feedback and selection use raw PDI_{val}. A budget of k trials consists of trial zero followed by $k-1$ proposed configurations. Let $\mathcal{T}_{\pi,s,k}$ contain the completed trial indices among the first k trials from proposer π and seed-s run. PACER selects

$$\widehat{\theta}_{\pi,s,k} \in \arg \min_{\theta \in \{\theta^{(r)}_{\pi,s}:\, r \in \mathcal{T}_{\pi,s,k}\}} \text{PDI}_{\text{val}}(\theta), \tag{8}$$

and evaluates $\widehat{\theta}_{\pi,s,k}$ once on the shared held-out set $\mathcal{I}_{\text{test}}$. We compare these held-out results with the remeasured reference. Section 4.4 places the search budgets alongside the historical call counts. Algorithm 1 specifies the default-initialized procedure. The archived SMAC3† runs retain their generated initialization, as reported in Section 4.1.

The solve ledger S_π records every CPLEX validation call caused by proposer π, including calls from failed evaluations. This accounting distinguishes a trial, which evaluates one candidate configuration, from the solver calls made across the validation instances. Controlled search budgets use equal CPLEX call counts at a given k under the common solve horizon and parallel-execution settings. Fixed evaluator setup, proposer computation, and wall-clock duration are reported separately. For k validation trials,

$$S_\pi(k) = k\,|\mathcal{I}_{\text{val}}|. \tag{9}$$

Successful and failed trials contribute all proposer-triggered CPLEX validation solves to S_π. On GISP-easy, one full-validation trial contains 100 solver calls, so budgets $k = 8$, 25, and 60 correspond to 800, 2,500, and 6,000 calls per run. Sharing ρ and the counting rule controls both measurement and search cost. Section 4.4 relates these budgets to the historical solver-call context. The experiments below use full validation for every trial. Appendix D records a staged-screening extension.

Algorithm 1 PACER controlled evaluation procedure.

Input: frozen evaluator ρ, proposers π, random seeds s, and budget k of validation trials
Output: one held-out PDI ratio for every proposer and seed

1. **for each** proposer π and seed s **do**
2. Initialize solve ledger $S_\pi \leftarrow 0$ and an empty list of completed validation trials
3. Initialize π so that its first candidate is the CPLEX default θ°
4. **for** $r = 0, \ldots, k-1$ **do**
5. $\theta^{(r)} \leftarrow \textsc{ask}(\pi)$
6. $\textsc{evaluate}(\rho, \theta^{(r)}, \mathcal{I}_{\text{val}})$ and add its CPLEX calls to S_π
7. **if** the evaluation returns a validation score **then**
8. $\textsc{tell}(\pi, \text{PDI}_{\text{val}}(\theta^{(r)}))$ and store the configuration and score
9. **else** report the recorded failure state through $\textsc{fail}(\pi)$
10. **end if**
11. **end for**
12. Select the completed configuration with the lowest validation PDI ratio
13. Evaluate the selected configuration once on $\mathcal{I}_{\text{test}}$ under ρ
14. **end for**
15. **return** all seed-level held-out PDI ratios

4 Experiments

The experiments provide three views of configuration performance. The controlled GISP-easy study compares the remeasured historical reference with validation-selected configurations from conventional search. A second study replays six configurations retained from the historical LLM workflow across three problem families. A component ablation then separates fixed settings from an instance-dependent solver input. Appendix K reports the original historical measurements.

4.1 Experimental setup

Table 1 summarizes the frozen evaluator and archive construction. GISP-easy uses the binary generalized independent set model (Colombi et al., 2017) with vertex set V, vertex rewards w_v, fixed-conflict edges E_0, and priced-conflict edges E_1. Each priced edge $(u, v) \in E_1$ has cost c_{uv} and admission variable y_{uv}. The variable x_v indicates whether vertex v is selected:

$$\begin{aligned} \max_{x,y} \quad & \sum_{v \in V} w_v x_v - \sum_{(u,v) \in E_1} c_{uv} y_{uv} \\ \text{subject to} \quad & x_u + x_v \le 1, && (u,v) \in E_0, \\ & x_u + x_v - y_{uv} \le 1, && (u,v) \in E_1, \\ & x_v \in \{0,1\}, && v \in V, \\ & y_{uv} \in \{0,1\}, && (u,v) \in E_1. \end{aligned} \tag{10}$$

Reversing the objective sign writes (10) in the canonical MILP form (1). Let $\mathcal{I}_{\text{GISP}}$ denote the 1,000-instance archive, partitioned into training, validation, and held-out test sets as

$$\mathcal{I}_{\text{GISP}} = \mathcal{I}_{\text{train}} \,\dot\cup\, \mathcal{I}_{\text{val}} \,\dot\cup\, \mathcal{I}_{\text{test}}, \quad (|\mathcal{I}_{\text{train}}|, |\mathcal{I}_{\text{val}}|, |\mathcal{I}_{\text{test}}|) = (800, 100, 100). \tag{11}$$

Search uses $\mathcal{I}_{\text{val}}$ for all feedback and selection. Each selected configuration then receives one evaluation on the shared held-out set $\mathcal{I}_{\text{test}}$. The evaluator uses stored default measurements. Thirty complete search traces yield 45 validation-selected configurations. All 1,020 validation records cover the full 100-instance set and pass the scoring checks. Appendix G gives the audit details.

The controlled search domain instantiates (2) with 31 categorical CPLEX parameter axes, 143 allowed values across their domains, and $|\Theta| \approx 1.12 \times 10^{20}$ configurations. These axes cover choices that govern presolve, cuts, branching, primal heuristics, and search strategy. The historical LLM interface additionally supports controls that fix or branch on individual variables, explicit lists of cutting planes, and a starting feasible solution.

Component	Setting
Workload	GISP-easy model (10) with an 800/100/100 training, validation, and test split (11).
Search space	31 categorical CPLEX axes with 143 allowed values and $\|\Theta\| \approx 1.12 \times 10^{20}$.
Evaluator	Paired PDI (5), CPLEX 22.2.0.0, 300 seconds, one thread per solve, and at most 100 concurrent solves.
Methods	TPE, GP, SMAC3†, RF-EI, and Random. The remeasured LLM-generated configuration is the reference.
Budgets	Seeds 1 to 3, $k \in \{8, 25, 60\}$ full-validation trials, and $S = 100k$ CPLEX calls.
Archive paths	Independent runs for $k = 8$ and nested prefixes of 60-trial runs for $k = 25$ and $k = 60$.
Initialization	Default at trial zero for GP, TPE, RF-EI, and Random. Generated configurations for SMAC3†.

Table 1: GISP-easy experimental setting. Search uses validation data, and each selected configuration receives one evaluation on the shared held-out set. Here, k is the number of full-validation trials and S is the billed CPLEX call count.

