LOG-DRIVEN SEMANTIC TUNING FOR MIXED INTEGER PROGRAMMING SOLVERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Configuring mixed-integer linear programming (MILP) solvers is costly when each candidate requires a solver run. Scalar outcomes rank configurations, but execution logs also reveal where the solver spends effort. We introduce Log-Driven Semantic Tuning (LDST), in which a large language model interprets these diagnostics using parameter meanings and a bounded history of trials. The model proposes changes to the best configuration found so far, automating the interpretation and editing performed between runs. Across 215 MILP instances evaluated with SCIP, LDST reduces geometric mean best-so-far loss by 43.9% and 16.7% relative to the best black-box baseline in the easy and intermediate regimes after 50 proposals. The benchmark protocol and tuning trajectories provide a common basis for studying how semantic knowledge and execution feedback guide configuration.

1 INTRODUCTION

The performance of mixed-integer programming (MIP) solvers such as SCIP (Achterberg, 2009) depends strongly on their configuration. Automatic configuration methods search for useful settings without manual trial-by-trial adjustment (Hutter et al., 2010; 2011). Evaluating each candidate remains expensive. Configurators based on scalar outcomes, such as runtime or a derived loss (Hutter et al., 2009; Lindauer et al., 2022; López-Ibáñez et al., 2016; Schede et al., 2022), can rank candidates but have limited information about why a run behaves as it does.

Runs with similar losses at the same time limit can exhibit different behavior. One may find primal solutions early but make little dual progress. Another may spend most of its effort on strong branching linear programming (LP) work. Bound trajectories and subsystem activity suggest different changes to separation or branching effort. Parameter meanings connect these observations to the operations that a configuration controls.

We introduce Log-Driven Semantic Tuning (LDST), which automates this interpretation using a large language model (LLM). The model combines compact run diagnostics with parameter semantics and a bounded history of trials to propose changes to the best configuration found so far. The solver evaluates the resulting configuration before the next decision. The loop operates between runs without task-specific model training or changes to solver source code. The objective is fixed by the regime determined from the default run and measures completion time, accumulated primal–dual gap, or dual progress; truncated easy runs use a log-based completion estimate (Section 3).

A second contribution is a curated benchmark of 215 public mixed-integer linear programming (MILP) instances spanning four application families and a general-purpose MIP collection. Documented formulations, parameter domains, difficulty groups, and evaluation budgets provide common conditions for comparing configuration methods. Instance lists, reference scores, and tuning trajectories enable further analysis of which information helps a configurator and how improvements vary across problems.

The experiments show lower aggregate loss for LDST in the easy and intermediate regimes, and the ordering holds for every seed in a repeated study on 20 instances. Prompt ablations show the largest degradation when parameter names, groups, and defaults are hidden. Together with the comparison against outcome-only feedback, these results motivate using parameter meanings to interpret solver behavior when choosing subsequent changes (Section 5).

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.

2 Related Work

Algorithm configuration searches parameter spaces using measured performance (Schede et al., 2022), and its application to MIP predates LLMs. Hutter et al. (2010) tuned MIP solvers with a general-purpose configurator that needs no MIP-specific knowledge. SMAC fits a surrogate to configurations and their outcomes, optionally conditioned on static instance features (Hutter et al., 2011; Lindauer et al., 2022). Iterated racing in irace discards statistically inferior candidates (López-Ibáñez et al., 2016), and RBFOpt searches with radial basis function surrogates (Costa & Nannicini, 2018). ISAC tunes one configuration per feature-based cluster of training instances (Kadioglu et al., 2010), Hydra builds portfolios of complementary configurations (Xu et al., 2011), and learned methods encode instances as graphs or sets to predict instance-specific settings (Hosny & Reda, 2023; Song et al., 2023).

LLM-based optimizers draw on evaluation histories, execution feedback, and domain knowledge to different degrees. OPRO prompts an LLM with previous candidates and their values (Yang et al., 2023), AgentHPO revises hyperparameters from earlier trials (Liu et al., 2024a), and Custode et al. (2024) adapt evolution-strategy step sizes from optimization history. Trace optimizes computational workflows using execution traces and rich feedback (Cheng et al., 2024), TextGrad backpropagates textual feedback through computation graphs (Yuksekgonul et al., 2025), and Wei et al. (2025) convert raw execution output into guidance for optimizing parallel programs. Domain knowledge enters LLAMBO through LLM-based surrogate modeling and candidate sampling (Liu et al., 2024b). In database tuning, GPTuner uses manuals to select parameters and narrow their ranges before Bayesian optimization (Lao et al., 2024), and λ-Tune generates complete configurations from the tuning context (Giannakouris & Trummer, 2025). LDST combines parameter semantics with diagnostics from recent solver runs at each tuning step.

MIP methods that act between runs differ in what they observe and modify. Lawless et al. (2025) prompt an LLM with descriptions of cutting-plane separators to generate configurations and select a portfolio. GRIMIP (Luo et al., 2026) uses a short Gurobi exploration run to extract static and dynamic features and choose a subset of parameters. Its iterative search predicts performance means and uncertainties from histories of complete configurations and scalar scores, and selects candidates by expected improvement. FormuEvo (Yuan et al., 2026) evolves executable programs encoding the mathematical formulation, using solver statistics and a structured memory to guide mutation, crossover, and repair. Compared with GRIMIP, LDST obtains new diagnostics after every evaluated configuration, including non-improving ones, and edits the incumbent. Unlike FormuEvo, it keeps the formulation fixed and changes only solver parameters.

Learned solver-control policies choose branching variables (Gasse et al., 2019; Gupta et al., 2020) and select individual cuts (Tang et al., 2020). At a coarser level, Li et al. (2023) predict instance-aware separator configurations. An LLM can also schedule branching rules from structured metrics of the recent tree state (Anonymous, 2025). LDST leaves the solver's branching and cut callbacks unchanged. Progress estimation is relevant because the evaluation objective of LDST includes a completion estimate. Hendel et al. (2022) estimate branch-and-bound tree size by combining signals such as gap, tree weight, and leaf frequency, and their most accurate variants use richer internal measures and trained predictors. LDST instead fits a small tree-growth model to progress reported in the ordinary solver log, motivated by the single-variable branching model of Le Bodic & Nemhauser (2017).

3 Problem Formulation

Configuration quality depends on the progress observable within a fixed solver time limit. A proof of optimality or infeasibility provides a completion time; feasible but unproved runs expose primal and dual progress; runs without a primal solution provide only dual information. We therefore select the objective for each instance from its default-run outcome and keep that choice fixed throughout tuning.

Table 1: Fixed default regimes and their tuning signals.

Case	Regime	Default outcome at τ	Signal
1	Easy	Optimality or infeasibility proved	Completion time
2	Intermediate	Finite primal, no proof	Primal–dual integral
3	Hard	No primal, finite dual	Dual improvement

3.1 Instance Classification and Loss Functions

For instance i, configuration p is evaluated with time limit $\tau = 600\,\mathrm{s}$ and produces an execution log $\Lambda_i(p)$. A reference run with the default configuration p^{def} fixes $c_i \in \{1, 2, 3\}$ according to Table 1. These labels describe reference difficulty. A tuned configuration remains subject to the same objective even when its termination status changes.

Let $z_i^{\mathrm{pr}}(t; p)$ and $z_i^{\mathrm{du}}(t; p)$ denote primal and dual bounds. Eq. (1) defines the raw score $q_i(p)$ and positive loss $L_i(p)$, where $s_i = -1$ for minimization and $s_i = +1$ for maximization. The completion signal $\widehat{T}_{\infty,i}(p)$ is defined in Section 3.2.

$$q_i(p) = \begin{cases} \dfrac{\widehat{T}_{\infty,i}(p)}{\widehat{T}_{\infty,i}(p^{\mathrm{def}})}, & c_i = 1, \\ \dfrac{\mathrm{PDI}_i^{\mathrm{SCIP}}(p)}{\mathrm{PDI}_i^{\mathrm{SCIP}}(p^{\mathrm{def}})}, & c_i = 2, \\ 1 + s_i \dfrac{z_i^{\mathrm{du}}(\tau; p) - z_i^{\mathrm{du}}(\tau; p^{\mathrm{def}})}{|z_i^{\mathrm{du}}(\tau; p^{\mathrm{def}})| + 10^{-6}}, & c_i = 3, \end{cases} \qquad (1)$$

$$L_i(p) = \max\{q_i(p), \delta\}, \qquad \delta = 10^{-12}.$$

Default denominators in Cases 1 and 2 are validated as positive and finite. Default loss is one, with smaller values preferred. The floor permits geometric aggregation even when a raw dual-improvement score is nonpositive. Invalid or nonfinite evaluations cannot become the best observed configuration.

Case 2 uses the `Integrals/primal-dual` value reported in the final solver log by SCIP (Berthold, 2013). The primal–dual integral (PDI) integrates the percentage gap over time, with the instantaneous gap capped at 100% and SCIP conventions for missing or opposite-sign bounds.

For a tuning set $\mathcal{I}_{\mathrm{T}}$, the objective is $L(p; \mathcal{I}_{\mathrm{T}}) = \exp\big(|\mathcal{I}_{\mathrm{T}}|^{-1} \sum_{i \in \mathcal{I}_{\mathrm{T}}} \log L_i(p)\big)$. The per-instance experiments use a singleton tuning set. Multi-instance tuning aggregates over the training instances. Each method normalizes against its own initial evaluation of the same default configuration, so that the default loss is one and improvements are expressed multiplicatively across runtime and objective scales. Every method uses this evaluation objective.

3.2 Completion Time Estimation from Solver Logs

Truncation can hide differences in remaining proof effort between configurations with the same elapsed runtime. For Case 1, the signal returns observed solve time $T_i(p)$ when optimality or infeasibility is proved; otherwise it extrapolates tree growth from the available log. At endpoint $\tau' = \min\{\tau, t_{\max}\}$, where $t_{\max}$ is the last logged time, snapshots contain elapsed time t_j, processed nodes n_j, and open nodes u_j. The primal bound available at the cutoff defines $g_j = |z_i^{\mathrm{pr}}(\tau'; p) - z_i^{\mathrm{du}}(t_j; p)|$. Processed nodes represent completed work. Open nodes and bound separation describe the remaining search.

For a growth factor $x > 1$, the surrogate $\widehat{N}_j(x) = n_j + u_j x^{g_j}$ estimates terminal tree size (Le Bodic & Nemhauser, 2017). A log-scale fit yields a common size $b^\star$. The recent time per node θ gives the raw prediction $\widetilde{T}_{\infty,i}(p) = \tau' + \theta \max\{b^\star - n_{\mathrm{last}}, 0\}$. The effective signal caps finite predictions at 10τ and assigns the same cap when fitting fails, before normalization. Appendix G specifies the fit and validation protocols. Retrospective prediction error on 100 completed default runs decreases as more of each run is observed (Table 18). Using only observations available at the cutoff, the

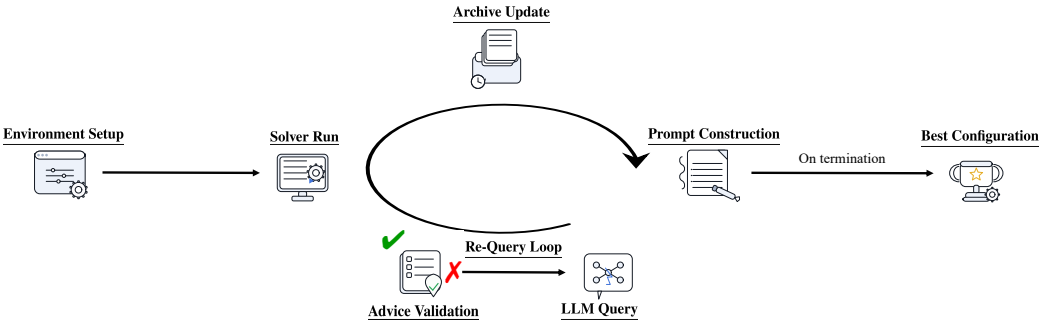

Figure 1: The LDST tuning loop. Each evaluated configuration produces logs that update the trial history for the next prompt. The loop returns the best evaluated configuration.

estimate correctly orders 26 of 34 configuration pairs within instances (Table 19). Validation on five completed longer runs gives a symmetric mean absolute percentage error of 0.214 (Appendix G.5).

4 Log-Driven Semantic Tuning

In each trial, LDST uses an LLM to interpret run diagnostics and propose edits to the current best configuration, guided by parameter semantics. Domain knowledge is provided by the set of tunable parameters $\mathcal{W}$ with their types and domains, together with the diagnostic summaries, diagnostic guide, objectives, and validation rules. The loop uses the tuning set $\mathcal{I}_{\mathrm{T}}$ and aggregate loss $L(p; \mathcal{I}_{\mathrm{T}})$ defined in Section 3, with lower values preferred.

4.1 The Tuning Loop

Figure 1 summarizes the sequential loop. The default configuration p^{def} is evaluated first to initialize the best observed configuration $p^{\star}$. At trial k, configuration $p^{(k)}$ is run on every instance in $\mathcal{I}_{\mathrm{T}}$ with per-instance time limit τ, yielding $L^{(k)} := L(p^{(k)}; \mathcal{I}_{\mathrm{T}})$ and logs $\Lambda^{(k)}$. The result is added to the trial history. For prompt construction, $p^{\star}$ is the configuration associated with the minimum loss in the trial history; exact ties favor the most recently evaluated configuration. The next proposal is always constructed relative to $p^{\star}$, including after a non-improving trial.

Each response undergoes deterministic checks of the required format, membership in the set of tunable parameters, and parameter types and domains. The edit list is truncated to the cap M, and the resulting configuration must not have been evaluated previously and must load successfully as solver settings. Invalid proposals receive validation feedback for another query when attempts remain. The loop stops after K evaluated proposals beyond the default or when all V proposal attempts (including the initial query) fail to produce a valid novel edit, returning the best observed configuration. Invalid proposals do not consume solver evaluations.

4.2 Diagnostic State and Prompting

Diagnostic summary. The observation from trial k is represented as $r^{(k)} = \big(L^{(k)}, p^{(k)}, \mathbf{d}^{(k)}\big)$, where $\mathbf{d}^{(k)} = \mathrm{Parse}\big(\Lambda^{(k)}\big)$. The fixed set of diagnostics covers search scale, progress of primal solutions and bounds, and activity in presolve, root relaxation, strong branching, separation, and major heuristics. For example, strong branching time and the number of LP iterations describe computational effort. Changes in the root bound summarize progress during root processing. The diagnostic quantities are computed in the same way across runs and shown in aligned rows when tuning multiple instances. Appendix E defines the observables.

Memory and edit representation. The prompt includes a bounded view of the trial history consisting of the B best and R most recent trials, with duplicates removed. The best trials preserve evidence from promising configurations. Recent trials retain the outcomes of subsequent attempts, including unsuccessful changes. The incumbent is shown explicitly, and every other configuration

Table 2: Results for the full benchmark at trial 50, with the default configuration eligible for selection. Coverage reports hard instances with improved dual bounds. Bold indicates the best value in each column.

	Primary loss ↓			Dual coverage (%) ↑
Method	Easy	Intermediate	Hard	
LDST	**0.274**	**0.557**	0.993	**89.6**
SMAC3	0.499	0.668	0.993	**89.6**
RBFOPT	0.490	0.675	**0.988**	85.4
RANDOM	0.519	0.669	0.993	87.5

p in this view is encoded relative to it as $\Delta(p\,;\,p^\star) := \{(j, p_j) : j \in \mathcal{W},\ p_j \neq p^\star_j\}$. The proposed action uses the same representation, listing indices of tunable parameters and replacement values of the required type. Only listed coordinates change, and all others inherit their incumbent values. The edit cap bounds the number of changed parameters, not the magnitude of numerical changes.

Parameter names, defaults, and domains, a diagnostic guide, and optional formulation text for the problem family accompany this memory. The prompt asks the model to identify a dominant bottleneck and provide a brief rationale linking its edits to observed diagnostics. A separate list identifies previously tried configurations not included in this view; novelty checks consider all evaluated configurations. Appendices D and B give the settings of B, R, M, and V. Appendix F provides prompt examples, and Appendix F.1 reproduces the diagnostic guides in full.