	Independent run	Nested 60-trial path	
Proposer	$k = 8$	$k = 25$	$k = 60$
LLM-generated reference		+34.3%	
GP	$+10.6 \pm 8.7$ (0/3)	$+43.5 \pm 25.6$ (1/3)	$+65.5 \pm 7.1$ (3/3)
TPE	$+23.6 \pm 30.0$ (1/3)	$+41.1 \pm 31.9$ (2/3)	$+52.8 \pm 30.3$ (2/3)
Random	$+28.2 \pm 38.3$ (1/3)	$+35.9 \pm 33.7$ (1/3)	$+36.1 \pm 33.2$ (1/3)
RF-EI	$+5.8 \pm 0.9$ (0/3)	$+16.0 \pm 19.4$ (1/3)	$+30.5 \pm 16.0$ (2/3)
SMAC3†	-101.3 ± 89.9 (0/3)	-5.3 ± 26.3 (0/3)	$+45.8 \pm 7.2$ (3/3)
All methods above reference	**2/15**	**5/15**	**11/15**
Default-initialized above reference	2/12	5/12	8/12

Table 2: Held-out PDI improvement (%). Method cells report mean ± sample standard deviation over three seeds, followed by seeds above the +34.3% reference in parentheses. Default-initialized methods are GP, TPE, Random, and RF-EI. SMAC3† uses generated initialization. Appendix Tables 6 through 8 report every seed.

The selected GISP-easy reference changes two global controls governing branching-variable selection and branch-and-bound backtracking. Both controls belong to the search space Θ and are available to every conventional proposer. The historical LLM agent generated method code and revised candidate configurations using solver feedback. We evaluate its selected configuration as the fixed reference. Workflow and implementation details appear in Appendix C.

4.2 Controlled comparison on GISP-easy

The fixed reference achieves $\mathrm{PDI}_{\mathrm{test}} = 0.6572$, equivalent to $\mathrm{Imp}_{\mathrm{test}} = +34.3\%$ by (5). Its PDI improves on 95 of 100 held-out instances. The original historical procedure reported +28.9% on GISP-easy, while the controlled comparison uses the +34.3% measurement from the common evaluator. A separate remeasurement of the CPLEX default gives $\mathrm{PDI}_{\mathrm{test}} = 0.9464$. The denominator remains the stored default measurement, so the remeasured ratio records execution variability under the same score definition (Appendix F).

Table 2 compares the fixed reference with configurations selected by conventional search. At $k = 8$, independent eight-trial searches exceed the reference in 2 of 15 runs. The $k = 25$ and $k = 60$ results are nested within the same 60-trial searches. Reference exceedance increases from 5 of 15 selected configurations at $k = 25$ to 11 of 15 at $k = 60$. This within-run comparison measures the effect of additional proposal and feedback rounds. Proposition 1 gives the corresponding finite-budget bound on reaching a target validation score.

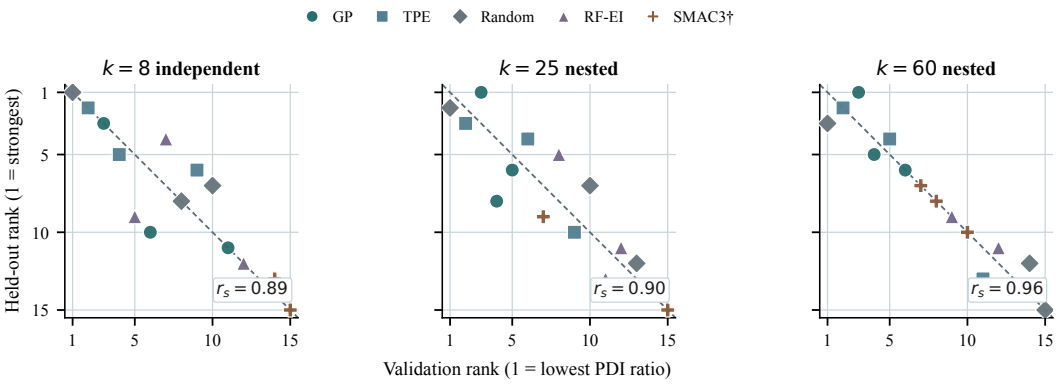

Figure 2: Validation-to-test rank agreement for all 45 selected configurations. Each marker represents one proposer and random seed. Lower validation PDI and higher held-out improvement receive rank 1, so the diagonal represents identical orderings. Spearman agreement r_s is 0.89, 0.90, and 0.96 for $k = 8$, 25, and 60. The $k = 8$ searches are independent. The $k = 25$ and $k = 60$ results use the same longer searches.

The seed-level results vary substantially within several methods. At $k = 60$, all three GP and all three archived SMAC3† configurations exceed the reference. Two TPE and two RF-EI configurations also exceed it. Among the four default-initialized methods, 8 of 12 configurations exceed the reference. One random-search seed reaches +72.0% held-out improvement, showing that direct sampling can also reach a strong configuration in this search space. These outcomes reflect both the guidance supplied by each search model and the opportunity created by additional trials. Appendix Tables 6 through 8 give all seed-level results.

The independent $k = 8$ runs provide a separate small-budget snapshot. The nested searches give the controlled budget comparison. From $k = 25$ to $k = 60$, reference exceedance rises from 5 of 15 to 11 of 15 endpoints overall and from 5 of 12 to 8 of 12 among the default-initialized methods. Validation ranking tracks held-out performance closely (Figure 2). Within each budget, lower validation PDI and higher held-out improvement receive rank 1. Spearman rank correlation r_s (Spearman, 1904) is 0.89, 0.90, and 0.96 at $k = 8$, 25, and 60 across the 15 proposer-seed results. These correlations indicate strong agreement between validation and held-out configuration quality.

4.3 Artifact replays and component analysis

We replay one retained artifact for each easy and medium variant of maximum independent set (MIS) (Gasse et al., 2019), combinatorial auction winner determination (CA) (Leyton-Brown et al., 2000; Sandholm, 2002), and generalized independent set (GISP) (Colombi et al., 2017). Each artifact contains fixed solver settings and, when present, an instance-dependent input to CPLEX. Every variant uses an 800/100/100 training, validation, and test split. Replays use CPLEX 22.2, one thread, a 300-second horizon, and the score in (5). These replays measure the held-out quality of retained artifacts across workloads, complementing the proposer comparison on GISP-easy. Appendix L gives the settings and development history. The GISP-easy replay uses a later three-parameter artifact, while the controlled comparison uses the earlier two-parameter reference.

All six point estimates improve both PDI and wall-clock time (Table 3). The CA and GISP artifacts reduce PDI by 21.4% to 34.4%. The MIS artifacts reduce PDI by 6.1% to 6.2%. The artifact contents identify the components supplied to the solver. MIS-easy applies the same seven nondefault settings to every test instance, and GISP-easy applies the same three settings. CA-easy combines four fixed settings with a learned *mixed-integer programming (MIP) start*, an instance-dependent starting solution supplied to CPLEX (IBM, 2026). The start is available for all 100 CA-easy predictions. The complete CA artifact gives a PDI ratio of 0.787882 and a wall-clock speedup of 1.290×. Removing the MIP start while retaining

Variant	Pairs	PDI ratio	PDI imp. (%)	Time ratio
MIS-easy	100	0.938	+6.2%	0.861
MIS-medium	100	0.939	+6.1%	0.616
CA-easy	98	0.786	+21.4%	0.780
CA-medium	100	0.656	+34.4%	0.693
GISP-easy	100	0.704	+29.6%	0.688
GISP-medium	100	0.740	+26.0%	0.907

Table 3: Held-out replay of six retained artifacts. PDI and time are geometric-mean configured/default ratios, so lower values are better. CA-easy contains 98 objective-matched pairs. The GISP-easy row uses a later artifact than the reference in Table 2.

the four fixed settings gives a PDI ratio of 0.788030 and a speedup of 1.282×. The PDI ratio changes by approximately 0.02%, which attributes nearly all of the measured gain to the fixed settings. Complete ablation is provided in Appendix L.