5 EXPERIMENTS

We evaluate configuration quality across the full benchmark, then examine repeated tuning, the information guiding proposed changes, and sensitivity to the model and configuration space.

5.1 EXPERIMENTAL PROTOCOL

We use SCIP 9.2.4 with one thread and a per-evaluation limit of $\tau = 600\,\mathrm{s}$. The baselines are SMAC3 (Lindauer et al., 2022), RBFOPT (Costa & Nannicini, 2018), and RANDOM. The reference LDST configuration uses GPT-5.2 with medium reasoning effort and an edit cap of $M = 20$. Unless otherwise stated, studies use seed 42, the curated set of 45 tunable parameters, and 50 proposals after the default evaluation. All methods share the parameter domains and regime-adaptive loss from Section 3. The scalar methods receive evaluated configurations and losses without parsed diagnostics. Appendices B, C, and D give the baseline settings, parameter domains, and LLM settings.

The default remains eligible throughout tuning. Best-so-far selection is $b_i(t) = \min_{0 \le u \le t} L_i(p_i^{(u)})$, where $p_i^{(0)} = p^{\mathrm{def}}$. Reported losses are geometric means of $b_i(t)$ over the specified instances, with lower values preferred.

The benchmark contains 215 instances from capacitated vehicle routing (CVRP) (Lima et al., 2014), generalized assignment (GAP) and job-shop scheduling (JSP) (Beasley, 1990), uncapacitated facility location (UFL) (Hoefer, 2018), and MIPLIB (Gleixner et al., 2021), partitioned into 76 easy (Case 1), 91 intermediate (Case 2), and 48 hard (Case 3) instances. We provide instance lists, formulations, and environment details in Appendices A and M.

5.2 TUNING INDIVIDUAL INSTANCES

Table 2 shows that LDST achieves the lowest aggregate primary loss in the easy and intermediate regimes, reducing loss by 43.9% and 16.7% relative to the respective best scalar baseline. Comparisons cover all 76 easy, 91 intermediate, and 48 hard instances at the same 50-proposal checkpoint.

Table 3: Auxiliary summaries of configurations selected at trial 50 on subsets common to all methods. The default configuration is eligible for selection. Shifted geometric means use shifts of 1 s for time and 0.01 percentage points for gap.

Method	Easy time (s), $n = 75$	Intermediate gap (%), $n = 90$
LDST	**10.5**	**4.88**
SMAC3	17.3	10.29
RBFOPT	17.1	10.77
RANDOM	17.9	10.01

Table 4: Five-seed paired comparisons over 100 instance-seed pairs. The geometric mean loss of LDST is 0.371; ratios compare LDST with each baseline.

Baseline	Loss ↓	Ratio [95% interval]	LDST wins
SMAC3	0.523	0.710 [0.641, 0.786]	88/100
RBFOPT	0.519	0.716 [0.649, 0.788]	85/100
RANDOM	0.532	0.698 [0.628, 0.775]	85/100

Configurations selected by the primary objective also give LDST the lowest shifted geometric means of observed solve time and final gap (Table 3). Using the same runs, we evaluate time on the 75 easy instances solved to optimality by all methods and gap on the 90 intermediate instances with finite final gaps under all methods.

In the hard regime, RBFOPT has the lowest aggregate loss, 0.988 versus 0.993 for LDST. LDST matches SMAC3 in improving dual bounds on 43 of 48 instances. Coverage counts instances with $b_i(50) < 1 - 10^{-12}$ equally, whereas aggregate loss also reflects improvement magnitude. The full-set lead of RBFOPT depends on its large improvement on `irish-electricity`, where its loss is 0.698 versus 0.953 for LDST. A sensitivity check omitting that instance gives LDST the lowest aggregate loss over the remaining 47 instances. The primary comparison retains all 48 instances. Appendix L gives the sensitivity results, auxiliary definitions, and full-suite curves (Figure 8).

5.3 ROBUSTNESS ACROSS FIVE SEEDS

Performance variability in MIP (Lodi & Tramontani, 2013) motivates repeating adaptive search under several seeds. We use 20 prespecified instances, 12 easy and eight intermediate, from the five benchmark sets. Seeds $\{17, 29, 42, 71, 97\}$ yield 400 tuning trajectories across the four methods, each completing 50 proposals beyond trial 0. Table 4 reports terminal losses and paired LDST/baseline ratios over 100 matched instance-seed pairs. All three hierarchical paired bootstrap 95% intervals lie below one, and LDST has the lowest aggregate loss under every seed. After averaging over seeds, LDST has lower loss on 18, 17, and 18 of the 20 instances against SMAC3, RBFOPT, and RANDOM, respectively. Every aggregate LDST/baseline ratio also remains below one with any single instance removed (Appendix H).

Figure 2 shows best-so-far curves for the easy and intermediate instances, with pointwise hierarchical bootstrap intervals. Appendix H gives the instance list, seed-level results, and resampling protocol. We also compare standard LDST with Restart10, which starts a new branch after ten consecutive proposals fail to improve the global incumbent, preserving the global best. Across 39 matched pairs at the same 50-proposal budget, standard LDST has lower observed aggregate loss, 0.550 versus 0.576 (Appendix H.3).

5.4 PROMPT ABLATIONS

We compare full LDST with three LLM prompt variants on the 76 easy instances, using 30 proposals and the same set of tunable parameters, model, edit cap, and validation rules. STATICCONTEXT retains diagnostics for the current best configuration and parameter semantics but removes the history of earlier trials, the diagnostic guide, and the list of previously tried configurations shown in the prompt. OUTCOMEONLY retains parameter semantics and outcome history, including loss, solve

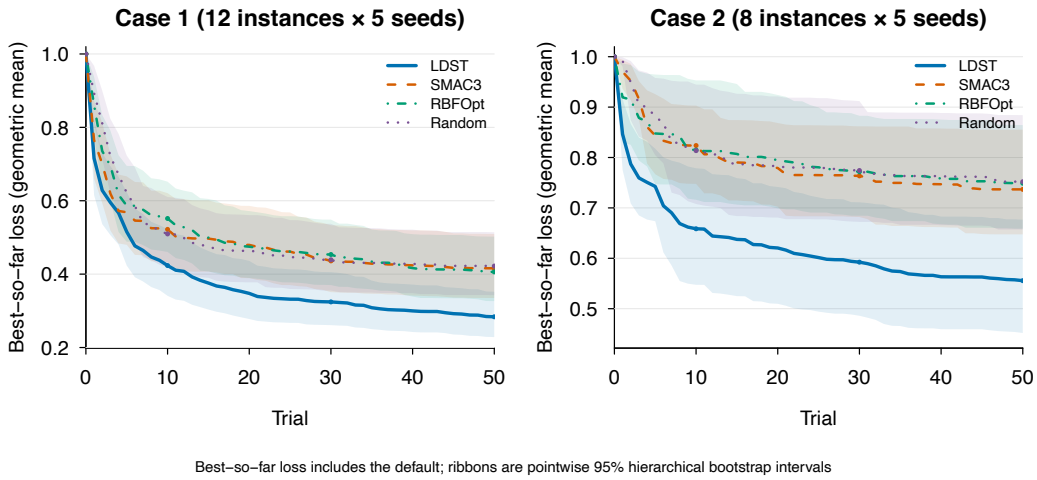

Figure 2: Convergence on 12 easy and eight intermediate instances across five seeds. Curves show geometric mean best-so-far losses including the default configuration; shaded bands are pointwise 95% intervals from 5,000 hierarchical bootstrap replicates, resampling instances within family strata and seeds within sampled instances while preserving method pairing.

Table 5: LLM prompt variants on 76 easy instances with a 30-proposal budget. Losses are geometric means with the default configuration eligible for selection. Early terminations retain their best observed values.

Variant	Final loss ↓	Trial-5 loss ↓	$H_{0.8}$ (%) ↑
Full (LDST)	**0.308**	**0.485**	**90.8**
StaticContext	0.342	0.509	86.8
OutcomeOnly	0.358	0.520	84.2
IndexOnly	0.523	0.707	76.3

time, final gap, and finite/feasible status, but removes process diagnostics and their guide. IndexOnly retains the trial history with diagnostics, the guide, and the list of previously tried configurations, replacing parameter names, groups, and defaults with indices and domain hints. Each variant changes several related inputs, so we assess their combined role. Appendix I gives the exact prompt definitions and termination details.

Full LDST has the lowest final and trial-5 losses and the highest $H_{0.8}$, the percentage of instances reaching $b_i(30) \leq 0.8$ (Table 5). Restricting parameter semantics produces the largest degradation. Against OutcomeOnly, full LDST achieves a paired geometric mean loss ratio of 0.860, with a 95% instance-bootstrap interval of $[0.753, 0.976]$. It wins on 49 instances and loses on 27 (exact two-sided sign test, $p = 0.0154$). Both variants complete all proposals. The terminal comparison favors supplementing outcome history with process diagnostics and the guide.

Early termination affects one StaticContext trajectory at trial 26 and six IndexOnly trajectories. We carry forward the best losses of all seven trajectories to the nominal budget. At trial 15, the full/IndexOnly ratio is 0.597, with a 95% interval of $[0.524, 0.674]$, so the difference precedes any early termination. The full/OutcomeOnly intervals include one at trials 5, 10, and 15 and exclude it at trial 30. Figure 3 shows the paired checkpoint contrasts.

All methods use quantities derived from solver logs in the common objective. The ablations suggest that using parameter semantics to connect observed effort to solver operations helps the model propose useful changes.

Figure 7 illustrates this connection in a generated rationale. The model associates substantial strong branching work on a small tree with a proposal to tighten strong branching LP iteration limits, weighing lower branching cost against the risk of more nodes. Of the 10,750 proposals in the main

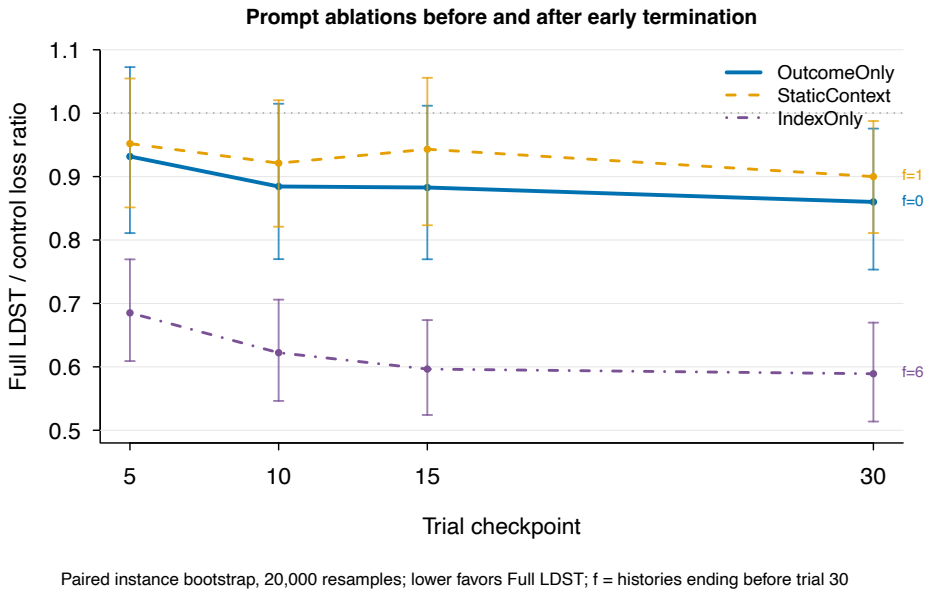

Figure 3: Geometric means of paired full LDST/control best-so-far loss ratios on 76 easy instances; lower values favor full LDST. Error bars show pointwise 95% intervals from 20,000 paired instance-bootstrap resamples. Best losses are carried forward after early termination (one STATICCONTEXT and six INDEXONLY trajectories).

study, 76.7% change parameters controlling cut separation; parameters controlling branching and node selection change in 56.7% and 60.8% of hard-instance proposals, respectively (Appendix H.4).

5.5 MODEL, PARAMETER, AND COST SENSITIVITY

Table 6(a) compares GPT-5.2 and GPT-5-mini on eight prespecified intermediate instances from four benchmark sets, using seed 42 and 50 proposals. We select the subset by difficulty under the default run within each family before comparing models. Every trajectory completes its proposal budget. Both LDST variants achieve lower loss than all scalar baselines, with GPT-5.2 performing best.

In panel (b), we expand the set of tunable parameters from 45 to 90 parameters on ten instances, split evenly between easy and intermediate regimes, with seed 42. We select the additional 45 parameters mechanically and independently of outcomes, nine each from branching, node selection, separation, presolving, and primal heuristics. Existing parameter domains and the proposal budget remain unchanged, and all methods use the same space within each comparison. The loss of LDST increases by only 0.081%. All scalar baselines have higher loss in the expanded space. The advantage of LDST therefore persists with broader configuration choices and the same number of solver evaluations.

Panel (c) reports costs for the same eight trajectories per model as panel (a), including additional model queries following validation. GPT-5-mini achieves lower aggregate loss than scalar tuning, with an 82.0% lower median model cost than GPT-5.2 at common uncached list rates. In a separate study with repeated model queries, 99.0% of GPT-5.2 responses and 97.4% of GPT-5-mini responses satisfy the response requirements on the first attempt. Appendix J gives these requirements, study protocols, and cost definitions.

5.6 SOLVER PORTABILITY AND TRANSFER WITHIN PROBLEM FAMILIES

A preliminary CBC study on eight instances uses a diagnostic guide derived from configurations produced by the scalar baselines on those instances. With the default configuration eligible and 600 s PDI normalized by its default value, LDST achieves geometric mean losses of 0.699 at trial 10

Table 6: Scoped sensitivity studies. Performance panels use seed 42 and 50 proposals; model names in panel (a) denote LDST variants. The cost panel uses the same eight complete trajectories per model as panel (a).

(a) Model choice: eight intermediate instances; loss ↓

GPT-5.2	GPT-5-mini	SMAC3	RBFOPT	RANDOM
0.591	0.652	0.771	0.776	0.790

(b) Parameter space: five easy and five intermediate instances; loss ↓

Space	LDST	SMAC3	RBFOPT	RANDOM
Curated-45	**0.462**	0.580	0.593	0.559
Expanded-90	**0.462**	0.749	0.743	0.770

(c) Matched model cost: eight complete trajectories per model

Model	Complete trajectories	Median API calls	Median uncached list cost (USD)
GPT-5.2	8	51.0	1.925
GPT-5-mini	8	53.5	0.346

and 0.662 at trial 20, improving on the best scalar baseline by 6.49% and 2.47%, respectively. The solver interface maps CBC diagnostics to parameter edits that respect the parameter types. Fully local Qwen3-8B execution with `Q4_K_M` quantization and reasoning disabled also produces useful early edits without using a hosted model service. All 51 model requests return responses. Of 24 validated proposals, 22 pass on the first attempt, and four of eight instances improve. Appendix K gives both protocols and the resulting trajectories.

The same solver interface can be used to tune a new instance from its own execution feedback. A selected configuration can instead be applied unchanged, avoiding another tuning loop. In Appendix L.2, we evaluate single-anchor reuse and joint tuning versus independent tuning followed by selection. Transfer quality and the relative performance of the two strategies vary across families. For single-anchor reuse, LDST achieves lower aggregate test loss than all three scalar baselines, both overall and within each of the four benchmark families.

6 LIMITATIONS

We use parameter spaces and diagnostic guides defined for each solver (Appendix K). Our comparisons use equal numbers of solver evaluations, and total tuning time also includes LLM response time. The completion estimate is intended for runs close to completion (Appendix G). Further research could extend the evaluation to additional problem classes and solvers and examine completion estimates at earlier stages of the search.

7 CONCLUSION

LDST uses parameter semantics to connect solver diagnostics and the final score to concrete configuration changes. The loop automates the resulting updates between successive runs and improves configuration quality in the tested easy and intermediate regimes under a shared evaluation budget. The repeated experiments establish that the observed aggregate advantage persists across the evaluated seeds and is not driven by a single instance. Prompt ablations favor combining parameter meanings with process diagnostics and their guide over outcome history alone. The curated benchmark provides common instances, objectives, and evaluation conditions, with tuning trajectories for studying feedback design and performance across problem families and difficulty regimes.