The replay and controlled-search studies answer two distinct questions. Artifact replays measure configuration quality after selection, while the controlled searches measure how conventional proposal methods perform under finite validation budgets. The CA-easy ablation locates nearly all of that artifact's measured PDI gain in its fixed solver settings. Proposition 1 separates the probability assigned to strong settings from the number of available proposals. The analysis also distinguishes two optimization levels. A CPLEX optimality certificate concerns the MILP decision vector x in (1) (Bixby, 2012; IBM, 2026), whereas PACER searches over the solver configuration θ.

4.4 Budget alignment and robustness checks

The historical workflow documents 2,064 solver calls, comprising 1,164 sandbox calls and 900 platform-validation calls. The retained pre-test round records sum to 2,109 calls. Under (9), both totals are closest to $k^\star = 21$ full-validation trials. We call this a *Partial solver-call alignment*: it places the historical and PACER costs in the same solver-call currency while retaining their respective evaluation procedures. Appendix Figure 3 and Tables 13 and 14 show conventional-search progress at this budget. Appendix K gives the call breakdown.

An alternative PDI aggregation preserves every endpoint's classification relative to the reference (Appendix H). PDI and wall-clock time both improve in 36 of 45 results. Four configurations improve PDI while increasing wall-clock time, and two reduce wall-clock time while worsening PDI (Appendix I). Two earlier artifacts selected for node reduction further illustrate the difference among metrics. They reduce node counts by 90.2% on MIS-easy and 65.2% on CA-easy, while worsening PDI by 53.5% and 88.8%, respectively. Appendix L.1 reports the corresponding runtime ratios.

The evaluator uses one stored per-instance default denominator for every candidate, placing every search evaluation on a common scale. Fresh default measurements range from 0.9571 to 1.3103 relative to these stored values across the 24 default-initialized runs. These measurements quantify execution variation under the shared 100-way concurrency setting. Table 2 summarizes variation across searches and their selected held-out evaluations. Per-instance diagnostics appear in Appendix G.

5 Conclusion

PACER separates the held-out quality of an LLM-generated solver configuration from the finite-budget performance of alternative search methods. On GISP-easy, the nested 25-to-60-trial comparison shows that additional search yields more reference-exceeding endpoints, while the independent eight-trial runs provide a small-budget snapshot. Proposition 1 links this budget effect to the probability assigned to target configurations. Replays across three problem families measure retained-artifact quality, and the CA-easy ablation attributes nearly all measured PDI gain to fixed CPLEX settings. PACER thereby supports controlled comparisons of configuration quality, proposal methods, and their search opportunity.

AI use statement

Generative artificial intelligence tools were used to assist with writing and editing, code development, and figure preparation. The authors reviewed and verified all AI-assisted content and take responsibility for the final manuscript.

Reproducibility statement

Section 3 defines PACER, its scoring rule, validation selection, held-out evaluation, and solver-call accounting. Section 4.1 specifies the workload, search space, proposer budgets, CPLEX 22.2.0.0 settings, initialization, and random seeds. Appendices C, F, and G report the proposer implementations, complete seed-level endpoints, validation summaries, and archive checks. The accompanying `bo_paper_kit_full/` and code archives contain the evaluated configurations, structured search records, software environments, evaluator commands, and figure inputs.

References

Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020. doi: 10.1287/ijoc.2018.0857.

Ali Ahmaditeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 577–596, 2024. URL `https://proceedings.mlr.press/v235/ahmaditeshnizi24a.html`.

Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Principles and Practice of Constraint Programming (CP)*, pp. 142–157, 2009. doi: 10.1007/978-3-642-04244-7_14.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, pp. 2546–2554, 2011.

Timo Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41 (6):611–614, 2013. doi: 10.1016/j.orl.2013.08.007.

Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. Enabling research through the SCIP optimization suite 8.0. *ACM Transactions on Mathematical Software*, 49(2):1–21, 2023. doi: 10.1145/3585516.

Robert E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pp. 107–121, 2012. doi: 10.4171/DMS/6/16. Extra Volume: Optimization Stories.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Marco Colombi, Renata Mansini, and Martin Savelsbergh. The generalized independent set problem: Polyhedral analysis and solution approaches. *European Journal of Operational Research*, 260(1):41–55, 2017. doi: 10.1016/j.ejor.2016.11.050.

Alexander I. Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, and Haitham Bou-Ammar. HEBO: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349, 2022. doi: 10.1613/jair.1.13643.

Pham Vu Tuan Dat, Long Doan, and Huynh Thi Thanh Binh. HSEvo: Elevating automatic heuristic design with diversity-driven harmony search and genetic algorithm using LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26931–26938, 2025. doi: 10.1609/aaai.v39i25.34898.

Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pp. 1114–1120, 2015. doi: 10.1609/aaai.v29i1.9375.

Katharina Eggensperger, Marius Lindauer, and Frank Hutter. Pitfalls and best practices in algorithm configuration. *Journal of Artificial Intelligence Research*, 64:861–893, 2019. doi: 10.1613/JAIR.1.11420.

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

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. `https://docs.gurobi.com/projects/optimizer/en/current/`, 2026. Accessed: 2026-08-21.

HiGHS Development Team. HiGHS: High-performance parallel linear optimization software. `https://highs.dev/`, 2026. Accessed: 2026-08-21.

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. doi: 10.1613/jair.2861.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Automated configuration of mixed integer programming solvers. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, pp. 186–202, 2010a. doi: 10.1007/978-3-642-13520-0_23.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Tradeoffs in the empirical evaluation of competing algorithm designs. *Annals of Mathematics and Artificial Intelligence*, 60:65–89, 2010b. doi: 10.1007/s10472-010-9191-0.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization (LION)*, pp. 507–523, 2011. doi: 10.1007/978-3-642-25566-3_40.

IBM. Branch and cut in CPLEX. `https://www.ibm.com/docs/en/icos/22.1.2?topic=c-branch-cut-in-cplex-1`, 2026. Accessed: 2026-08-27.

Zangir Iklassov, Yali Du, Farkhad Akimov, and Martin Takáč. Self-guiding exploration for combinatorial problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. doi: 10.52202/079017-4150.

Jihwan Jeong, Parth Jaggi, Andrew Butler, and Scott Sanner. An exact symbolic reduction of linear smart Predict+Optimize to mixed integer linear programming. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10053–10067, 2022. URL `https://proceedings.mlr.press/v162/jeong22a.html`.

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. LLMOPT: Learning to define and solve general optimization problems from scratch. In *International Conference on Learning Representations (ICLR)*, 2025a.

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025b. doi: 10.52202/085713-5495.

Connor Lawless, Yingxi Li, Anders Wikum, Madeleine Udell, and Ellen Vitercik. LLMs for cold-start cutting plane separator configuration. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pp. 51–69, 2025. doi: 10.1007/978-3-031-95976-9_4.