AI-USE STATEMENT

Large language models are the core proposal mechanism of LDST; their role, settings, prompts, and validation procedures are documented in Section 4 and the appendices. Claude, ChatGPT Pro, and Codex also assisted with editorial planning, manuscript drafting and revision, methodology feedback, interpretation of results, and development of analysis and reconstruction code. The authors retain responsibility for verifying technical claims, experimental results, analysis code, and references, and for the accuracy and integrity of the final manuscript.

REPRODUCIBILITY STATEMENT

The appendices specify solver interfaces, objectives, prompts, the curated set of tunable parameters with their types and domains, baseline settings, instance sets, seed controls, and evaluation budgets. The anonymous analysis supplement provides numerical results for each tuning trajectory, code for checking these results, and data used to produce the principal figures. Precomputed bootstrap intervals are included. The package excludes the full execution code, logs, and benchmark files; it supports reconstruction of the reported summaries, not new solver or model runs.

REFERENCES

Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, July 2009. ISSN 1867-2957. doi: 10.1007/s12532-008-0001-1. URL https://doi.org/10.1007/s12532-008-0001-1.

Joseph Adams, Egon Balas, and Daniel Zawack. The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, 34(3):391–401, March 1988. ISSN 0025-1909. doi: 10.1287/mnsc.34.3.391. URL https://pubsonline.informs.org/doi/abs/10.1287/mnsc.34.3.391.

Anonymous. Large Language Model Guided Dynamic Branching Rule Scheduling in Branch-and-Bound, October 2025. URL https://openreview.net/forum?id=8LCdjf7uIk.

David Applegate and William Cook. A Computational Study of the Job-Shop Scheduling Problem. *ORSA Journal on Computing*, 3(2):149–156, May 1991. ISSN 0899-1499. doi: 10.1287/ijoc.3.2.149. URL https://pubsonline.informs.org/doi/abs/10.1287/ijoc.3.2.149.

P. Augerat, D. Naddef, J. M. Belenguer, E. Benavent, A. Corberan, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem, September 1995. URL https://www.osti.gov/etdeweb/biblio/289002.

R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi. An Exact Algorithm for the Capacitated Vehicle Routing Problem Based on a Two-Commodity Network Flow Formulation. *Operations Research*, 52(5):723–738, October 2004. ISSN 0030-364X. doi: 10.1287/opre.1040.0111. URL https://pubsonline.informs.org/doi/abs/10.1287/opre.1040.0111.

J. E. Beasley. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, 41(11):1069–1072, November 1990. ISSN 0160-5682. doi: 10.1057/jors.1990.166. URL https://doi.org/10.1057/jors.1990.166.

Timo Berthold. Measuring the impact of primal heuristics. *Operations Research Letters*, 41(6):611–614, November 2013. ISSN 01676377. doi: 10.1016/j.orl.2013.08.007. URL https://linkinghub.elsevier.com/retrieve/pii/S0167637713001181.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next AutoDiff: Generative optimization with rich feedback, execution traces, and LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 71596–71642, 2024. doi: 10.52202/079017-2287.

N. Christofides and S. Eilon. An Algorithm for the Vehicle-dispatching Problem. *Journal of the Operational Research Society*, 20(3):309–318, September 1969. ISSN 0160-5682. doi: 10.1057/jors.1969.75. URL https://doi.org/10.1057/jors.1969.75.

P. C. Chu and J. E. Beasley. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24(1):17–23, January 1997. ISSN 0305-0548. doi: 10.1016/S0305-0548(96)00032-9. URL https://www.sciencedirect.com/science/article/pii/S0305054896000329.

Alberto Costa and Giacomo Nannicini. RBFOpt: An open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10(4):597–629, December 2018. ISSN 1867-2957. doi: 10.1007/s12532-018-0144-7. URL https://doi.org/10.1007/s12532-018-0144-7.

Leonardo Lucio Custode, Fabio Caraffini, Anil Yaman, and Giovanni Iacca. An investigation on the use of large language models for hyperparameter tuning in evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1838–1845. ACM, 2024. doi: 10.1145/3638530.3664163.

H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. In John F. Muth and Gerald L. Thompson (eds.), *Industrial Scheduling*, pp. 225–251. Prentice-Hall, Englewood Cliffs, NJ, 1963.

Marshall L. Fisher. Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees. *Operations Research*, 42(4):626–642, August 1994. ISSN 0030-364X. doi: 10.1287/opre.42.4.626. URL https://pubsonline.informs.org/doi/abs/10.1287/opre.42.4.626.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact Combinatorial Optimization with Graph Convolutional Neural Networks, October 2019. URL http://arxiv.org/abs/1906.01629.

Victor Giannakouris and Immanuel Trummer. λ-Tune: Harnessing large language models for automated database system tuning. *Proceedings of the ACM on Management of Data*, 3(1):1–26, 2025. doi: 10.1145/3709652.

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, September 2021. ISSN 1867-2957. doi: 10.1007/s12532-020-00194-3. URL https://doi.org/10.1007/s12532-020-00194-3.

Prateek Gupta, Maxime Gasse, Elias B. Khalil, M. Pawan Kumar, Andrea Lodi, and Yoshua Bengio. Hybrid Models for Learning to Branch, October 2020. URL http://arxiv.org/abs/2006.15212.

Gregor Hendel, Daniel Anderson, Pierre Le Bodic, and Marc E. Pfetsch. Estimating the Size of Branch-and-Bound Trees. *INFORMS Journal on Computing*, 34(2):934–952, March 2022. ISSN 1091-9856, 1526-5528. doi: 10.1287/ijoc.2021.1103. URL https://pubsonline.informs.org/doi/10.1287/ijoc.2021.1103.

Martin Hoefer. UflLib: Library of (uncapacitated) facility location instances. Max-Planck-Institut für Informatik benchmark repository, 2018. URL https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/.

Abdelrahman Hosny and Sherief Reda. Automatic MILP Solver Configuration By Learning Problem Similarities, July 2023. URL http://arxiv.org/abs/2307.00670.

F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stuetzle. ParamILS: An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009. ISSN 1076-9757. doi: 10.1613/jair.2861. URL https://www.jair.org/index.php/jair/article/view/10628.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Automated Configuration of Mixed Integer Programming Solvers. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard

Weikum, Andrea Lodi, Michela Milano, and Paolo Toth (eds.), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140, pp. 186–202. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-13519-4 978-3-642-13520-0. doi: 10.1007/978-3-642-13520-0_23. URL http://link.springer.com/10.1007/978-3-642-13520-0_23.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In Carlos A. Coello Coello (ed.), *Learning and Intelligent Optimization*, pp. 507–523, Berlin, Heidelberg, 2011. Springer. ISBN 978-3-642-25566-3. doi: 10.1007/978-3-642-25566-3_40.

Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC – Instance-Specific Algorithm Configuration. In *ECAI 2010*, pp. 751–756. IOS Press, 2010. doi: 10.3233/978-1-60750-606-5-751. URL https://ebooks.iospress.nl/doi/10.3233/978-1-60750-606-5-751.

M. G. KENDALL. A NEW MEASURE OF RANK CORRELATION. *Biometrika*, 30(1-2):81–93, June 1938. ISSN 0006-3444. doi: 10.1093/biomet/30.1-2.81. URL https://doi.org/10.1093/biomet/30.1-2.81.

Alfred A. Kuehn and Michael J. Hamburger. A Heuristic Program for Locating Warehouses. *Management Science*, 9(4):643–666, July 1963. ISSN 0025-1909. doi: 10.1287/mnsc.9.4.643. URL https://pubsonline.informs.org/doi/abs/10.1287/mnsc.9.4.643.

Jiale Lao, Yibo Wang, Yufei Li, Jianping Wang, Yunjia Zhang, Zhiyuan Cheng, Wanghu Chen, Mingjie Tang, and Jianguo Wang. GPTuner: A manual-reading database tuning system via GPT-guided bayesian optimization. *Proceedings of the VLDB Endowment*, 17(8):1939–1952, 2024. doi: 10.14778/3659437.3659449.

Connor Lawless, Yingxi Li, Anders Wikum, Madeleine Udell, and Ellen Vitercik. LLMs for Cold-Start Cutting Plane Separator Configuration, September 2025. URL http://arxiv.org/abs/2412.12038.

S. Lawrence. Supplement to resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Technical Report (Supplement), Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA, 1984.

Pierre Le Bodic and George Nemhauser. An abstract model for branching and its application to mixed integer programming. *Mathematical Programming*, 166(1):369–405, November 2017. ISSN 1436-4646. doi: 10.1007/s10107-016-1101-8. URL https://doi.org/10.1007/s10107-016-1101-8.

Sirui Li, Wenbin Ouyang, Max Paulus, and Cathy Wu. Learning to configure separators in branch-and-cut. In *Advances in Neural Information Processing Systems*, volume 36, pp. 60021–60034, 2023. doi: 10.52202/075280-2622.

Ivan Lima, Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. CVRPLib: Capacitated Vehicle Routing Problem Library. Website (Galgos / DI PUC-Rio and collaborators), 2014. URL https://galgos.inf.puc-rio.br/cvrplib/.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhopf, René Sass, and Frank Hutter. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization, February 2022. URL http://arxiv.org/abs/2109.09831.

Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization, 2024a. URL https://arxiv.org/abs/2402.01881.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large Language Models to Enhance Bayesian Optimization, March 2024b. URL http://arxiv.org/abs/2402.03921.

Andrea Lodi and Andrea Tramontani. Performance variability in mixed-integer programming. In Huseyin Topaloglu (ed.), *Theory Driven by Influential Applications*, pp. 1–12. INFORMS, 2013. doi: 10.1287/educ.2013.0112.

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. ISSN 22147160. doi: 10.1016/j.orp.2016.09.002. URL https://linkinghub.elsevier.com/retrieve/pii/S2214716015300270.

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, June 2026. URL https://arxiv.org/abs/2606.23299.

Alan S. Manne. On the Job-Shop Scheduling Problem. *Operations Research*, 8(2):219–223, April 1960. ISSN 0030-364X. doi: 10.1287/opre.8.2.219. URL https://pubsonline.informs.org/doi/abs/10.1287/opre.8.2.219.

G. Terry Ross and Richard M. Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, 8(1):91–103, December 1975. ISSN 1436-4646. doi: 10.1007/BF01580430. URL https://doi.org/10.1007/BF01580430.

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, October 2022. ISSN 1076-9757. doi: 10.1613/jair.1.13676. URL https://jair.org/index.php/jair/article/view/13676.

Wen Song, Yi Liu, Zhiguang Cao, Yaoxin Wu, and Qiqiang Li. Instance-specific algorithm configuration via unsupervised deep graph clustering. *Engineering Applications of Artificial Intelligence*, 125:106740, October 2023. ISSN 0952-1976. doi: 10.1016/j.engappai.2023.106740. URL https://www.sciencedirect.com/science/article/pii/S0952197623009247.

Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement Learning for Integer Programming: Learning to Cut, July 2020. URL http://arxiv.org/abs/1906.04859.

Anjiang Wei, Allen Nie, Thiago S. F. X. Teixeira, Rohan Yadav, Wonchan Lee, Ke Wang, and Alex Aiken. Improving parallel program performance with LLM optimizers via agent-system interfaces. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66155–66177. PMLR, 2025. URL https://proceedings.mlr.press/v267/wei25j.html.

Lin Xu, F. Hutter, H. Hoos, and Kevin Leyton-Brown. Hydra-MIP : Automated Algorithm Configuration and Selection for Mixed Integer Programming, 2011. URL https://www.semanticscholar.org/paper/Hydra-MIP-%3A-Automated-Algorithm-Configuration-and-Xu-Hutter/302f936148bb19527646b523456d36ff7afbd505.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers, 2023. URL https://arxiv.org/abs/2309.03409.

Haofeng Yuan, Jianing Peng, Jieyi Bi, Ni Zhang, Shiji Song, and Zhiguang Cao. FormuEvo: LLM-guided evolution for discovering solver-efficient mixed-integer programming formulations, August 2026. URL https://arxiv.org/abs/2608.23353. To appear in the Proceedings of EMNLP 2026.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4.

APPENDIX ROADMAP

Appendix A documents benchmark construction, regime assignment, and transfer splits. Appendices B, C, and D specify the shared protocol, parameter space, and model settings; Appendices E and F document diagnostics and prompts. Appendix G gives the completion estimator and cutoff-only validation. The five-seed, prompt-ablation, model/cost, and portability studies appear in Appendices H–K. Appendix L contains full-suite solver summaries, transfer results, and the hard-regime comparison. Appendix M describes the anonymous analysis supplement, environment, and accounting, and Appendix N gives the full instance lists and formulations.

A BENCHMARK SOURCES, FORMULATIONS, AND REGIME ASSIGNMENT

We document the source collections, MPS construction protocol, default-based regime assignment, and within-family split construction used in the experiments. Full family-specific formulations and complete benchmark instance lists are given in Appendix N.

We draw instances from the public collections CVRPLIB CVRP (Lima et al., 2014), OR-Library GAP and JSP (Beasley, 1990), UflLib UFL (Hoefer, 2018), and MIPLIB 2017 (Gleixner et al., 2021). The final benchmark contains 215 instances, comprising 90 CVRP, 12 GAP, 24 JSP, 30 UFL, and 59 MIPLIB instances. For the four domain-specific families, we export standard mathematical formulations to `.mps` using an automated procedure; for each generated file we verify the numbers of variables and constraints, integrality patterns, and agreement of the objective with the original model within numerical tolerances. For MIPLIB, the initial fixed-seed draw contains 60 instances from the official `benchmark-v2` subset; we remove `k1mushroom` because it does not complete presolving under default SCIP settings within the 600-second time limit, leaving 59 MIPLIB instances for the reported experiments. We provide the final instance lists, sources of the instance files, and regime labels.

A.1 REGIME ASSIGNMENT UNDER DEFAULT SETTINGS

Each instance is assigned once from a reference run under the default configuration p^{def} and the common time limit $\tau = 600\,\mathrm{s}$, following the case definitions in Section 3. Case 1 contains instances for which the reference run proves optimality or infeasibility, Case 2 contains instances with a feasible primal solution but no proof of optimality, and Case 3 contains the remaining hard instances evaluated through the dual-progress objective. These labels remain fixed in all later experiments and are not recomputed after tuning, so the classification of instances is consistent across methods and tuned configurations do not change the evaluation objective.

Table 7 gives the source collection, construction method, and fixed default-regime counts by family.

Table 7: Benchmark family, construction method, and fixed default-regime counts for the 215-instance evaluation set.

Family	Source collection	Construction route	#	Case 1	Case 2	Case 3
CVRP	CVRPLIB	two-commodity flow MILP export	90	11	39	40
GAP	OR-Library `gapc/gapd`	assignment MILP export	12	7	5	0
JSP	OR-Library `jobshop1`	disjunctive MILP export	24	7	17	0
UFL	UflLib Uniform	facility-location MILP export	30	30	0	0
MIPLIB	MIPLIB 2017 `benchmark-v2`	direct MPS benchmark instances	59	21	30	8

Tables 8 and 9 give the complete instance-level case assignments used by all experiments. The labels are the fixed default-regime labels from Table 7.

A.2 Training and Test Sets within Each Problem Family

Exp. C uses the Case 1 pools of CVRP, GAP, JSP, and UFL for within-family transfer. C1 uses every instance in these pools as a single training anchor, giving 55 anchor splits, and tests the selected configuration on the remaining Case 1 instances in the same family. For C2, each eligible family s and train size $m \in \{2, 3, 4, 5, 6\}$ has five unique training subsets, generated using the order of the instance list and stable seed key `family|m|42`. Each subset leaves at least four Case 1 test instances, and its test set is the complement of the training subset within that family. Same-family Case 2 and Case 3 instances, when available, provide additional transfer evaluations after tuning. Table 10 reports the C2 split design. Logs from test instances are excluded from the tuning loop.

B Experimental Protocol and Baselines

We specify the shared comparison protocol and the Exp. B controls. Table 11 summarizes the settings used throughout the reported tuning experiments. The subsections describe method-specific feedback, representations of the search space, and prompt components removed by each control. Appendix F gives concrete prompt/output examples. Figure 6 shows the prompt components used by LDST, Figure 7 shows a representative JSON response, and Table 17 identifies the prompt components retained or removed by each Exp. B control.

B.1 Shared Experimental Protocol

Table 11 specifies the primary SCIP comparison protocol. Extensions to the model and the set of tunable parameters are documented in Appendix J, and the CBC implementation in Appendix K. The protocol fixes solver calls, objective accounting, proposal validation, and randomness controls before each method-specific policy is applied.

Time measurements. Both SCIP and its LP solver use one thread. The 215-instance main study runs one instance job at a time and uses an externally configured CPU clock for SCIP, with a per-evaluation limit of 600 CPU seconds. Solver times are read from the `Solving Time (sec)` field in SCIP output. Direct API latency and total duration of a tuning run in the cost study are elapsed wall-clock measurements. The tuning budget specifies the number of proposals.

B.2 Baselines Using Scalar Feedback

The adaptive scalar-feedback baselines use the full history of evaluated configuration–loss pairs maintained by the optimizers. Random search is non-adaptive. None of the three methods receives raw logs, parsed diagnostics, formulation text, prompt memory rendered for the LLM, or validation feedback provided to the LLM. Table 12 summarizes their representation of the search space, feedback, and treatment of invalid proposals under the protocol in Table 11.

SMAC3 configuration. We use SMAC3 2.3.1 with its standard `HyperparameterOptimizationFacade`, a deterministic scenario, optimizer seed 42, and 50 proposal evaluations. The configuration space has the same parameter types and domains as the other methods. The initial design contains 12 Sobol configurations. The surrogate is a ten-tree random forest fitted to log-transformed losses, with log expected improvement and $\xi = 0$. Acquisition maximization uses ten local-search starts and 10,000 random challengers, with random proposals interleaved at probability 0.2. The standard intensifier has `max_config_calls=3`; the deterministic scenario uses one seed. Each tuning trajectory uses one worker and evaluates configurations sequentially. Each method evaluates the default separately before its proposal budget, uses that measurement for normalization, and retains the default among the candidates for final selection.

RBFOpt configuration. RBFOpt uses genetic global search with local search disabled. Its initial design contains 19 configurations in each of the 215 main-study trajectories, consuming 38% of the 50-proposal budget. The default evaluation is separate from that initial design. Both adaptive baselines receive objective values at the precision returned by the evaluator.

Table 8: Case assignments for all benchmark instances (part 1 of 2). Cases are assigned once under p^{def} with time limit $\tau = 600\,\mathrm{s}$ and then kept fixed for all experiments.

Family	Instance	Case	Family	Instance	Case
CVRP	A-n32-k5	2	CVRP	A-n33-k5	2
CVRP	A-n33-k6	2	CVRP	A-n34-k5	2
CVRP	A-n36-k5	2	CVRP	A-n37-k5	2
CVRP	A-n37-k6	2	CVRP	A-n38-k5	2
CVRP	A-n39-k5	2	CVRP	A-n39-k6	2
CVRP	A-n44-k6	2	CVRP	A-n45-k6	3
CVRP	A-n45-k7	2	CVRP	A-n46-k7	2
CVRP	A-n48-k7	2	CVRP	A-n53-k7	2
CVRP	A-n54-k7	3	CVRP	A-n55-k9	3
CVRP	A-n60-k9	3	CVRP	A-n61-k9	3
CVRP	A-n62-k8	3	CVRP	A-n63-k10	3
CVRP	A-n63-k9	3	CVRP	A-n64-k9	3
CVRP	A-n65-k9	3	CVRP	A-n69-k9	3
CVRP	A-n80-k10	3	CVRP	B-n31-k5	2
CVRP	B-n34-k5	2	CVRP	B-n35-k5	2
CVRP	B-n38-k6	2	CVRP	B-n39-k5	2
CVRP	B-n41-k6	2	CVRP	B-n43-k6	2
CVRP	B-n44-k7	2	CVRP	B-n45-k5	2
CVRP	B-n45-k6	3	CVRP	B-n50-k7	2
CVRP	B-n50-k8	2	CVRP	B-n51-k7	3
CVRP	B-n52-k7	2	CVRP	B-n56-k7	2
CVRP	B-n57-k7	3	CVRP	B-n57-k9	3
CVRP	B-n63-k10	3	CVRP	B-n64-k9	3
CVRP	B-n66-k9	3	CVRP	B-n67-k10	3
CVRP	B-n68-k9	3	CVRP	B-n78-k10	3
CVRP	E-n101-k14	3	CVRP	E-n101-k8	3
CVRP	E-n13-k4	1	CVRP	E-n22-k4	1
CVRP	E-n23-k3	1	CVRP	E-n30-k3	2
CVRP	E-n31-k7	1	CVRP	E-n33-k4	2
CVRP	E-n51-k5	2	CVRP	E-n76-k10	3
CVRP	E-n76-k14	3	CVRP	E-n76-k7	3
CVRP	E-n76-k8	3	CVRP	F-n135-k7	3
CVRP	F-n45-k4	2	CVRP	F-n72-k4	2
CVRP	P-n101-k4	2	CVRP	P-n16-k8	1
CVRP	P-n19-k2	1	CVRP	P-n20-k2	1
CVRP	P-n21-k2	1	CVRP	P-n22-k2	1
CVRP	P-n22-k8	1	CVRP	P-n23-k8	2
CVRP	P-n40-k5	1	CVRP	P-n45-k5	2
CVRP	P-n50-k10	3	CVRP	P-n50-k7	2
CVRP	P-n50-k8	3	CVRP	P-n51-k10	3
CVRP	P-n55-k10	3	CVRP	P-n55-k15	3
CVRP	P-n55-k7	2	CVRP	P-n55-k8	2
CVRP	P-n60-k10	3	CVRP	P-n60-k15	3
CVRP	P-n65-k10	3	CVRP	P-n70-k10	3
CVRP	P-n76-k4	3	CVRP	P-n76-k5	3
GAP	gapc-1	1	GAP	gapc-2	1
GAP	gapc-3	1	GAP	gapc-4	1
GAP	gapc-5	1	GAP	gapc-6	1
GAP	gapd-1	1	GAP	gapd-2	2
GAP	gapd-3	2	GAP	gapd-4	2
GAP	gapd-5	2	GAP	gapd-6	2
JSP	abz5	2	JSP	abz6	1
JSP	abz7	2	JSP	abz8	2
JSP	abz9	2	JSP	ft10	2

Table 9: Case assignments for all benchmark instances (part 2 of 2). Cases are assigned once under p^{def} with time limit $\tau = 600\,\mathrm{s}$ and then kept fixed for all experiments.

Family	Instance	Case	Family	Instance	Case
JSP	ft20	2	JSP	la01	1
JSP	la02	1	JSP	la03	1
JSP	la04	1	JSP	la05	1
JSP	la06	2	JSP	la07	2
JSP	la08	2	JSP	la09	2
JSP	la10	2	JSP	la11	2
JSP	la12	2	JSP	orb01	2
JSP	orb02	1	JSP	orb03	2
JSP	orb04	2	JSP	orb05	2
MIPLIB	50v-10	2	MIPLIB	CMS750_4	2
MIPLIB	assign1-5-8	2	MIPLIB	atlanta-ip	2
MIPLIB	b1c1s1	2	MIPLIB	beasleyC3	1
MIPLIB	cbs-cta	1	MIPLIB	chromaticindex512-7	2
MIPLIB	cmflsp50-24-8-8	3	MIPLIB	cod105	1
MIPLIB	comp21-2idx	2	MIPLIB	cryptanalysis kb128n5obj16	3
MIPLIB	dano3_3	1	MIPLIB	drayage-25-23	1
MIPLIB	dws008-01	2	MIPLIB	fhnw-binpack4-48	3
MIPLIB	gfd-schedule n180f7d50m30k18	3	MIPLIB	glass-sc	2
MIPLIB	glass4	2	MIPLIB	gmu-35-50	2
MIPLIB	h80x6320d	1	MIPLIB	irish-electricity	3
MIPLIB	lectsched-5-obj	2	MIPLIB	momentum1	2
MIPLIB	mushroom-best	2	MIPLIB	mzzv11	1
MIPLIB	n3div36	2	MIPLIB	neos-1354092	3
MIPLIB	neos-1445765	1	MIPLIB	neos-3046615-murg	2
MIPLIB	neos-3083819-nubu	1	MIPLIB	neos-3627168-kasai	1
MIPLIB	neos-3988577-wolgan	3	MIPLIB	neos-5052403-cygnet	2
MIPLIB	neos-662469	2	MIPLIB	neos-827175	1
MIPLIB	neos-911970	1	MIPLIB	neos8	1
MIPLIB	neos859080	1	MIPLIB	nexp-150-20-8-5	2
MIPLIB	nursesched-sprint02	1	MIPLIB	p200x1188c	1
MIPLIB	piperout-27	1	MIPLIB	radiationm18-12-05	2
MIPLIB	radiationm40-10-02	2	MIPLIB	ran14x18-disj-8	2
MIPLIB	rocI-4-11	1	MIPLIB	rocII-5-11	2
MIPLIB	roll3000	1	MIPLIB	s100	2
MIPLIB	snp-02-004-104	2	MIPLIB	supportcase22	3
MIPLIB	supportcase40	2	MIPLIB	supportcase42	2
MIPLIB	swath1	1	MIPLIB	thor50dday	2
MIPLIB	traininstance2	2	MIPLIB	uct-subprob	2
MIPLIB	unitcal_7	1	UFL	1023UnifS	1
UFL	1123UnifS	1	UFL	1223UnifS	1
UFL	123UnifS	1	UFL	1323UnifS	1
UFL	1423UnifS	1	UFL	1523UnifS	1
UFL	1623UnifS	1	UFL	1723UnifS	1
UFL	1823UnifS	1	UFL	1923UnifS	1
UFL	2023UnifS	1	UFL	2123UnifS	1
UFL	2223UnifS	1	UFL	223UnifS	1
UFL	2323UnifS	1	UFL	2423UnifS	1
UFL	2523UnifS	1	UFL	2623UnifS	1
UFL	2723UnifS	1	UFL	2823UnifS	1
UFL	2923UnifS	1	UFL	3023UnifS	1
UFL	323UnifS	1	UFL	423UnifS	1
UFL	523UnifS	1	UFL	623UnifS	1
UFL	723UnifS	1	UFL	823UnifS	1
UFL	923UnifS	1			

Table 10: Deterministic within-family split design for C2. Each eligible family and training size has five unique splits generated with base seed 42 and at least four Case 1 test instances. C1 separately uses all 55 Case 1 instances as single anchors.

Family	Case 1 pool	Eligible m	Splits per m	Case 1 test count	Held-out transfer targets
CVRP	11	$2, 3, 4, 5, 6$	5	$11 - m$	Case 2 and Case 3
GAP	7	$2, 3$	5	$7 - m$	Case 2 only
JSP	7	$2, 3$	5	$7 - m$	Case 2 only
UFL	30	$2, 3, 4, 5, 6$	5	$30 - m$	none beyond Case 1

Table 11: Shared experimental protocol for the reported tuning comparisons.

Component	Settings
Solver call	SCIP 9.2.4, one thread, one blocking solver run per evaluated configuration and instance, with per-run time limit $\tau = 600\,\mathrm{s}$.
Objective and normalization	The regime-adaptive loss in Section 3 is fixed before comparison and used identically by LDST, LLM controls, and scalar-feedback baselines. The default run is trial 0; it defines the normalized reference and remains eligible as the reported best-so-far incumbent.
Evaluation budget	Each history begins with default trial 0 followed by at most K proposal trials. Trial 0 does not consume the K-proposal budget. A history may end early if the validation and query-attempt loop cannot produce a valid novel configuration.
Parameter space	All methods search the same set $\mathcal{W}$ of 45 tunable SCIP parameters in Appendix C; numeric parameters use a linear scale.
LLM proposals	LLM methods use edit cap $M = 20$, edits around the current best configuration, the same JSON format, at most $V = 3$ total query attempts per proposal including the initial query, and the same check against all previously evaluated configurations.
Validation	Candidate edits must follow the required format, use selected parameters within their domains, satisfy the edit cap after truncation, differ from earlier configurations, and load successfully as SCIP settings before a solver call.
Randomness control	The 215-instance main bank fixes `randomization/randomseedshift`=42; stochastic scalar-feedback optimizers use seed 42. The independent five-seed follow-up is reported separately.

Table 12: Implementation details of the scalar-feedback baselines. SMAC3 and RBFOpt adapt using their own histories of scalar evaluations; none of the methods receives prompt memory rendered for the LLM or diagnostics of solver processes.

Method	Search-space encoding	Feedback	Settings and invalid handling
SMAC3	ConfigSpace with the specified parameter types and linear numeric domains.	scalar evaluation history	Deterministic main study, seed 42, up to K proposals; default trial 0 eligible separately; nonfinite scores cannot be selected.
RBFOpt	Bounds for real and integer parameters; Boolean and categorical parameters represented by integers.	scalar evaluation history	Main study seed 42, no local search, genetic global search, up to K proposals; default trial 0 eligible separately; nonfinite scores cannot be selected.
Random	Independent sampling within parameter domains; positive real intervals log-uniform, all other numeric intervals uniform.	nonadaptive	Main study seed 42, K independent samples; default trial 0 eligible separately.

B.3 LLM CONTROLS

Let Π^{LDST} denote the full LDST prompt. The static header specifies the objective, set of tunable parameters, current best configuration, edit cap, and required output format. Other components include optional formulation text, validation feedback during a repeated attempt, a representation of the trial history for the LLM, a list of previously tried edits, and a diagnostic guide relating parsed log fields to likely solver bottlenecks. Each proposal permits at most $V = 3$ total query attempts, including the initial query, and accepted edit lists contain at most $M = 20$ edits. Figure 4 shows an excerpt of a full LDST prompt from an Exp. A run and identifies the regions most relevant to the Exp. B controls. The three LLM controls remove or restrict selected prompt inputs while leaving the outer tuning loop and validation rules unchanged.

LLM-STATICCONTEXT. LLM-STATICCONTEXT retains the static portion of Π^{LDST} together with the parsed diagnostics of the run of the current best configuration. The retained information includes parameter names with defaults and domains, edits defining the current best configuration, edit and task constraints, optional formulation text, and a single block of current diagnostics. Relative to Figure 4, it keeps A and B but removes the history across trials in C, the diagnostic guide, and the list of previously tried edits from the prompt. The shared procedure still checks novelty against all evaluated configurations before execution. The resulting control exposes only the state of the current best configuration to the LLM.

LLM-OUTCOMEONLY. LLM-OUTCOMEONLY replaces the trial history summary in Π^{LDST} with a scalar-only memory $\mathcal{M}^{\text{SCALAR}}$ that records only the trial id, loss, solve time, final gap, and finite/feasible status when these quantities are defined. Relative to Figure 4, it keeps A and the scalar part of C, but removes the parsed diagnostic fields in C and the diagnostic guide. The control preserves trial-to-trial outcome feedback while removing process-level information from the solver log.

LLM-INDEXONLY. LLM-INDEXONLY keeps the trial history, diagnostics derived from solver logs, diagnostic guide, and list of previously tried edits, but restricts the semantic information about the parameters. Relative to Figure 4, it keeps C but replaces the parameter names, groups, and defaults in A with editable parameter indices, domain hints for each index, and the subset of indices corresponding to Boolean parameters. These hints support valid editing but do not expose parameter semantics. The loop, validator, and required JSON format are unchanged except that the optional textual `whitelist` field is omitted from the returned edits.

Formulation and Leakage Control. All reported LLM-based runs use `formulation=auto` unless stated otherwise. The formulation text supplied to the model is a family-level mathematical description obtained from `utilities/formulations.py`; it depends only on the benchmark family and is identical across instances from the same family. It is never constructed from held-out logs or solve traces of the target instance. In Exp. C, the LLM sees only logs from the training instances during the active tuning loop; held-out Case 1/2/3 logs are evaluated only after a configuration has been selected and are never returned to the prompt during tuning. Table 13 summarizes the corresponding information available to the black-box baselines, the LLM controls, and the full LDST loop.

Table 13: Information and validation available to the main control families. The LLM scalar-history column records only history shown in a prompt; SMAC3 and RBFOpt separately retain scalar history in the optimizers. The BBO row groups SMAC3, RBFOpt, and Random because none receives an LLM prompt. For LLM-INDEXONLY, domain hints associated with indices are not counted as parameter semantics because parameter names and defaults are hidden.