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

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 (NeurIPS)*, volume 36, 2023.

Sirui Li, Janardhan Kulkarni, Ishai Menache, Cathy Wu, and Beibin Li. Towards foundation models for mixed integer linear programming. In *International Conference on Learning Representations (ICLR)*, 2025.

Xijun Li, Jiexiang Yang, Jinghao Wang, Bo Peng, Jianguo Yao, and Haibing Guan. STR-CMP: Integrating graph structural priors with language models for combinatorial optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2026.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223, 2024. URL `https://proceedings.mlr.press/v235/liu24bs.html`.

Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002.

Yidong Luo, Xuemin Chen, Chenguang Wang, Fangzhou Zhu, Tao Zhong, and Tianshu Yu. GRIMIP: A general framework for instance-specific configuration of MIP solvers using LLMs, 2026. Concurrent work.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25444–25470, 2023. URL `https://proceedings.mlr.press/v202/muller23a.html`.

Theodore P. Papalexopoulos, Christian Tjandraatmadja, Ross Anderson, Juan Pablo Vielma, and David Belanger. Constrained discrete black-box optimization using mixed-integer programming. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17295–17322, 2022. URL `https://proceedings.mlr.press/v162/papalexopoulos22a.html`.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1–2):1–54, 2002. doi: 10.1016/S0004-3702(01)00159-X.

Lara Scavuzzo, Karen Aardal, Andrea Lodi, and Neil Yorke-Smith. Machine learning augmented branch and bound for mixed integer linear programming. *Mathematical Programming*, 217:123–166, 2024. doi: 10.1007/s10107-024-02130-y.

Elias Schede, Jasmin Brandt, Alexander Tornede, Marcel Wever, Viktor Bengs, Eyke Hüllermeier, and Kevin Tierney. A survey of methods for automated algorithm configuration. *Journal of Artificial Intelligence Research*, 75:425–487, 2022. doi: 10.1613/jair.1.13676.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pp. 2951–2959, 2012.

Artur L. F. Souza, Luigi Nardi, Leonardo B. Oliveira, Kunle Olukotun, Marius Lindauer, and Frank Hutter. Bayesian optimization with a prior for the optimum. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, volume 12977 of *Lecture Notes in Computer Science*, pp. 265–296. Springer, 2021. doi: 10.1007/978-3-030-86523-8_17.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. doi: 10.2307/1412159.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR)*, 2019. URL `https://arxiv.org/pdf/1711.07356`.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *International Conference on Learning Representations (ICLR)*, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. OptiBench meets ReSocratic: Measure and improve LLMs for optimization modeling. In *International Conference on Learning Representations (ICLR)*, 2025.

Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 27144–27152, 2025. doi: 10.1609/aaai.v39i25.34922.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. doi: 10.52202/079017-1381.

Huigen Ye, Hua Xu, An Yan, and Yaoyang Cheng. Large language model-driven large neighborhood search for large-scale MILP problems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 72131–72180, 2025a. URL `https://proceedings.mlr.press/v267/ye25j.html`.

Mingxuan Ye, Jie Wang, Fangzhou Zhu, Zhihai Wang, Yufei Kuang, Xijun Li, Weilin Luo, Jianye Hao, and Feng Wu. Dynamic configuration for cutting plane separators via reinforcement learning on incremental graph. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025b. doi: 10.52202/085713-4582.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78338–78373, 2025. URL `https://proceedings.mlr.press/v267/zheng25o.html`.

A Notation and experimental conventions

The list gives notation used across sections. Model-specific symbols are defined beside their equations.

x Decision vector optimized for one MILP instance.

Θ, θ, θ° Global CPLEX configuration space, candidate configuration, and CPLEX default.

ρ, π Frozen evaluator and conventional-search proposer.

$\mathcal{I}_{\text{val}}$, $\mathcal{I}_{\text{test}}$ Validation and held-out test instance sets.

T Fixed time horizon for one CPLEX solve.

$\gamma_{i,\theta}(t)$, $\text{PDI}_i(\theta, T)$ Normalized primal-dual gap and its integral for instance i.

D, $\mathcal{U}_D(\theta)$ Data split, either validation or test, and its valid paired instances for θ.

$\text{PDI}_D(\theta)$, $\text{Imp}_D(\theta)$ Paired PDI ratio and percentage improvement on split D.

s, r, k, $\widehat{\theta}_{\pi,s,k}$ Seed, trial index, validation budget, and selected configuration.

S, $S_\pi(k)$ Solver-call ledger and CPLEX calls billed to proposer π through k validation trials.

$k^\star$ PACER validation-trial budget corresponding most closely to the historical GISP-easy call-count summaries in Section 4.4.

$\mathcal{A}_\tau$, $p_\pi(\tau)$, $Q_{\pi,\tau}$ Configurations meeting validation target τ, the proposer's conditional probability of reaching that set, and the first proposal index that reaches it.

B Proof of Proposition 1

Proof. Conditioned on the absence of an earlier hit, the probability of missing $\mathcal{A}_\tau$ in the next round is at most $1 - p_\pi(\tau)$. The chain rule bounds the probability of q consecutive misses by $(1 - p_\pi(\tau))^q$, which is at most $\exp(-qp_\pi(\tau))$. Taking complements gives (7). Summing the tail probabilities gives $\mathbb{E}[Q_{\pi,\tau}] \leq \sum_{q\geq 0}(1 - p_\pi(\tau))^q = 1/p_\pi(\tau)$. □

C Historical workflow and proposer implementations

Historical LLM workflow records. The retained platform documentation describes an LLM designer agent that wrote method code, chose candidate CPLEX configurations for sandbox evaluation, and revised them between platform refinement rounds. For GISP-easy, the archive records 1,164 sandbox solver calls that produced 12 distinct candidates and nine platform validation rounds of 100 calls each. It also retains the proposed configurations, round outcomes, and final selected configuration. The retained metadata cover these execution artifacts. Model identity, prompt revisions, LLM sampling controls, and human-intervention records are unavailable in the retained archive. The historical interface supports the 31 global CPLEX axes below together with per-variable fixing and branching, explicit cut lists, and a mixed-integer starting solution. The shipped GISP-easy configuration sets two global axes and belongs to the controlled space Θ. For the other workloads, the archive preserves the configurations and results produced through the richer historical interface.

Controlled proposer implementations. Table 4 records the implementation choices for the five controlled proposers. All proposer-generated validation calls pass through the evaluator in Section 3. The Optuna software-library implementations of TPE, GP, and Random, together with RF-EI, evaluate the default at trial zero. SMAC3† starts from generated configurations. Algorithm 1 specifies default initialization.