Method	LLM scalar hist.	Parsed diag.	Param. sem.	Form. text	Rationale	Validator
BBO baselines	✗	✗	✗	✗	✗	✗
LLM-STATICCONTEXT	✗	✓	✓	optional	✓	✓
LLM-OUTCOMEONLY	✓	✗	✓	optional	✓	✓
LLM-INDEXONLY	✓	✓	✗	optional	✓	✓
LDST	✓	✓	✓	optional	✓	✓

A Semantic/static context

```
You are tuning SCIP parameters for a mixed-integer programming (MIP) solver.
Objective: min loss (intuition: ~ min primal-dual integral) (loss is normalized)
Current best loss: 0.2931
Edit cap M: 20
Whitelist names: branching: scorefunc(1;p), preferbinary(2;false), relpscost/{minreliable(3;1), maxreliable(4;5),
sbiterquot(5;0.50), sbiterofs(6;100000)} | nodeselection: estimate/stdpriority(7;200000), dfs/stdpriority(8;0),
bfs/stdpriority(9;100000), childsel(10;h) | separating: maxrounds(11;-1), maxroundsroot(12;-1), maxcuts(13;100),
maxcutsroot(14;2000), gomory/freq(15;10), cmir/freq(16;10), flowcover/freq(17;10), clique/freq(18;0),
knapsackcover/freq(19;10), oddcycle/freq(20;-1) | presolving: maxrounds(21;-1), maxrestarts(22;-1), abortfac(23;0.00) |
heuristics: feaspump/{freq(24;20), freqofs(25;0), maxdepth(26;-1), maxlpiterquot(27;0.01), maxlpiterofs(28;1000),
beforecuts(29;true)}, rins/{nodesofs(30;500), nodesquot(31;0.30), minnodes(32;50), maxnodes(33;5000),
nwaitingnodes(34;200), minfixingrate(35;0.30)}, localbranching/{neighborhoodsize(36;18), nodesofs(37;1000),
nodesquot(38;0.05), lplimfac(39;1.50)}, rens/{nodesofs(40;500), nodesquot(41;0.10), minnodes(42;50), maxnodes(43;5000),
minfixingrate(44;0.50), startsol(45;1)}
Note: format is name(index;default).
Boolean params must use true/false (not 0/1).
Bool params: branching/preferbinary(2), heuristics/feaspump/beforecuts(29)
Tuning set: 1 instance (single-instance); loss = normalized single-instance loss; diagnostics: raw per-run values
```

B Current-best configuration and formulation

```
Incumbent edits (delta based on defaults):
{ (11)=5, (12)=500, (13)=500, (14)=50000, (15)=-1, (16)=-1, (17)=-1, (18)=-1, (19)=-1 }

Problem formulation
We adopt the two-commodity network-flow formulation of \citet{baldacci2004exact}, which is a compact mixed-integer model
 that can be solved to optimality without column or row generation:
\begin{equation*}
\begin{alignedat}{2}
\min\quad & \sum_{\{i,j\}\in \mathcal{E}} c_{ij}\,x_{ij} & & \\
\text{s.t.}\quad
& \sum_{j\in \mathcal{V}\setminus\{i\}} \big(y_{ji} - y_{ij}\big) = 2d_i,
&\quad& \forall i\in \mathcal{C}, \\
& \sum_{j\in \mathcal{C}} y_{0j} = d(\mathcal{C}), & & \\
& \sum_{j\in \mathcal{C}} y_{j0} = KQ - d(\mathcal{C}), & & \\
& \sum_{j\in \mathcal{C}} y_{n+1,j} = KQ, & & \\
& y_{ij} + y_{ji} = Q\,x_{ij},
&\quad& \forall \{i,j\}\in \mathcal{E}, \\
& \sum_{\substack{j\in \mathcal{V}\\ j>i}} x_{ij}
 + \sum_{\substack{j\in \mathcal{V}\\ j<i}} x_{ji} = 2,
&\quad& \forall i\in \mathcal{C}, \\
& y_{ij} \ge 0,\; y_{ji} \ge 0,
&\quad& \forall \{i,j\}\in \mathcal{E}, \\
& x_{ij} \in \{0,1\},
&\quad& \forall \{i,j\}\in \mathcal{E}.
\end{alignedat}
\end{equation*}
Here, $\mathcal{G}=(\mathcal{V},\mathcal{E})$ is a complete undirected graph with two depot copies, where
$\mathcal{V}=\{0,1,\dots,n,n+1\}$ and the customer set is $\mathcal{C}=\mathcal{V}\setminus\{0,n+1\}$. The edge set is
\[
\mathcal{E}=\big\{\{i,j\}: i,j\in\mathcal{V}\setminus \{n+1\},\ i<j\big\}\cup\big\{\{i,n+1\}: i\in \mathcal{C}\big\}.
\]
The travel cost on edge $\{i,j\}$ is $c_{ij}=c_{ji}\ge 0$. Each customer $i\in\mathcal{C}$ has demand $d_i\ge 0$, and
$d(\mathcal{C})=\sum_{i\in\mathcal{C}} d_i$. Vehicles have capacity $Q>0$, and $K\in\mathbb{Z}_{\ge 0}$ denotes the
fleet size. Binary variables $x_{ij}$ indicate whether edge $\{i,j\}$ is used, while continuous variables
$y_{ij},y_{ji}\ge 0$ represent the two commodities (vehicle load and residual capacity) flowing on edge $\{i,j\}$.
```

C Archive memory and parsed diagnostics

```
[MEMORY] 3 best + 3 recent (after dedup)
cols: trial | loss | delta | solve_time | solving_nodes | nodes_left | max_depth | nsols | num_improvements |
first_incumbent_time | first_primal | final_primal | final_dual | final_gap | primal_dual_integral | presolve_time |
presolved_vars | presolved_cons_init | presolved_cons_max | root_dual0 | root_dual_end | root_dual_gain | root_gain_rel
| root_first_lp_iters | root_first_lp_time | root_final_root_iters | root_lp_estimate | sb_time | sb_calls | sb_iters |
sb_root_calls | sb_root_iters | cuts_selected_root | sep_gomory_time | sep_gomory_applied | sep_clique_time |
sep_clique_applied | sep_oddcycle_time | sep_oddcycle_applied | sep_cmir_applied | sep_flowcover_applied |
sep_knapsackcover_applied | heur_feaspump_time | heur_feaspump_calls | heur_feaspump_found | heur_feaspump_best |
heur_rins_time | heur_rins_calls | heur_rins_found | heur_rins_best | heur_localbranching_time |
heur_localbranching_calls | heur_localbranching_found | heur_localbranching_best | heur_rens_time | heur_rens_calls |
heur_rens_found | heur_rens_best
- 2 | 0.29 | {} | 600 | 298922 | 180760 | 108 | 23 | 12 | 25.25 | 1043 | 964 | 838.07 | 125.93 | 10991.56 | 0.06 | 2106
| 777 | 5689 | 797.49 | 801.25 | 3.76 | 0.00 | 1632 | 0.04 | 3025 | 932.07 | 11.24 | 11653 | 188191 | 34 | 948 | 2 | 0 |
 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.55 | 19 | 0 | 0 | 1.92 | 63 | 9 | 4 | 0 | 0 | 0 | 0 | 0.02 | 1 | 0 | 0
- 1 | 0.92 | { (11)=-1, (12)=300, (14)=20000, (15)=0, (16)=1, (17)=1, (19)=1 } | 600 | 41473 | 19092 | 75 | 8 | 6 |
304.06 | 1296 | 998 | 876.96 | 121.04 | 34401.07 | 0.06 | 2106 | 777 | 5143 | 797.49 | 865.28 | 67.79 | 0.09 | 1632 |
0.04 | 9595 | 1043.07 | 41.01 | 5342 | 416249 | 34 | 13304 | 975 | 0.03 | 1516 | 0 | 0 | 0 | 0 | 75815 | 7984 | 675 |
2.41 | 52 | 0 | 0 | 2.48 | 16 | 4 | 4 | 0 | 0 | 0 | 0 | 1.11 | 1 | 0 | 0
- 0 | 1 | { (11)=-1, (12)=-1, (13)=100, (14)=2000, (15)=10, (16)=10, (17)=10, (18)=0, (19)=10 } | 600 | 43008 | 15158 |
77 | 10 | 6 | 352.03 | 1029 | 968 | 882.06 | 85.94 | 37502.32 | 0.06 | 2106 | 777 | 4563 | 797.49 | 865.28 | 67.79 |
0.09 | 1632 | 0.04 | 9595 | 1020.89 | 45.25 | 6049 | 443851 | 34 | 13304 | 975 | 5.93 | 19091 | 0 | 0 | 0 | 0 | 66741 |
7229 | 422 | 2.73 | 56 | 0 | 0 | 2.19 | 18 | 5 | 5 | 0 | 0 | 0 | 0 | 1.11 | 1 | 0 | 0
```

Figure 4: Prompt excerpt from a full LDST run on CVRP instance A-n37-k6 at trial 2. Panels show semantic and static context (A), current best configuration and formulation (B), and trial history with parsed diagnostics (C).

C SCIP PARAMETERS AND CONFIGURATION SPACE

The primary SCIP experiments and their LLM controls use the same curated set $\mathcal{W}$ of 45 tunable parameters. The 90-parameter extension is documented in Appendix J. The set covers branching, node selection, cut separation, presolving, and primal heuristics, the main solver subsystems represented in SCIP logs. Table 14 specifies the raw SCIP parameter path, type, default value, allowed domain, and scale for each of the 45 entries. All numeric entries are edited on their original linear scale; there are no hidden parameter-specific transformations. The validator used for LLM proposals checks compliance with the required JSON format, truncates edits to the cap, checks novelty against all evaluated configurations, and verifies that the resulting SCIP settings file loads successfully.

Table 14: Full set of tunable parameters and their types used in the experiments, generated directly from the code under SCIP 9.2.4.

SCIP parameter path	Type	Default	Allowed domain	Scale
branching/scorefunc	cat	p	s, p, q	lin.
branching/preferbinary	bool	false	false, true	lin.
branching/relpscost/minreliable	float	1	[0, 1000000000]	lin.
branching/relpscost/maxreliable	float	5	[0, 1000000000]	lin.
branching/relpscost/sbiterquot	float	0.5	[0, 1000000000]	lin.
branching/relpscost/sbiterofs	int	100000	[0, 1000000000]	lin.
nodeselection/estimate/stdpriority	int	200000	[-536870912, 1000000000]	lin.
nodeselection/dfs/stdpriority	int	0	[-536870912, 1000000000]	lin.
nodeselection/bfs/stdpriority	int	100000	[-536870912, 1000000000]	lin.
nodeselection/childsel	cat	h	d, u, p, i, l, r, h	lin.
separating/maxrounds	int	-1	[-1, 1000000000]	lin.
separating/maxroundsroot	int	-1	[-1, 1000000000]	lin.
separating/maxcuts	int	100	[0, 1000000000]	lin.
separating/maxcutsroot	int	2000	[0, 1000000000]	lin.
separating/gomory/freq	int	10	[-1, 1000000000]	lin.
separating/cmir/freq	int	10	[-1, 1000000000]	lin.
separating/flowcover/freq	int	10	[-1, 1000000000]	lin.
separating/clique/freq	int	0	[-1, 1000000000]	lin.
separating/knapsackcover/freq	int	10	[-1, 1000000000]	lin.
separating/oddcycle/freq	int	-1	[-1, 1000000000]	lin.
presolving/maxrounds	int	-1	[-1, 1000000000]	lin.
presolving/maxrestarts	int	-1	[-1, 1000000000]	lin.
presolving/abortfac	float	0.0008	[0, 1]	lin.
heuristics/feaspump/freq	int	20	[-1, 1000000000]	lin.
heuristics/feaspump/freqofs	int	0	[0, 1000000000]	lin.
heuristics/feaspump/maxdepth	int	-1	[-1, 1000000000]	lin.
heuristics/feaspump/maxlpiterquot	float	0.01	[0, 1000000000]	lin.
heuristics/feaspump/maxlpiterofs	int	1000	[0, 1000000000]	lin.
heuristics/feaspump/beforecuts	bool	true	false, true	lin.
heuristics/rins/nodesofs	int	500	[0, 1000000000]	lin.
heuristics/rins/nodesquot	float	0.3	[0, 1]	lin.
heuristics/rins/minnodes	int	50	[0, 1000000000]	lin.
heuristics/rins/maxnodes	int	5000	[0, 1000000000]	lin.
heuristics/rins/nwaitingnodes	int	200	[0, 1000000000]	lin.
heuristics/rins/minfixingrate	float	0.3	[0, 1]	lin.
heuristics/localbranching/neighborhoodsize	int	18	[1, 1000000000]	lin.
heuristics/localbranching/nodesofs	int	1000	[0, 1000000000]	lin.
heuristics/localbranching/nodesquot	float	0.05	[0, 1]	lin.
heuristics/localbranching/lplimfac	float	1.5	[1, 1000000000]	lin.
heuristics/rens/nodesofs	int	500	[0, 1000000000]	lin.
heuristics/rens/nodesquot	float	0.1	[0, 1]	lin.
heuristics/rens/minnodes	int	50	[0, 1000000000]	lin.
heuristics/rens/maxnodes	int	5000	[0, 1000000000]	lin.
heuristics/rens/minfixingrate	float	0.5	[0, 1]	lin.
heuristics/rens/startsol	cat	l	n, l	lin.

D LLM HYPERPARAMETERS AND SHARED SETTINGS

The LLM acts as a constrained proposal mechanism. Table 15 reports the model settings for the main LDST experiments that are not already fixed by the protocol in Appendix B; experiment-specific proposal budgets are reported in the corresponding experiment descriptions and in Appendix M.

Table 15: Model hyperparameters used by the main LLM tuning experiments.

Item	Value
Prompt memory limits (B, R)	3, 3
Edit cap M	20
Query-attempt cap V	3 total attempts, including the initial query
LLM model	GPT-5.2
Reasoning effort	medium
Max output tokens	20,000

E DIAGNOSTIC INFORMATION FROM SCIP LOGS

For each run k with raw SCIP log $\Lambda^{(k)}$, the parser returns the diagnostic vector with a fixed format $\mathbf{d}^{(k)} = \mathrm{Parse}(\Lambda^{(k)})$ used to construct the prompt in Section 4. The parser reads the standard SCIP branch-and-bound progress table, root node report, presolve summary, heuristic table, separator table, and final solving summary. Figure 5 illustrates the conversion from a raw SCIP excerpt to the compact diagnostic state used by LDST.

```
=== RAW SCIP LOG (excerpt) ===
presolved problem has 953 vars,
358 cons
presolving time: 0.12

 time | node  | left  | gap
 0.2s |     1 |     0 | 12.44%
 0.3s |     1 |     0 |  4.50%
 7.5s |  4200 |  3971 |  0.23%
10.1s |  6691 |  6318 |  0.17%
23.7s | 18580 | 17424 |  0.11%
```

```
=== PARSED DIAGNOSTICS
    (excerpt) ===
t_total             = 600.0
nodes_last          = 432241
nodes_left          = 319848
max_depth           = 108
num_improvements    = 12
first_inc_time      = 0.0
gap_final           = 1500.39
pdi                 = 76.83
presolve_time       = 0.12
root_dual_gain      = 9.06e-03
sb_time             = 6.05
sb_iters            = 116575
cuts_selected_total= 95704
sep_cmir_applied    = 29937
heur_rins_time      = 0.41
```

Figure 5: Representative raw SCIP log excerpt and corresponding compact diagnostic summary returned by the parser. The left box is taken from a real Case 2 run; the parser also reads the corresponding bound, LP, cut, separator, heuristic, and summary sections from the same log. The right box shows the diagnostic state in a fixed format used by LDST.

Table 16 defines the diagnostic quantities used to construct the prompt. Let $\mathcal{R}^{(k)}$ be the ordered progress-table rows parsed from run $\Lambda^{(k)}$; row $r \in \mathcal{R}^{(k)}$ contains elapsed time t_r, processed nodes n_r, primal bound z_r^{pr}, and dual bound z_r^{du} when these columns are present. For any row field x, let $\mathrm{first}_f(x)$ and $\mathrm{last}_f(x)$ denote the first and last finite parsed values, and set $\eta = 10^{-12}$. Let $\mathcal{S} = \{\mathrm{gomory}, \mathrm{clique}, \mathrm{oddcycle}, \mathrm{cmir}, \mathrm{flowcover}, \mathrm{knapsackcover}\}$ and $\mathcal{H} = \{\mathrm{feaspump}, \mathrm{rins}, \mathrm{localbranching}, \mathrm{rens}\}$ denote the tracked separator and heuristic families. Summary quantities are taken from the final SCIP summary whenever available, with the alternative calculation from progress rows shown below.

Table 16: Diagnostic quantities parsed from standard SCIP logs and passed to the prompt. The aliases in parentheses are the names shown in the prompts used in the experiments.

Block	Parser fields exposed to the policy	Implementation definition	SCIP source
Run size and status	`status`, `t_total` (shown as `solve_time`), `nodes_last` (shown as `solving_nodes`), `nodes_left`, `max_depth`	`t_total` is the reported solving time, or $\mathrm{last}_f(t)$ if missing; `nodes_last` is the reported solving-node count, or $\mathrm{last}_f(n)$ if missing. `nodes_left` and `max_depth` are the B&B-tree summary values.	final summary, B&B tree, progress rows
Bounds and integral	`primal_final`, `dual_final`, `gap_final`, `pdi` (shown as `primal_dual_integral`)	`primal_final` and `dual_final` are final-summary bounds, falling back to $\mathrm{last}_f(z^{\mathrm{pr}})$ and $\mathrm{last}_f(z^{\mathrm{du}})$. `gap_final`$= \lvert$`primal_final` $-$ `dual_final`$\rvert$. `pdi`$= \mathrm{PDI}^{\mathrm{SCIP}}(\Lambda^{(k)})$ is the SCIP-reported primal-dual integral parsed from the Integrals block.	final summary, Integrals block
Incumbents	`nsols`, `num_improvements`, `first_inc_time` (shown as `first_incumbent_time`), `first_primal`	`nsols` and `num_improvements` are solution-summary counts. `first_inc_time` and `first_primal` use the first-solution summary when present; otherwise they use the time and value at $\mathrm{first}_f(z^{\mathrm{pr}})$.	solution summary, progress rows
Presolve	`presolve_time`, `presolved_vars`, `presolved_cons_init`, `presolved_cons_max`	Direct parser entries for presolving time, presolved-variable count, initial presolved-constraint count, and maximum presolved-constraint count.	presolve summary
Root progress	`root_dual0`, `root_dual_end`, `root_dual_gain`, `root_gain_rel`, `root_first_lp_iters`, `root_first_lp_time`, `root_final_root_iters`, `root_lp_estimate`	`root_dual0` is the first root-LP value and `root_dual_end` is the final root dual bound. The derived fields are `root_dual_gain`$=$ `root_dual_end` $-$ `root_dual0` and `root_gain_rel`$=$ `root_dual_gain`$/\max\{\lvert$`root_dual0`$\rvert, \eta\}$.	root-node report
LP and branching effort	`sb_time`, `sb_calls`, `sb_iters`, `sb_root_calls`, `sb_root_iters`	`sb_time`, `sb_calls`, and `sb_iters` are parsed from the LP-statistics row for strong branching; `sb_root_calls` and `sb_root_iters` are parsed from the root-node LP row.	LP statistics
Cut separation	`cuts_selected_total`, `cuts_selected_root`, and applied-cut or time fields for $c \in \mathcal{S}$	`cuts_selected_total`$= \sum_q$ `selected`$_q$ and `cuts_selected_root`$= \sum_q$ `rootselec`$_q$ over cut selectors q. For each separator family c, the parser records the applied-cut count and records execution time when the SCIP table exposes that entry.	cut-selector and separator tables
Heuristic activity	time, call, found-solution, and best-solution counts for $h \in \mathcal{H}$	For each heuristic family h, the parser exposes the `calls`, `exectime`, `found`, and `best` columns reported by SCIP as prompt diagnostics.	primal-heuristics table

The parser converts raw solver logs into the compact state used by the decision policy. In single-instance mode, the resulting summary is shown directly in the trial history in the prompt. In multi-instance mode, the same quantities are computed for each instance and shown as aligned rows, preserving comparability across instances without exposing the model to raw logs.

F Prompt Construction and Representative Outputs

We document a representative prompt/output pair and its relation to the Exp. B ablations. The examples specify the information available to each policy and the representation of parameter edits and their generated rationale. An accepted edit list contains at most $M = 20$ edits, and the validation loop permits at most $V = 3$ total query attempts, including the initial query.

```
=== LDST Prompt ===
You are tuning SCIP parameters for a
mixed-integer programming solver.
Objective: min loss
(intuition: ~ min solution time for opt)
(loss is normalized)
Current best loss: 0.4436
Edit cap M: 20
Whitelist names:
branching: scorefunc(1;p),
preferbinary(2;false), ...
| nodeselection: ... | separating: ...
| presolving: ... | heuristics: ...
Tuning set: 1 instance (single-instance);
loss = normalized single-instance loss;
diagnostics: raw per-run values

Current-best edits (delta based on defaults):
{ (2)=true, (4)=1, (5)=0.10, (6)=5000, (11)=0, (12)=50,
  (13)=200, (14)=5000, ... }

Problem formulation
We adopt the two-commodity network-flow
formulation of \citet{baldacci2004exact} ...

[MEMORY] 3 best + 3 recent (after dedup)
cols: trial | loss | delta | solve_time |
      solving_nodes | nodes_left | ...
- 2 | 0.44 | {} | 1.18 | 1074 | 0 | 26 | ...
- 1 | 0.57 | { (4)=2, (5)=0.20, (6)=20000, (11)=-1, (12)=-1 }
  | 1.52 | ...
- 0 | 1 | { (2)=false, (4)=5, (5)=0.50, (6)=100000, ... }
  | 2.66 | ...

[TIPS]
Use the diagnostics and memory to identify
ONE dominant bottleneck before ...

[DEDUP]
Avoid these (loss, delta based on current best)
other than already listed above
(most recent first):
(empty)

[TASK]
Propose the next configuration as a sparse edit set Delta p
relative to the current best configuration ...
Use <= 20 edits and only whitelist indices.
Return ONLY valid JSON with this schema:
{"edits": [
  {"index": "<int>", "whitelist": "<str>",
   "value": "<val>"}
 ],
 "rationale":
 "<brief reasoning tied to the observed diagnostics>"}
```

Figure 6: Representative prompt excerpt (single instance, trial 2). The shown components comprise static context, edits defining the current best configuration, formulation text when available, trial history, the diagnostic guide, the list of previously tried edits, and the required output format.

Figure 6 shows that the prompt is centered on the current best configuration rather than the last trial, and that the trial history summarizes both the best and most recent trials using the same defined

quantities. The LLM receives a compact, comparable representation across trials instead of raw solver logs.

Table 17: Prompt-content ablations used in Exp. B, instantiated on the same prompt excerpt as Figure 6. Each control keeps the same outer tuning loop and changes only the listed prompt components. "Current diagnostics only" removes trial history but keeps the diagnostic block for the current best configuration; "scalar only" retains trial id, loss, solve time, final gap, and finite/feasible status; "full, indexed" keeps the diagnostic history while replacing semantic parameter names by indices.

Prompt block in Figure 6	LDST	StaticContext	OutcomeOnly	IndexOnly
Header, parameters, current best, edit cap, format	semantic names, defaults, domains	same	same	indices and domains
Optional formulation text	optional	optional	optional	optional
Validator feedback after a failed attempt	kept	later attempts only	kept	kept
`[MEMORY]` trial history	scalar + diagnostics	current diagnostics only	scalar only	full, indexed
Multi-instance diagnostic rows	kept	removed	removed	kept
`[TIPS]` diagnostic-feature guide	kept	removed	removed	kept
`[DEDUP]` prior configurations	kept	removed	kept	kept

Table 17 specifies the prompt ablations in Exp. B. The variants share the same tuning loop while changing the listed combinations of process diagnostics, scalar history shown to the LLM, parameter semantics, the diagnostic guide, and the list of previously tried edits shown in the prompt. The `[MEMORY]` and `[DEDUP]` rows describe prompt content. The tuning procedure retains the full trial history, validation, and novelty checks when either prompt block is removed.

```
=== LLM_RESPONSE_JSON (excerpt) ===
{
  "edits": [
    {"index":"3",
     "whitelist":"branching: relpscost/minreliable",
     "value":"2"},
    {"index":"4",
     "whitelist":"branching: relpscost/maxreliable",
     "value":"4"},
    {"index":"5",
     "whitelist":"branching: relpscost/sbiterquot",
     "value":"0.05"},
    {"index":"6",
     "whitelist":"branching: relpscost/sbiterofs",
     "value":"50"}
  ],
  "rationale":
  "1) The current best configuration already fixes
   the dominant issue at root (root_dual_gain 4.84,
   nodes drop to 126), and separation times are tiny;
   so root cutting is not the bottleneck anymore.
   2) The remaining dominant cost is expensive branching:
   sb_time=16 out of 28\,s with very large sb_iters
   (128118) despite the small tree.
   3) Therefore, reduce strong-branching effort by
   making pseudocosts become 'reliable' sooner
   (lower minreliable/maxreliable) and by tightening
   the SB LP-iteration limits
   (lower sbiterquot/sbiterofs).
   This should cut sb_time substantially
   while accepting a modest risk of more nodes."
}
```

Figure 7: Example model response proposing reductions in strong branching effort from the reported time and LP iteration statistics. The rationale is generated by the model.

F.1 DIAGNOSTIC GUIDES

The prompt includes a fixed diagnostic guide. Cases 1 and 2 share the following text.

```
Use the diagnostics and memory to identify ONE dominant bottleneck before
     changing parameters: (A) weak root relaxation (small root_dual_gain
    or root_gain_rel), (B) exploding tree (very large solving_nodes with
    high nodes_left and large max_depth), (C) expensive branching (large
    sb_calls/sb_iters or high sb_root_calls/sb_root_iters), or (D) late
    incumbents (large first_incumbent_time, few num_improvements, high
    primal_dual_integral). Spend effort where ROI is highest: if
    root_dual_gain is small but root_first_lp_iters/time is modest,
    increase root separation budget and enable only the most effective
    cut families; if sb_calls/sb_iters are high with little improvement
    in final_gap, dial back strong-branching effort. If the tree is huge
    early, improve branching quality and node selection; if incumbents
    are late, increase budgets for heuristics that actually improve best
    solutions (heur_*_best/heur_*_found) and reduce those that consume
    time without improvements. As the gap becomes small, avoid escalating
     expensive separation/strong-branching further; prefer faster node
    processing and targeted heuristics to close the gap. Early
    exploration: when testing the benefit of a single mode (e.g.,
    aggressive cuts or branching focus), push that mode to an extreme
    setting to get a clear signal; once you have evidence, switch to
    smaller, targeted changes to test combinations and exploit the best
    mix. Exploration ordering: when signals are mixed or unclear, start
    with a root-cut probe that is truly aggressive (e.g., maxroundsroot
    200-500 and maxcutsroot 10000-50000, plus 1-2 strong cut families
    enabled), then compare against cut-reduction in the next step.
```

Case 3 uses the following guide for improving the dual bound.

```
Case 3 is a dual-bound regime. Primary goal: improve the final dual bound
     by tau; do not spend most edits on incumbent-finding unless the logs
     show that no incumbent exists or a heuristic has already produced
    real gains. Rank bottlenecks in this order: (A) weak root relaxation,
     (B) expensive branching / exploding tree, (C) late incumbents only
    as a rescue path. Use the diagnostics and memory to choose EXACTLY
    ONE mode for this trial: (1) root_cut_probe, (2) branch_light_probe,
    (3) incumbent_rescue, or (4) mixed_followup. If root_dual_gain or
    root_gain_rel is weak, start with root_cut_probe: raise root cut
    budgets enough to get a clear signal (e.g., maxroundsroot 200-500,
    maxcutsroot 10000-50000, plus 1-2 cut families). If sb_time, sb_iters
    , nodes_left, or max_depth are large, use branch_light_probe: reduce
    strong-branching burden, improve branching quality, and adjust node
    selection before adding even more expensive cuts. Only use
    incumbent_rescue when nsols=0, first_incumbent_time is very late, or
    heur_*_found / heur_*_best show that a heuristic direction is already
     working. When recent losses are nearly flat, avoid tiny local edits;
     use one coordinated probe with 5-10 edits so the next trial yields a
     measurable signal. After an aggressive probe, either refine the
    winning direction or test the opposite direction; do not keep making
    small ambiguous changes.
```

G VALIDATION OF THE CASE 1 EXTRAPOLATION SIGNAL

We evaluate fractional-cutoff accuracy on completed default runs, prediction-only configuration ranking, and absolute error on a small long-horizon sample. The ranking comparison is restricted to configurations whose runs remain unfinished at the cutoff.

G.1 Tree Growth Model and Constrained Fit

Remaining proof effort reflects both the open search tree and its distance from closing the bound gap. For an unfinished run, retain rows at or before $\tau' = \min\{\tau, t_{\max}\}$ with finite time, number of processed nodes, number of open nodes, primal bound, and dual bound, discarding rows with at most one processed node. Observation j contains elapsed time t_j, processed nodes n_j, and open nodes u_j; define $g_j = |z^{\mathrm{pr}}(\tau') - z^{\mathrm{du}}(t_j)|$ using the primal bound available at the cutoff.

Following the single-variable branching (SVB) model (Le Bodic & Nemhauser, 2017), approximate the remaining subtree size per open node by x^{g_j}. Adding the already processed nodes yields the surrogate for the terminal tree size $\widehat{N}_j(x) = n_j + u_j x^{g_j}$, where $x > 1$ controls growth per unit gap. Fitting a common terminal size reconciles estimates from different observation times on a logarithmic scale. At least eight valid rows are required; each attempt uses up to 25 stratified observations indexed by $\mathcal{T}$. Eq. (2) minimizes absolute log residuals with a regularizer, using residual slacks e_j and nonnegative penalty slack v.

$$
\begin{aligned}
\min_{x,b,\{e_j\}_{j\in\mathcal{T}},v} \quad & \sum_{j\in\mathcal{T}} e_j + \lambda v \\
\text{s.t.} \quad & e_j \geq \log b - \log \widehat{N}_j(x), && j \in \mathcal{T}, \\
& e_j \geq \log \widehat{N}_j(x) - \log b, && j \in \mathcal{T}, \\
& v \geq \sum_{j\in\mathcal{T}} e_j - k \log b, \qquad v \geq 0, \\
& x \in [1.0001, 10^5], \qquad b \in [b_{\mathrm{lb}}, 2b_{\mathrm{lb}}].
\end{aligned}
\tag{2}
$$

Here n_{last} and u_{last} are taken from the last sampled observation, $b_{\mathrm{lb}} = n_{\mathrm{last}} + u_{\mathrm{last}}$, $k = 0.15|\mathcal{T}|$, and $\lambda = 10$. The regularizer penalizes residual mass above $k \log b$, while the interval for b restricts the fitted terminal size.

G.2 Completion Time Prediction

Solved runs return their observed completion time. For a truncated run, recent time per node is $\theta = (t_{\mathrm{last}} - t_0)/(n_{\mathrm{last}} - n_0)$, using the earliest and latest of the last five sampled points. If the number of processed nodes does not increase, the fallback is 0.01 seconds per node. With fitted terminal size $b^\star$, the raw prediction is $\widetilde{T}_\infty = \tau' + \theta \max\{b^\star - n_{\mathrm{last}}, 0\}$.

At most ten sampled fits are attempted. The first prediction satisfying $\tau'/\widetilde{T}_\infty \geq 0.5$ is accepted; if none satisfies this condition but finite predictions exist, the minimum finite prediction is used. All finite outputs are capped at 10τ. When every fit fails, the internal value of 10^{12} passes through the same cap, giving an effective completion time of $6{,}000\,\mathrm{s}$ at $\tau = 600\,\mathrm{s}$. The Case 1 objective subsequently divides this value by the default completion time.

Actual tuning runs enforce the $600\,\mathrm{s}$ limit, so their final summaries contain information available at the endpoint. For the prediction-only ranking and long-horizon studies, we reconstruct each truncated summary from rows at or before the cutoff, including the primal bound available at the cutoff. Full-run outcomes provide reference completion times and determine the retrospective cutoff in the fractional-cutoff studies.