Configurations are canonicalized by removing axes set to the CPLEX default, so the empty dictionary represents θ°. Repeated canonical configurations remain eligible and receive

Proposer	Software	Proposal and representation	Initial design
TPE	Optuna 4.9.0 `TPESampler`	Native categorical axes. Density-estimation and candidate-selection options use the versioned library defaults.	3 startup trials for $k = 8$ and 8 for the nested $k = 25, 60$ paths.
GP	Optuna 4.9.0 `GPSampler`	Native categorical axes with a categorical-distance covariance function. Library defaults select candidates.	The same 3-trial or 8-trial startup rule.
RF-EI	scikit-learn 1.4.2 random forest	One numerical indicator per categorical value, known as one-hot encoding, 100 trees, minimum leaf size 1, and one worker. Expected improvement uses per-tree mean and standard deviation with a 0.005 standard-deviation floor and 2,048 fresh uniform candidates.	The same 3-trial or 8-trial startup rule.
Random	Optuna 4.9.0 `RandomSampler`	Independent uniform sampling over the allowed values of every categorical axis.	Default first, then random samples.
SMAC3†	SMAC3 2.4.0, scikit-learn 1.6.1, and pyrfr 0.9.0	Categorical `ConfigSpace` with the `HyperparameterOptimizationFacade`, a deterministic scenario, per-seed history, and a random forest.	Archived generated initial design with 3 or 8 configurations.

Table 4: Controlled proposer implementations. All methods use the same categorical domains, seeds, validation feedback, and billed CPLEX evaluations. TPE, GP, and SMAC3 use the listed software defaults unless specified.

separate validation evaluations and billed calls. Every axis is exposed categorically, including integer-valued strategy codes. The evaluator delegates parameter application to the CPLEX bridge, and the applicator audit accepts all 143 allowed axis values.

CPLEX parameter	Allowed values	CPLEX parameter	Allowed values
emphasis.mip	0, 1, 2, 3, 4, 5	mip.strategy.branch	−1, 0, 1
mip.cuts.cliques	−1, 0, 1, 2, 3	mip.strategy.dive	0, 1, 2, 3
mip.cuts.covers	−1, 0, 1, 2, 3	mip.strategy.fpheur	−1, 0, 1, 2
mip.cuts.disjunctive	−1, 0, 1, 2, 3	mip.strategy.heuristicfreq	−1, 0, 5, 10, 20, 50, 100
mip.cuts.flowcovers	−1, 0, 1, 2	mip.strategy.lbheur	0, 1
mip.cuts.gomory	−1, 0, 1, 2	mip.strategy.nodeselect	0, 1, 2, 3
mip.cuts.gubcovers	−1, 0, 1, 2	mip.strategy.probe	−1, 0, 1, 2, 3
mip.cuts.implied	−1, 0, 1, 2	mip.strategy.rinsheur	−1, 0, 10, 25, 50, 100
mip.cuts.liftproj	−1, 0, 1, 2, 3	mip.strategy.variableselect	−1, 0, 1, 2, 3, 4
mip.cuts.localimplied	−1, 0, 1, 2, 3	preprocessing.aggregator	−1, 0, 1, 5, 20
mip.cuts.mircut	−1, 0, 1, 2	preprocessing.boundstrength	−1, 0, 1
mip.cuts.pathcut	−1, 0, 1, 2	preprocessing.coeffreduce	−1, 0, 1, 2, 3
mip.cuts.zerohalfcut	−1, 0, 1, 2	preprocessing.dependency	−1, 0, 1, 2, 3
mip.strategy.backtrack	0.05, 0.25, 0.5, 0.9, 0.9999, 1.0	preprocessing.linear	0, 1
mip.strategy.bbinterval	0, 1, 3, 7, 15, 50	preprocessing.reduce	0, 1, 2, 3
		preprocessing.symmetry	−1, 0, 1, 2, 3, 4, 5

Table 5: Global CPLEX search space Θ. Every axis is categorical and includes its CPLEX default. Prespecified ladders discretize the five wide or continuous library domains.

D Staged evaluation and solver-call accounting

PACER evaluates candidates through L prespecified stages and records every stage in one solve ledger. A *screen* evaluates a candidate on a smaller instance set, and a *promotion rule* selects which screened candidate receives a larger-stage evaluation. If stage $\ell \in \{1, \ldots, L\}$ evaluates $N_{\pi,\ell}$ candidates on instance set $\mathcal{I}_\ell$, proposer π receives the billed cost

$$S_\pi = \sum_{\ell=1}^{L} N_{\pi,\ell} |\mathcal{I}_\ell|. \tag{12}$$

The implemented two-stage prototype screens five candidates on 10 instances, selects the best completed screen for evaluation on all 100 validation instances, and bills $5 \cdot 10 + 100 = 150$ calls per batch. Screen scores guide subsequent proposals. Completed 100-instance evaluations determine the selected configuration. The controlled experiments in Section 4 evaluate every trial on all 100 validation instances, as specified in Algorithm 1.

E Extended related work

MILP solver ecosystems and configuration trade-offs. CPLEX (Bixby, 2012; IBM, 2026), Gurobi (Gurobi Optimization, LLC, 2026), the Solving Constraint Integer Programs (SCIP) suite (Bestuzheva et al., 2023), and HiGHS (HiGHS Development Team, 2026) provide complementary solver ecosystems for MILPs. CPLEX and Gurobi offer mature commercial optimizers with documented parameter interfaces. SCIP provides plug-ins for research extensions. HiGHS offers open-source linear and mixed-integer solvers.

Algorithm configuration. Automated algorithm configuration selects solver settings from performance observations on an instance family (Hutter et al., 2009; 2010a; 2011; López-Ibáñez et al., 2016; Ansótegui et al., 2009; Bergstra et al., 2011; Snoek et al., 2012; Schede et al., 2022). Bayesian optimization and modern configuration packages guide proposals with learned performance models, also called surrogates (Lindauer et al., 2022; Cowen-Rivers et al., 2022; Müller et al., 2023). Random search measures configurations accessible through direct sampling (Bergstra & Bengio, 2012). Prior-guided Bayesian optimization incorporates expert beliefs about promising regions to improve finite-budget search (Souza et al., 2021). PACER compares these methods using the same scoring rule and solver-call accounting.

LLM-based optimization and solver configuration. LLM-driven systems generate programs, heuristics, and policies for combinatorial optimization (Romera-Paredes et al., 2024; Liu et al., 2024; Ye et al., 2024; Yang et al., 2024; Zheng et al., 2025; Dat et al., 2025; Yao et al., 2025; Li et al., 2026; Ye et al., 2025a; Iklassov et al., 2024). They formulate MILP models from natural language (Xiao et al., 2024; Ahmaditeshnizi et al., 2024; Li et al., 2025; Jiang et al., 2025a; Yang et al., 2025), solve combinatorial problems end to end (Jiang et al., 2025b), and configure cutting-plane separators or other instance-level settings (Lawless et al., 2025; Luo et al., 2026). Appendix K reports historical configurations from this setting.

Trajectory-aware evaluation and controlled comparison. Trajectory integrals capture solution and bound quality throughout a time limit (Berthold, 2013; Scavuzzo et al., 2024; Li et al., 2023; Ye et al., 2025b). Empirical comparisons of configuration methods require controlled budgets, repeated runs, and benchmark designs that account for tuning choices (Hutter et al., 2010b; Eggensperger et al., 2019). Surrogate benchmarks reduce the cost of evaluating hyperparameter optimizers (Eggensperger et al., 2015), and recent Bayesian-optimization methods study efficient search under finite evaluation budgets (Cowen-Rivers et al., 2022; Müller et al., 2023). We use CPLEX because its global parameter interface and solve records expose the configuration choices and primal-bound trajectories required by the evaluator (Bixby, 2012; IBM, 2026).