G.3 Retrospective Accuracy on Completed Runs

Following the offline simulation protocol of Hendel et al. (2022), we use 100 completed default runs from the CVRPLIB XML100 collection (Lima et al., 2014), each certifying optimality within an extended time limit of $1000\,\mathrm{s}$ under the default configuration p^{def}. For each instance with ground-truth time-to-optimality $T^\star$, we truncate the log at a grid of fractions $t_k/T^\star$ between 50% and 90%, set $\tau = t_k$, and apply the SVB fit to obtain $\widehat{T}_\infty(k)$. The final incumbent from the completed default run supplies the primal reference in g_j, while the truncated log supplies the tree, dual-bound, and time observations.

We summarize prediction error through the ratio $r_k = \widehat{T}_\infty(k)/T^\star$ and report the symmetric mean absolute percentage error (SMAPE) together with the geometric mean ratio Q, where $Q = 1$ in-

Table 18: Retrospective accuracy of $\widehat{T}_\infty$ on truncated XML100 default-run logs. Here $r = \widehat{T}_\infty/T^\star$, cov. is the accepted-fit fraction, SMAPE is $2|\widehat{T}_\infty - T^\star|/(\widehat{T}_\infty + T^\star)$ averaged over accepted truncations, $Q = \mathrm{GM}(r)$ measures multiplicative calibration, and med. fit is the median fitting time per truncation.

$t_k/T^\star$	n	cov.	SMAPE	Q	mean r	med r	med. fit (s)
0.50	100	1.000	0.155	0.885	0.898	0.909	0.126
0.60	100	1.000	0.143	0.977	0.993	1.015	0.154
0.70	100	1.000	0.136	1.018	1.033	1.009	0.140
0.80	100	1.000	0.103	1.015	1.026	0.997	0.130
0.90	100	1.000	0.063	1.013	1.018	1.001	0.121

Table 19: Prediction-only ranking at a 600 s cutoff on 11 XML100 instances. All metrics use the same 34 within-instance configuration pairs.

Signal	Correct pairs	Mean within-instance τ_b
Completion estimate $\widehat{T}_\infty$	26/34	0.515
Final gap	25/34	0.485
Logged-gap integral (PDI proxy)	17/34	0.121

dicates agreement on the geometric mean scale. Table 18 reports coverage, accuracy, and median fitting time. SMAPE decreases across the reported cutoffs, with accepted fits for all 100 runs at each cutoff.

G.4 Ranking Configurations Using Predicted Completion Times

The XML100 study uses CVRPLIB logs (Lima et al., 2014) collected with a 1,000 s limit. We retain eight configurations, comprising default, `CFG02`, `CFG03`, `CFG04`, `CFG05`, `CFG07`, `CFG09`, and `CFG10`. The extreme configurations `CFG01`, `CFG06`, and `CFG08` are excluded because they solve only 26, 23, and 24 of the 100 instances, respectively. Ground-truth comparisons use the 49 instances solved by all eight retained configurations, totaling 392 configuration runs. The resulting evidence concerns this common-solved subset.

At the simulated 600 s cutoff, 350 runs already provide observed solve times, while 42 require extrapolation. All 42 cutoff-only fits succeed. Table 19 isolates prediction quality by restricting comparisons to the 11 instances with at least two censored configurations, yielding 34 within-instance configuration pairs. The completion estimate, final logged gap, and a logged-gap integral use the same cutoff and pairs and are compared against true completion time. The integral is a PDI proxy formed by left-step integration of the logged percentage gaps up to the cutoff, using the first visible gap for the initial interval and 100 for missing gaps. It differs from the PDI reported by SCIP and used in the Case 2 objective. We report correctly ordered pairs and mean within-instance Kendall correlation τ_b, which accounts for ties (KENDALL, 1938).

The completion signal has greater observed agreement than the logged-gap integral and is close to final gap. These results support the use of extrapolation for ranking configurations but do not establish significant superiority over final gap. The 350 runs solved before the cutoff are excluded from these prediction-only statistics.

G.5 Validation on Longer Runs

Six prespecified Case 1 runs are evaluated with a 6,000 s limit. Five complete, and one remains right-censored and is excluded from error statistics for completed runs. For each completed run, the simulated cutoff is 60% of its true completion time $T^\star$, and the fit uses only bounds available at that cutoff. All five fits succeed. The geometric mean ratio is $Q = \mathrm{GM}(\widehat{T}_\infty/T^\star) = 0.894$. The symmetric mean absolute percentage error (SMAPE), averaging $2|\widehat{T}_\infty - T^\star|/(\widehat{T}_\infty + T^\star)$, is 0.214,

Table 20: Terminal geometric mean best-so-far losses. Each seed column aggregates 20 instances; "All" aggregates all 100 instance-seed pairs. Lower values are better.

Method	17	29	42	71	97	All
LDST	0.360	0.380	0.403	0.374	0.343	0.371
SMAC3	0.470	0.566	0.547	0.528	0.509	0.523
RBFOPT	0.459	0.565	0.544	0.529	0.503	0.519
RANDOM	0.491	0.578	0.542	0.540	0.512	0.532

Table 21: The 20 instances used in the robustness study. A check mark identifies the eight Case 2 instances used to compare models or the ten instances used to study parameter sensitivity.

Instance	Family	Case	Model-8	Parameters-10
cvrp_P-n22-k2	CVRP	1	–	✓
cvrp_E-n31-k7	CVRP	1	–	–
gap_gapc-5	GAP	1	–	✓
gap_gapc-6	GAP	1	–	–
jsp_la03	JSP	1	–	✓
miplib_neos-827175	MIPLIB	1	–	–
miplib_roll3000	MIPLIB	1	–	✓
miplib_nursesched-sprint02	MIPLIB	1	–	–
ufl_2623UnifS	UFL	1	–	–
ufl_723UnifS	UFL	1	–	✓
ufl_1323UnifS	UFL	1	–	–
ufl_423UnifS	UFL	1	–	–
cvrp_F-n45-k4	CVRP	2	✓	✓
cvrp_F-n72-k4	CVRP	2	✓	–
cvrp_B-n44-k7	CVRP	2	✓	✓
gap_gapd-3	GAP	2	✓	✓
jsp_orb03	JSP	2	✓	✓
jsp_la09	JSP	2	✓	–
miplib_uct-subprob	MIPLIB	2	✓	✓
miplib_nexp-150-20-8-5	MIPLIB	2	✓	–

and eight of ten cross-instance completion-time pairs are ordered correctly. These are descriptive results for five heterogeneous completed cases.

H ROBUSTNESS ACROSS SEEDS AND SENSITIVITY TO RESTARTS

H.1 INSTANCES AND REPEATED TUNING

The robustness study evaluates complete tuning on 20 prespecified instances from five benchmark sets, comprising 12 easy and eight intermediate instances. We select these instances by within-family difficulty under the default run before comparing policies; Table 21 gives the exact instance list. Each instance is evaluated under seeds $\{17, 29, 42, 71, 97\}$ with LDST, SMAC3, RBFOPT, and RANDOM, yielding 400 complete trajectories. Every trajectory includes the default evaluation and 50 evaluated proposals, and best-so-far selection includes trial 0 at every checkpoint. The shared solver, baseline, and parameter-space settings are specified in Appendices B and C.

The SCIP setting `randomization/randomseedshift` and applicable optimizer seeds vary jointly. GPT-5.2 request settings, including medium reasoning effort, remain fixed (Appendix D); a seed for model requests is not guaranteed. Each repetition runs the entire adaptive search, including its sequence of proposals and evaluations. Seed 42 reuses the corresponding main-study trajectories. Table 20 reports terminal losses separately by seed and across all instance-seed pairs. LDST achieves the lowest aggregate loss under every seed.

Table 22: Paired terminal comparisons and instance influence. Ratios below one favor LDST. Wins count matched instance-seed pairs; leave-one-out ranges remove one instance and all its seeds at a time.

Baseline	Paired ratio 95% interval	Wins	Leave-one-out ratio range
SMAC3	0.710 [0.641, 0.786]	88/100	[0.690, 0.732]
RBFOPT	0.716 [0.649, 0.788]	85/100	[0.696, 0.735]
RANDOM	0.698 [0.628, 0.775]	85/100	[0.678, 0.718]

H.2 PAIRED UNCERTAINTY AND INSTANCE INFLUENCE

For each baseline, the comparison aggregates paired LDST/baseline best-so-far loss ratios at trial 50 on the log scale and exponentiates their mean. Terminal 95% intervals use 10,000 hierarchical paired bootstrap resamples. Instances are sampled with replacement within strata defined by family and regime while preserving the original stratum sizes. Within each selected instance, five seed slots are then resampled with replacement. All methods use the same sampled instance-seed pairs, preserving the paired comparison at both levels. Table 22 reports these ratios and intervals together with matched-pair wins.

Figure 2 uses 5,000 hierarchical resamples with the same resampling scheme separately within each regime. Its intervals are pointwise, not simultaneous bands over the tuning horizon. Instance-level comparisons use log loss ratios averaged across the five seeds. On this basis, 18, 17, and 18 of 20 instances favor LDST against SMAC3, RBFOPT, and RANDOM, respectively. Ratios aggregated within each family or regime are also below one against every baseline. The two instances unfavorable to LDST against all three baselines are `miplib_nursesched-sprint02` and `cvrp_B-n44-k7`. All leave-one-instance-out aggregate ratios remain below one (Table 22), so removing any single instance preserves the aggregate ordering. These results concern the instances selected in advance, not a five-seed replication of all 215 benchmark instances.

H.3 RESTARTS UNDER THE SAME EVALUATION BUDGET

Standard LDST attains a lower observed terminal geometric mean loss than Restart10, 0.550 versus 0.576, under the same evaluation budget. Restart10 evaluates a deterministic valid anchor after ten proposals without improving the global incumbent, preserves the global best, and continues semantic editing from the new branch. Anchor evaluations consume the same 50-proposal, 51-total-evaluation budget as standard LDST.

The study covers eight intermediate instances across five seeds. Of 40 planned comparisons, 39 complete matched pairs are available; no result is available for `miplib_uct-subprob` at seed 71. No solver or model failure was reported for the missing run, and its outcome is not imputed. The Restart10/standard ratio is 1.048, with a hierarchical bootstrap 95% interval of $[0.942, 1.191]$ accounting for instance and seed dependence. Restart10 wins 16 pairs and standard LDST wins 23 (sign test, $p = 0.337$).

All 39 Restart10 trajectories restart at least once. Across these runs, 84 branches occur; 39 branches yield a new global incumbent, and 30 trajectories improve after a restart. For the tested budget and restart rule, the observed aggregate favors retaining the standard policy and allocating proposal evaluations to continued semantic refinement.

H.4 PARAMETER MODIFICATION FREQUENCIES

We examine all 10,750 evaluated proposals from the 215-instance main study. For each proposal, we compare the evaluated configuration with the incumbent supplied to the model and assign changed parameters to their SCIP subsystems. The comparison includes returns to default values and accounts for fixed experimental settings. Table 23 reports 80,399 parameter changes across the five subsystems. Cut separation is the most frequent target, appearing in 76.7% of proposals. The allocation also changes with the evaluation regime. Branching and node selection appear in 56.7% and 60.8% of hard-instance proposals, compared with 34.6% and 10.4% on easy instances. The

Table 23: Parameter changes in 10,750 evaluated proposals across the 215-instance main study. Trial percentages can overlap because one proposal can change several subsystems.

Subsystem	Parameter changes	Share (%)	Trials	Share (%)
Cut separation	46,182	57.4	8,250	76.7
Branching	13,125	16.3	4,516	42.0
Node selection	7,118	8.9	3,045	28.3
Primal heuristics	13,686	17.0	2,529	23.5
Presolving	288	0.4	243	2.3

Table 24: Prompt ablation results and paired comparisons. Panel (a) reports geometric mean best-so-far losses and threshold-hit rates. Panel (b) reports full/control ratios with 95% paired instance-bootstrap intervals from 20,000 resamples; ratios below one favor full LDST.

(a) Aggregate outcomes Variant	Final loss	Trial-5 loss	$H_{0.8}$ (%)
Full (LDST)	0.308	0.485	90.8
StaticContext	0.342	0.509	86.8
OutcomeOnly	0.358	0.520	84.2
IndexOnly	0.523	0.707	76.3

(b) Full/control ratios and intervals Trial	StaticContext	OutcomeOnly	IndexOnly
5	0.952 [0.851, 1.055]	0.932 [0.811, 1.073]	0.685 [0.609, 0.770]
10	0.921 [0.821, 1.020]	0.884 [0.770, 1.015]	0.622 [0.546, 0.706]
15	0.943 [0.823, 1.056]	0.883 [0.770, 1.012]	0.597 [0.524, 0.674]
30	0.900 [0.811, 0.988]	0.860 [0.753, 0.976]	0.589 [0.514, 0.670]

observed edits therefore extend beyond cut settings and place greater emphasis on tree search in the hard regime.

The unit of analysis is an evaluated proposal, excluding rejected or unexecuted model responses. A proposal can change several subsystems, so the trial percentages overlap. The experimental procedure fixes `presolving/maxrestarts` at zero throughout the study. Changes requested for that parameter are excluded, leaving six evaluated proposals with no effective change. All other comparisons use the incumbent selected by the tuning loop, including its rule favoring the most recent trial for exact ties.

I Prompt Ablations and Checkpoint Comparisons

All four prompt variants are evaluated on the 76 easy instances. Each uses seed 42, GPT-5.2 with medium reasoning effort, a budget of $K = 30$ proposals, and the experimental edit cap $M = 20$. Validation, re-querying, and novelty checks against previously evaluated configurations follow the shared protocol in Appendix B; Appendix F specifies the prompt structure.

The prompt controls are defined in the LLM Controls subsection of Appendix B. Each control removes several related inputs together, so the comparisons do not isolate individual causal effects.

Best-so-far selection includes the default at every checkpoint. Table 24 reports geometric mean losses and $H_{0.8}$, the percentage reaching best-so-far loss at most 0.8 by trial 30. For each checkpoint and control, we compute 95% intervals using 20,000 paired instance-bootstrap resamples of the 76 instances. Within each resample, full/control log loss ratios are averaged and exponentiated, preserving method pairing within each instance.

Full LDST and OutcomeOnly complete all 30 proposals on every instance. The StaticContext run on `miplib_cbs-cta` stops at trial 26, and six IndexOnly trajectories stop at trials 16–23. We carry their observed incumbents forward to subsequent nominal checkpoints while treating

Table 25: Matched model usage and cost on eight complete trajectories per model. Entries are per-trajectory medians; monetary estimates use the July 23, 2026 uncached list price schedule.

Quantity	GPT-5.2	GPT-5-mini
API calls	51	53.5
Input tokens	230,581	205,560
Output tokens	108,902	150,408
Uncached list cost (USD)	1.9253	0.3457

the searches as incomplete. Trial 15 precedes all seven terminations, permitting a comparison before carry-forward affects the results.

The full/OUTCOMEONLY intervals include one at trials 5, 10, and 15 and exclude one at trial 30. At trial 30, full LDST wins 49 instances and loses 27 (exact two-sided sign test, $p = 0.0154$).

J MODEL CHOICE, COST, AND PARAMETER SENSITIVITY

J.1 MATCHED MODEL AND COST COMPARISON

The model comparison uses eight prespecified intermediate instances from four benchmark sets, seed 42, and 50 proposals after the default. Both LLM variants and all three scalar baselines complete 51 evaluations per instance. Terminal geometric mean losses are 0.591 for GPT-5.2 and 0.652 for GPT-5-mini, compared with 0.771, 0.776, and 0.790 for SMAC3, RBFOPT, and RANDOM. Both models therefore retain an aggregate advantage over the scalar methods on the same eight instances.

Table 25 reports usage and cost for these same eight complete trajectories per model. Each entry is a median over trajectories; the number of requests includes validation re-queries and need not equal the number of solver evaluations. Monetary estimates apply the price schedule dated July 23, 2026, using uncached list rates for both models. The median model cost for GPT-5-mini is 82.0% lower; accounting for cached tokens gives its secondary median cost of USD 0.3365.