Proposer	k	Seed	PDI	Imp. (%)	Time	Nodes	PDI B/W
CPLEX default	reference		0.9464	+5.4	0.964	1.000	67/33
LLM reference	reference		0.6572	+34.3	0.733	0.683	95/5
GP	8	1	0.9446	+5.5	0.952	0.991	65/35
GP	8	2	0.7940	+20.6	1.183	0.805	82/18
GP	8	3	0.9446	+5.5	0.960	1.000	67/33
GP	25	1	0.7032	+29.7	0.656	0.873	90/10
GP	60	1	0.4156	+58.4	0.384	0.747	99/1
GP	25	2	0.7219	+27.8	0.677	0.948	90/10
GP	60	2	0.3462	+65.4	0.408	0.511	100/0
GP	25	3	0.2692	+73.1	0.388	0.451	100/0
GP	60	3	0.2735	+72.6	0.391	0.451	100/0
Random	8	1	0.9390	+6.1	0.956	1.000	67/33
Random	8	2	0.9387	+6.1	0.955	1.000	67/33
Random	8	3	0.2761	+72.4	0.400	0.448	100/0
Random	25	1	0.9407	+5.9	0.955	1.000	65/35
Random	60	1	0.9359	+6.4	0.952	1.000	67/33
Random	25	2	0.7045	+29.5	1.209	1.990	87/13
Random	60	2	0.7002	+30.0	1.208	1.990	87/13
Random	25	3	0.2764	+72.4	0.401	0.448	100/0
Random	60	3	0.2803	+72.0	0.403	0.448	100/0

Table 6: Complete held-out results, part 1 of 3. PDI, time, and nodes are configured/default ratios. PDI B/W counts test instances with better/worse PDI.

Proposer	k	Seed	PDI	Imp. (%)	Time	Nodes	PDI B/W
RF-EI	8	1	0.9498	+5.0	0.961	1.000	67/33
RF-EI	8	2	0.9323	+6.8	0.950	1.000	67/33
RF-EI	8	3	0.9425	+5.8	0.953	1.000	65/35
RF-EI	25	1	0.6173	+38.3	0.733	0.420	92/8
RF-EI	60	1	0.6209	+37.9	0.735	0.420	91/9
RF-EI	25	2	0.9278	+7.2	0.946	1.000	67/33
RF-EI	60	2	0.5853	+41.5	0.588	0.629	97/3
RF-EI	25	3	0.9754	+2.5	0.867	0.441	63/37
RF-EI	60	3	0.8786	+12.1	0.831	1.108	81/19
SMAC3†	8	1	1.4747	-47.5	0.960	1.190	9/91
SMAC3†	8	2	1.5130	-51.3	1.099	0.949	6/94
SMAC3†	8	3	3.0507	-205.1	1.796	1.840	0/100
SMAC3†	25	1	0.7497	+25.0	0.749	1.080	88/12
SMAC3†	60	1	0.5747	+42.5	0.594	0.857	95/5
SMAC3†	25	2	1.2121	-21.2	1.160	0.832	23/77
SMAC3†	60	2	0.5909	+40.9	0.524	0.919	92/8
SMAC3†	25	3	1.1979	-19.8	0.675	1.237	25/75
SMAC3†	60	3	0.4590	+54.1	0.526	0.429	98/2

Table 7: Complete held-out results, part 2 of 3. Table 6 defines the columns.

F Complete held-out endpoint results

Tables 6 through 8 contain every held-out result in the final archive. The PDI ratio column reports $\mathrm{PDI}_{\mathrm{test}}$ from (5). Wall-clock time and branch-and-bound node count are geometric ratios relative to the paired CPLEX default, and PDI B/W gives the number of held-out instances with better/worse PDI. All reported held-out evaluations completed successfully. Every PDI ratio uses stored paired default measurements as its denominator. The displayed CPLEX-default row is a separate held-out remeasurement with $\mathrm{PDI}_{\mathrm{test}} = 0.9464$, which records variation around the stored denominator. The archive stores full settings.

Proposer	k	Seed	PDI	Imp. (%)	Time	Nodes	PDI B/W
TPE	8	1	0.9370	+6.3	0.953	1.000	68/32
TPE	8	2	0.4177	+58.2	0.548	0.801	98/2
TPE	8	3	0.9381	+6.2	0.952	1.000	67/33
TPE	25	1	0.9163	+8.4	1.184	0.374	65/35
TPE	60	1	0.8210	+17.9	0.716	0.930	83/17
TPE	25	2	0.5731	+42.7	0.549	0.736	93/7
TPE	60	2	0.3179	+68.2	0.408	0.481	100/0
TPE	25	3	0.2786	+72.1	0.405	0.448	100/0
TPE	60	3	0.2769	+72.3	0.402	0.448	100/0

Table 8: Complete held-out results, part 3 of 3. Table 6 defines the columns.

Proposer	Seed	Best validation PDI ratio			Best trial	Completed/failed		Time (h)
		$k = 8$	$k = 25$	$k = 60$		$k = 8$	$k = 60$	
GP	1	1.0525	0.8051	0.4803	50	8/0	60/0	1.8
GP	2	0.8903	0.8050	0.3977	54	8/0	60/0	1.8
GP	3	1.2228	0.3766	0.3766	21	8/0	60/0	2.3
TPE	1	1.0002	0.9345	0.8767	54	8/0	60/0	2.0
TPE	2	0.6002	0.8744	0.4555	37	8/0	60/0	2.2
TPE	3	1.1872	0.3583	0.3583	3	8/0	60/0	2.4
Random	1	1.1607	1.1751	1.1751	0	8/0	60/0	4.0
Random	2	1.1981	0.9714	0.9714	10	8/0	60/0	3.8
Random	3	0.3072	0.2934	0.2934	3	8/0	60/0	3.5
RF-EI	1	1.2904	0.9081	0.9081	22	8/0	60/0	2.5
RF-EI	2	1.0646	1.0746	0.6779	36	8/0	60/0	2.1
RF-EI	3	1.0223	1.0085	0.9314	58	8/0	60/0	2.4
SMAC3†	1	1.6961	0.8860	0.6416	51	8/0	60/0	2.6
SMAC3†	2	1.6946	1.4102	0.7109	43	8/0	60/0	3.0
SMAC3†	3	3.1946	1.2595	0.4875	57	8/0	60/0	2.0

Table 9: Validation-search summary using $\mathrm{PDI}_{\mathrm{val}}$ from (5). Completion columns correspond to $k = 8$ and $k = 60$. Time is the duration of the 60-trial run.

G Validation summaries and audit details

Table 9 summarizes 15 proposer-seed pairs drawn from 30 archived searches. Its three validation PDI columns report the best completed $\mathrm{PDI}_{\mathrm{val}}$ after 8, 25, and 60 validation trials. The 8-trial value comes from the independent run. The 25- and 60-trial values come from the same 60-trial search. The best-trial column gives the zero-based index of the lowest-scoring trial in that search. The completion columns report completed and failed trials, and the wall-time column gives the duration in hours for the 60-trial search. Every archived run completed its scheduled validation trials.

The configuration-applicator check verifies that each enumerated parameter value can be passed to CPLEX, and it succeeds for all 143 values. The structured archive stores score and evaluator metadata for all 1,020 validation trials. The raw-log subset additionally stores per-instance PDI fields, coverage checks, validation configurations, and *fingerprints*, which identify the cache and evaluator settings used for a record. Both sources agree with every recorded proven optimum. The final numerical source consists of `final_test.v4.json`, `final_configs.v4.json`, and the 30 `search.*.v4.json` records in `bo_paper_kit_full/results/`. The companion archive documents its construction through intermediate reference reruns, predecessor records, and deferred-run materials.

H Alternative PDI aggregation

The primary score PDI_D in (5) is the geometric mean of paired PDI ratios. The archive also stores an aggregate PDI ratio, defined as the sum of configured PDI values divided by

Audit measure	Value
Structured validation records	1,020 across 30 complete search traces
PDI coverage	100 paired instances for every validation evaluation
Time-limited solves in structured archive	19,458
Raw-log audit subset	944 validation records
Usable instances and cache coverage	100/100 for every audited record
Objective mismatches, exclusions, and score errors	0, 0, and 0
Held-out selection boundary	45 validation-selected configurations, one remeasured CPLEX default, and one remeasured LLM-generated configuration
Initialization and billed cost	All proposer-triggered validation calls are billed. SMAC3† uses generated initialization.
Default score at trial zero	0.9571 to 1.3103 across the 24 GP, TPE, Random, and RF-EI archive runs
SMAC3† initial values	29.5390 to 47.0744 with generated initialization
Raw records with a time-limited solve	244
Time-limited solves in raw audit	16,681

Table 10: Archive audit. Structured metadata covers every validation evaluation. The raw-log subset adds per-instance traceability.

Aggregation	$k = 8$	$k = 25$	$k = 60$
Geometric-mean PDI ratio	2/15	5/15	11/15
Aggregate PDI ratio	2/15	5/15	11/15

Table 11: Reference exceedance under two PDI aggregation rules. Entries count selected configurations above the remeasured LLM value among 15 results per budget. The two rules agree on all 45 classifications.

PDI direction	Time < 1	Time > 1	Nodes < 1	Nodes > 1
PDI ratio < 1	36	4	25	15
PDI ratio > 1	2	3	2	3

Table 12: Metric directions for 45 conventional-search results. Time and node values are geometric-mean configured/default ratios.

the sum of their paired default values over the usable instances. Table 11 compares which selected configurations exceed the separately remeasured LLM value under each aggregation.

I Wall-clock and node-count diagnostics

Tables 6 through 8 report the PDI, wall-clock, and node ratios for every selected configuration. Table 12 summarizes the directions of these changes across the 45 conventional-proposer results. A ratio below one favors the selected configuration for the indicated metric.

J Historical solver-call records

The solver-call values in Section 4.4 are reconstructed directly from the archived 60-trial searches. Figure 3 shows the best validation score found as the call count increases, and Table 14 reports the values after 21, 25, and 60 trials.

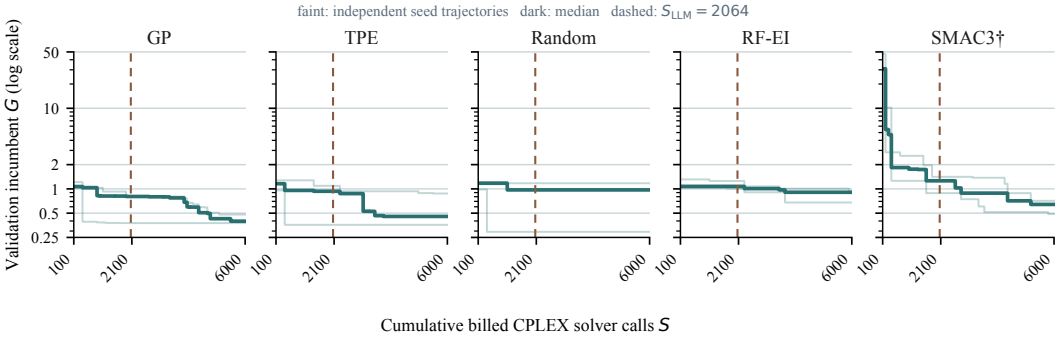

Figure 3: Best validation PDI ratio against billed CPLEX calls. Faint curves show three seeds, and dark curves show their medians on a logarithmic PDI scale. The dotted horizontal line marks the default ratio of one. The dashed vertical line marks $k^\star = 21$ PACER trials from Section 4.4. SMAC3† starts from generated configurations.

Proposer	Seed 1	Seed 2	Seed 3	Median [min., max.]
GP	0.8051	0.8050	0.3768	0.8050 [0.3768, 0.8051]
TPE	0.9345	1.0912	0.3583	0.9345 [0.3583, 1.0912]
Random	1.1751	0.9714	0.2934	0.9714 [0.2934, 1.1751]
RF-EI	1.2540	1.0746	1.0085	1.0746 [1.0085, 1.2540]
SMAC3†	0.8860	1.4102	1.2595	1.2595 [0.8860, 1.4102]

Table 13: Best validation PDI ratio after $k^\star = 21$ trials. Seeds are independent. The final column gives the median and observed range. Lower is better.

Proposer	$k = 21$	$k = 25$	$k = 60$
GP	0.8050 [0.3768, 0.8051]	0.8050 [0.3766, 0.8051]	0.3977 [0.3766, 0.4803]
TPE	0.9345 [0.3583, 1.0912]	0.8744 [0.3583, 0.9345]	0.4555 [0.3583, 0.8767]
Random	0.9714 [0.2934, 1.1751]	0.9714 [0.2934, 1.1751]	0.9714 [0.2934, 1.1751]
RF-EI	1.0746 [1.0085, 1.2540]	1.0085 [0.9081, 1.0746]	0.9081 [0.6779, 0.9314]
SMAC3†	1.2595 [0.8860, 1.4102]	1.2595 [0.8860, 1.4102]	0.6416 [0.4875, 0.7109]

Table 14: Best validation PDI ratio after 21, 25, and 60 trials. Entries give the median and observed range across three seeds. Lower is better.

K Historical LLM results across five workloads

K.1 Optimization models for the four additional variants

The historical archive contains four workload variants with completed held-out evaluations beyond GISP-easy: MIS-easy, CA-easy, CA-medium, and GISP-medium. They instantiate three optimization models. MIS-easy uses maximum independent set (MIS), CA-easy and CA-medium share the combinatorial auction (CA) winner-determination model, and GISP-medium uses the generalized independent set problem (GISP) in (10). The suffixes *easy* and *medium* identify instance-generator regimes. Each pair shares one mathematical formulation, stated once below.

Maximum independent set. Given an undirected graph $\mathcal{G} = (V, E)$, an independent set contains no adjacent pair of vertices. Let $x_v = 1$ when vertex v is selected and $x_v = 0$ otherwise. The maximum independent set model is (Gasse et al., 2019)

$$\begin{aligned} \max_{x \in \{0,1\}^{|V|}} \quad & \sum_{v \in V} x_v \\ \text{subject to} \quad & x_u + x_v \leq 1, \quad \{u, v\} \in E. \end{aligned} \tag{13}$$

The objective counts the selected vertices, and each edge constraint prevents both endpoints from being selected. A clique $C \subseteq V$ permits the stronger inequality $\sum_{v \in C} x_v \leq 1$, which preserves the feasible integer solutions and tightens the linear-programming relaxation. MIS-easy follows (13) with these equivalent clique inequalities where applicable.

Combinatorial auction winner determination. Let $\mathcal{M}$ denote the auction items and $\mathcal{B}$ the submitted bids. Each bid $b \in \mathcal{B}$ offers price p_b for a bundle $Q_b \subseteq \mathcal{M}$. With $z_b = 1$ when bid b is accepted, the winner-determination problem is (Leyton-Brown et al., 2000; Sandholm, 2002)

$$\begin{aligned} \max_{z \in \{0,1\}^{|\mathcal{B}|}} \quad & \sum_{b \in \mathcal{B}} p_b z_b \\ \text{subject to} \quad & \sum_{b \in \mathcal{B}:\, i \in Q_b} z_b \leq 1, \quad i \in \mathcal{M}. \end{aligned} \tag{14}$$

The objective maximizes accepted revenue. Each item constraint allocates an item to at most one accepted bid. Dummy items encode mutually exclusive bids from the same bidder and preserve this constraint structure. CA-easy and CA-medium both use (14) with different generator regimes.

Generalized independent set, medium. GISP-medium uses the model in (10). The medium generator changes the vertex rewards, fixed conflicts, priced conflicts, and admission costs. Its decision variables, objective, and constraint structure match GISP-easy.

All three models are binary MILPs. Negating their maximization objectives gives the canonical minimization form in (1). For each workload variant, the evaluator records the CPLEX incumbent and proof-bound trajectories, computes the PDI in (4), and compares the selected configuration with the paired default measurements through (5). Trajectory-based scoring stays fixed across workloads.

K.2 Historical validation and held-out results

The historical archive contains completed held-out PDI results for five workload variants and a validation-only result for MIS-medium. Rows R0 through R8 report successive validation rounds, beginning with the initial LLM-proposed configuration at R0. The Test row evaluates the configuration selected by the historical workflow. Positive percentages indicate improvement over the CPLEX default. Blank cells mark incomplete rounds.

These tables retain the original LLM evaluation and call-counting procedure. The historical GISP-easy result is +28.9%, and its PACER remeasurement is +34.3%. For GISP-easy, Table 16 records 2,109 calls before test and 2,309 including test. Section 4.4 reports both

Round	MIS-easy	CA-easy	GISP-easy	CA-medium	GISP-medium
R0	−19.3/ − 15.7	+12.1/ + 5.1	−34.9/ − 86.7	+0.8/ + 4.7	+16.3/ + 1.4
R1		+25.8/ + 13.3	+27.3/ + 29.5	+12.8/ + 9.5	
R2	+10.6/ + 21.8	+35.5/ + 28.0	+27.9/ + 30.1		
R3	+12.4/ + 23.7	+35.0/ + 28.1	+28.4/ + 30.1	−7.9/ − 20.1	
R4	+8.5/ + 23.4	+36.0/ + 28.7	+28.4/ + 30.1	+13.8/ + 12.2	
R5	+8.3/ + 23.2	+35.3/ + 28.6	+28.2/ + 29.9	+36.1/ + 31.8	
R6	+12.7/ + 23.9	+37.5/ + 31.4	+28.4/ + 30.1	+36.1/ + 31.8	
R7	+12.5/ + 23.8	+37.2/ + 30.4	+28.5/ + 30.2		
R8	+12.4/ + 23.8	+38.0/ + 31.3	+27.5/ + 29.4		
Test	+13.2/ + 13.9	+26.3/ + 22.1	+28.9/ + 31.2	+36.0/n/a	+18.1/ + 2.1

Table 15: Historical improvement relative to the CPLEX default (%). Each cell gives PDI improvement / time improvement. R0 to R8 are validation rounds, and Test evaluates the selected configuration on held-out instances. The CA-medium test-time field is unavailable.

Round	MIS-easy	CA-easy	GISP-easy	CA-medium	GISP-medium
R0	191	205	261	133	238
R1	70	285	285	105	
R2	147	325	237	72	
R3	301	325	225	105	
R4	287	245	249	145	
R5	175	245	225	145	
R6	317	185	213	105	
R7	203	265	237		
R8	287	225	177		
Test	200	200	200	200	200

Table 16: Historical LLM solver calls by round. GISP-easy uses 2,109 calls before test, corresponding most closely to $k^\star = 21$ PACER trials.

the separately documented 2,064-call tally and the round-level sum of 2,109 calls. Both correspond most closely to $k^\star = 21$ PACER trials. The CA-medium archive ends during R7 and tests the R6 configuration. Its PDI is retained, and its wall-clock field is empty. The GISP-medium archive ends during R1 and tests the R0 configuration. The retained MIS-medium archive ends after R1 validation, which improves PDI by 9.8% and wall-clock time by 38%.

L Artifact specifications and component measurements

The artifacts in Table 3 were retained after code execution, repair, and validation selection. The workflow could generate fixed settings, code for learned instance-dependent actions, or both. Each retained artifact was fixed before test evaluation.

Fixed settings and selection provenance. The retained artifacts change different groups of global CPLEX controls. MIS-easy modifies cut generation, probing, search emphasis, diving, and symmetry handling. CA-easy modifies search emphasis, branching-variable selection, and two primal-heuristic controls, together with an instance-dependent MIP start. The later GISP-easy artifact modifies search emphasis, diving, and probing. The medium artifacts change search emphasis or the diving and branching-variable-selection policies. The controlled comparison retains the earlier GISP-easy configuration, which changes branching-variable selection and uses a backtracking tolerance of 0.05. The companion archive records the exact parameter assignments.

Instance-dependent component. MIS-easy and GISP-easy use fixed settings on all 100 instances, with PDI ratios of 0.9380 and 0.7044. CA-easy supplies a MIP start on all 100 instances. Table 17 holds its global settings fixed and removes that start. Speedup is

CA-easy artifact	PDI ratio	Time speedup
Fixed settings + learned MIP start	0.787882	1.290×
Fixed settings only	0.788030	1.282×

Table 17: CA-easy component ablation within a separately archived 100-instance execution. PDI is the configured/default ratio, and time speedup is the reciprocal time ratio.

Earlier artifact	Node ratio	PDI ratio	Wall-clock ratio
MIS-easy	0.0985	1.5347	2.1537
CA-easy	0.3478	1.8878	3.5689

Table 18: Earlier artifacts selected for node reduction. Entries are configured/default ratios. Both artifacts reduce nodes while increasing PDI and time.

the reciprocal of the geometric-mean configured/default wall-clock ratio. The relative PDI change is $100(0.788030/0.787882 - 1) \approx 0.02\%$.

Measurement provenance. The replay in Table 3 uses the 98 objective-matched pairs retained by the common evaluator. The component ablation in Table 17 uses a separately archived 100-instance execution. The replay table reports the common-evaluator result, and the ablation compares the two CA-easy component variants within its execution.

L.1 Node reduction and solve quality

Table 18 reports two earlier artifacts optimized for branch-and-bound node reduction. These differ from the retained artifacts in Table 3.