GPT-5-mini records median SCIP time of $30{,}600\,\mathrm{s}$ and direct API time of $1{,}753.9\,\mathrm{s}$. GPT-5.2 request timing is unavailable for the paired seed-42 trajectories, preventing a matched latency or end-to-end time comparison.

J.2 PROPOSAL VALIDITY AND PARAMETER SENSITIVITY

A replay study uses 100 prompts from the set of 20 instances at trials $\{1, 10, 20, 30, 40\}$, with five repeated calls per prompt and model. The 500 calls per model yield first-attempt valid proposal rates of 99.0% for GPT-5.2 and 97.4% for GPT-5-mini. The model versions are `gpt-5.2-2025-12-11` and `gpt-5-mini-2025-08-07`, with medium reasoning effort, a 20,000-token output cap, and JSON-object responses; temperature and top-p use provider defaults. Validation checks the required format, parameter types and domains, edit cap, no-op edits, and novelty against the evaluated configurations in the corresponding trial history. This analysis does not run the solver or test whether SCIP accepts the settings. The replay measures response validity, not optimization quality or repeatability of a complete tuning trajectory.

The parameter sensitivity study uses ten instances, five easy and five intermediate, with seed 42, four methods, and 50 proposals. Curated-45 is defined in Appendix C. Expanded-90 adds 45 parameters, nine each from branching, node selection, separation, presolving, and primal heuristics. A lexicographic round-robin over plugins, defined independently of outcomes, applies the specified exclusions and a check that settings load successfully under SCIP 9.2.4. The domains of existing parameters and the evaluation budget remain unchanged, and every method uses the same space within each comparison. The anonymous analysis supplement provides the added parameter definitions and selection details. With the default configuration eligible for selection, LDST loss changes from 0.461804 to 0.462177, a relative increase of 0.081%, although both round to 0.462. In the order SMAC3, RBFOPT, and RANDOM, scalar losses are 0.580, 0.593, and 0.559 in Curated-45

Table 26: Preliminary CBC results on eight matched instances. Entries are geometric means of best-so-far normalized PDI, with the default configuration eligible for selection; lower values are better.

Method	Trial 10	Trial 20
LDST	**0.698965**	**0.662117**
SMAC3	0.747482	0.678862
RBFOPT	0.816635	0.790759
RANDOM	0.892970	0.786098

and 0.749, 0.743, and 0.770 in Expanded-90. The aggregate loss of LDST changes little under the tested expansion.

K SOLVER PORTABILITY AND LOCALLY RUN MODELS

K.1 CBC INTERFACE TRANSFER

The preliminary CBC study evaluates the refined solver interface using CBC 2.10.11, a $600\,\mathrm{s}$ time limit, and seed 42. All four methods use the same 16 tunable parameters and a budget of 20 proposals after the default. We normalize the $600\,\mathrm{s}$ PDI by its default value and include trial 0 in best-so-far selection. The LDST runs use GPT-5.2 with medium reasoning effort.

The set of eight instances spans four benchmark sets and comprises `cvrp_B-n44-k7`, `cvrp_F-n45-k4`, `cvrp_F-n72-k4`, `gap_gapd-3`, `jsp_la09`, `jsp_orb03`, `miplib_nexp-150-20-8-5`, and `miplib_uct-subprob`. The CBC diagnostic guide was derived from the best configurations found by the three scalar baselines on these same eight instances; the baseline trajectories and shared default evaluations are reused.

Table 26 reports the geometric mean losses at trials 10 and 20. LDST has the lowest loss at both checkpoints, reducing loss relative to SMAC3, the best scalar baseline, by 6.49% and 2.47%, respectively.

K.2 LOCAL QWEN3-8B CONFIGURATION

Local Qwen3-8B execution through `llama.cpp` supports the LDST model interface and produces useful early solver edits without a hosted model service. All 51 model requests return responses, and 24 edit proposals pass validation, including 22 on the first attempt. Four of the eight instances improve during the observed early search.

We use `Q4_K_M` quantization with reasoning disabled, an 8,192-token context window, and a 2,048-token output limit. Initial calls are deterministic; retries use temperature 0.7 and include the previous response and validation feedback, with at most three total attempts. Five trajectories stop after repeated non-novel proposals exhaust the attempt limit, and three stop after no-op edits. Table 27 reports the accepted proposals and observed evaluations, including completion relative to the planned budget.

L ADDITIONAL RESULTS, TRANSFER, AND SENSITIVITY IN THE HARD REGIME

L.1 SOLVER PERFORMANCE AND CONVERGENCE ON THE FULL BENCHMARK

For full-suite reporting, every method includes default trial 0 and selects the representative configuration in chronological order, retaining the earliest trial when losses differ by at most 10^{-12}. The reporting convention differs from the exact-tie rule used for prompt construction.

The selected configurations in Table 2 also permit direct comparisons of solver outcomes. Table 3 reports time on the 75 easy instances solved to optimality by all four selected configurations and final gap on the 90 intermediate instances with finite values under all four methods. Final gap is the

Table 27: Execution outcomes for the tested local Qwen3-8B configuration. The number of solver evaluations includes the default run.

Quantity	Outcome
Instances with early improvements	4/8
Validated edit proposals	24
Proposals valid on the first attempt	22
Requests receiving responses	51/51
Instances evaluated	8
Observed solver evaluations	32
Evaluations per trajectory	2–12
Trajectories reaching the planned 51 evaluations	0/8

percentage reported in the terminal `Gap` field of the SCIP log; the diagnostic feature `gap_final` instead records the absolute difference between primal and dual bounds. The shifted geometric mean is $\exp\{n^{-1}\sum_i \log(x_i + a)\} - a$, with shift $a = 1\,\mathrm{s}$ for time and $a = 0.01$ percentage points for gap. Optimal-solution coverage is $75/76$ for every method; the remaining instance is proved infeasible.

Figure 8 shows the full-suite, seed-42 convergence curves separately by regime. These curves use the same best-so-far selection as the primary table, with the default configuration included. The Case 3 display transform magnifies deviations from one and does not change the reported loss values.

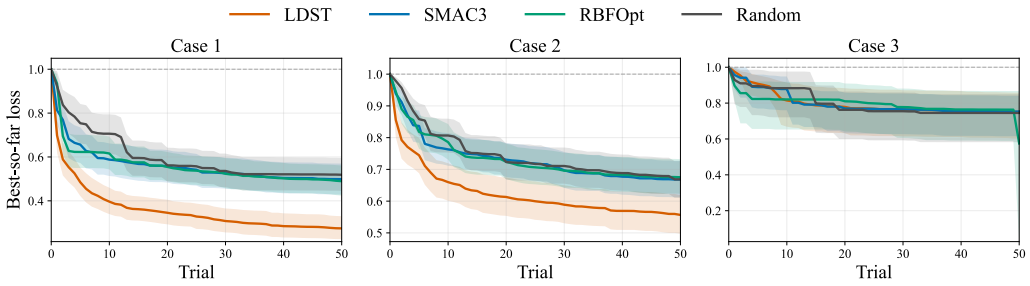

Figure 8: Best-so-far loss in Exp. A. For each regime c and trial t, the shaded band is computed by drawing 2000 bootstrap samples of size $|\mathcal{I}_c|$ from $\mathcal{I}_c$ with replacement. For each bootstrap sample, we recompute the plotted aggregate best-so-far loss at trial t using the same formula as for the main curve. The lower and upper edges of the band are the 2.5th and 97.5th percentiles of these 2000 bootstrap values at that trial. The Case 3 panel applies the transform $\widetilde{L}^{(3)} = 1 + 36(L^{(3)} - 1)$.

L.2 Exp. C: Transfer within Problem Families

C1: Transfer from a single anchor. C1 tests whether a configuration tuned on one instance transfers within its benchmark family. We use 55 same-family anchor splits from the Case 1 instances in CVRP, GAP, JSP, and UFL. Anchor selection uses the original training objectives, retained at the original precision for LDST and at two decimal places for the scalar baselines. The default remains eligible. The 209 selected proposals are applied unchanged to the remaining Case 1 instances in their families; the 11 default selections use the target-normalized reference loss of one. Table 28 reports training scores at the available solver-summary precision for these fixed settings. LDST has the lowest aggregate anchor and test losses and leads in all four family-level test comparisons.

C2: Multi-instance tuning versus independent selection. C2 compares independent tuning followed by candidate selection with joint tuning on the full training subset when multiple instances are available. For each eligible benchmark set s and train size $m \in \{2, 3, 4, 5, 6\}$, we construct five deterministic Case 1 train/test splits whenever s contains at least $m + 4$ Case 1 instances, ensuring at least four held-out test instances. Because eligibility changes with m and across regimes, pooled aggregates should be read with the changing family mix in mind.

Table 28: C1 transfer on 55 Case 1 anchors. Configurations are selected using the training objectives retained from the original runs and applied without further tuning. Entries are geometric means over anchor splits. Gap denotes $L_{\text{test}} - L_{\text{train}}$, computed before rounding. The last four columns report test loss by family.

	Overall			Family-level Case 1 test			
Method	Case 1 train	Case 1 test	Gap ↓	CVRP	GAP	JSP	UFL
LDST	**0.238**	**0.499**	**0.261**	**1.190**	**0.734**	**0.262**	**0.385**
SMAC3	0.429	0.799	0.370	2.067	0.966	0.657	0.565
RBFOPT	0.427	0.798	0.370	1.978	1.078	0.716	0.547
RANDOM	0.449	0.838	0.390	1.866	1.313	0.635	0.601

For each split, independent tuning optimizes each training instance separately and selects the candidate with the smallest aggregated training loss; joint tuning optimizes directly on the full training split. The selected configuration is evaluated on the held-out Case 1 test set and, when available, on same-family Case 2 and Case 3 instances. Table 29 compares the two strategies metric by metric. At $m = 2$ and $m = 3$, joint tuning lowers both Case 1 train and test loss and also improves the harder-regime columns, though with a wider train-test gap, from 0.150 to 0.226 at $m = 2$ and 0.099 to 0.143 at $m = 3$. At $m = 4$, Case 1 test losses are essentially tied (0.560 versus 0.561), while joint tuning still improves training loss and the Case 2–3 columns. At $m = 5, 6$, independent tuning has the lower pooled Case 1 test loss.

At $m = 2, 3$, the wider train-test gap accompanies lower absolute test loss under joint tuning. Paired sign tests on Case 1 test loss do not reject parity for any m (all $p \geq 0.11$). The benchmark-specific results in Appendix L show that neither strategy dominates across families.

Table 29: Within-class generalization in Exp. C. Each metric reports Independent and Joint tuning; Gap is $L_{\text{test}} - L_{\text{train}}$. At $m = 2, 3$, Case 1 uses all four families, Case 2 uses CVRP/GAP/JSP, and Case 3 uses CVRP only. At $m = 4, 5, 6$, Case 1 uses CVRP/UFL, while Case 2–3 use CVRP only.

m	Case 1 train		Case 1 test		Gap ↓		Case 2		Case 3	
	Indep.	Joint	Indep.	Joint	Indep.	Joint	Indep.	Joint	Indep.	Joint
2	0.390	**0.282**	0.540	**0.508**	**0.150**	0.226	0.919	**0.849**	1.010	**1.002**
3	0.396	**0.325**	0.495	**0.468**	**0.099**	0.143	0.864	**0.828**	1.018	**1.011**
4	0.421	**0.395**	**0.560**	0.561	**0.139**	0.166	0.715	**0.689**	1.025	**1.007**
5	**0.389**	0.435	**0.543**	0.625	**0.154**	0.190	**0.687**	0.775	1.030	**1.008**
6	0.467	**0.376**	**0.474**	0.508	**0.007**	0.132	**0.705**	0.720	1.009	**1.008**

Exp. C results by split and benchmark. Table 30 reports benchmark-specific Case 1 test losses for every evaluated train size, while Table 31 reports the representative $m = 3$ transfer breakdown across regimes. CVRP consistently favors joint tuning across all five train sizes, whereas UFL consistently favors independent tuning across all five. GAP favors joint tuning on both eligible sizes. For JSP, independent tuning is slightly better at $m = 2$, whereas joint tuning is better at $m = 3$.

Case 3 sensitivity to individual instances. With the default eligible for every method, the geometric mean losses over all 48 instances are 0.993083, 0.993138, 0.988193, and 0.992921 for LDST, SMAC3, RBFOPT, and RANDOM, respectively. The largest difference between LDST and RBFOPT is 0.2556 on `miplib_irish-electricity`, in favor of RBFOPT (Table 32). The losses on that instance are 0.953325, 0.921758, 0.697736, and 0.899144 in the same method order. The second-largest difference, 0.0785 on `miplib_neos-1354092`, favors LDST; every remaining difference is below 0.01. Omitting only `miplib_irish-electricity` gives 47-instance geometric means of 0.993946, 0.994715, 0.995538, and 0.995019, respectively, making LDST best in this sensitivity check. The full-set lead of RBFOPT therefore depends on its large improvement on that instance. Aggregate loss reflects the magnitude of improvements, whereas coverage counts improved instances equally. Table 2 retains all 48 instances.

Table 30: Benchmark-specific Case 1 test loss in Exp. C for every eligible benchmark/train-size pair. Each entry is the geometric mean of split-level Case 1 test losses for the corresponding benchmark, strategy, and train size; – indicates that the family is not eligible for that train size under the rule requiring four held-out instances. Lower is better.

Benchmark	$m = 2$		$m = 3$		$m = 4$		$m = 5$		$m = 6$	
	Indep.	Joint	Indep.	Joint	Indep.	Joint	Indep.	Joint	Indep.	Joint
CVRP	1.122	**0.796**	1.014	**0.874**	0.985	**0.885**	1.028	**0.877**	0.769	**0.675**
GAP	0.775	**0.673**	0.841	**0.680**	–	–	–	–	–	–
JSP	**0.265**	0.325	0.217	**0.204**	–	–	–	–	–	–
UFL	**0.369**	0.382	**0.323**	0.395	**0.318**	0.355	**0.287**	0.445	**0.292**	0.382

Table 31: Benchmark-specific transfer summary for Exp. C at $m = 3$. The columns report Case 1 test, same-family Case 2, and same-family Case 3 losses for each strategy. Each entry is the geometric mean over the repeated splits for that benchmark and strategy; – indicates that the benchmark family has no held-out target instances in that regime. Lower is better.

Benchmark	Independent Case 1	Joint Case 1	Independent Case 2	Joint Case 2	Independent Case 3	Joint Case 3
CVRP	1.014	**0.874**	0.776	**0.749**	1.018	**1.011**
GAP	0.841	**0.680**	1.030	**0.913**	–	–
JSP	0.217	**0.204**	**0.807**	0.830	–	–
UFL	**0.323**	0.395	–	–	–	–

M Supplementary Material and Computational Settings

Anonymous analysis supplement. The supplement provides per-instance results and convergence histories for the full suite, five-seed and prompt-ablation histories, model cost data, the diagnostic guides, and subsystem edit data. The command `python3 scripts/verify_results.py` checks file hashes and reconstructs the principal geometric means, paired comparisons, sensitivity analyses, and edit frequencies. The full-suite scores use the available precision in solver summaries. Three default completion estimates for `jsp_la05` retain six significant digits; the supplied rounding intervals leave every displayed aggregate and percentage improvement unchanged. An R script regenerates the three five-seed and ablation figures. Bootstrap intervals are supplied precomputed. The package supports offline numerical verification and excludes full execution code, raw solver/-model logs, and third-party benchmark instances.

Licenses and data sources. The author-created analysis code and documentation in the supplement use the MIT license. Benchmark source citations, upstream URLs, instance lists, and checksums identify the public collections described in Appendix A; benchmark files remain subject to their upstream terms and are not redistributed in this package. We plan to release the full LDST implementation under the MIT license after de-anonymization. Table 33 reports the hardware, software, and token usage needed for reproducibility.

Computational budget. We report per-run budgets in the main text and token usage for the main study with seed 42 above. Each solver call uses one thread and finishes before the next call. The nominal Exp. A budget is one default run at trial 0 followed by up to 50 proposal evaluations, and the nominal Exp. B budget is trial 0 followed by up to 30 proposals. Trial 0 does not consume the proposal budget. The Exp. B histories that exhaust the validation and query-attempt loop end early, and their last observed incumbents are carried forward only for analysis. Exp. C uses the same proposal budget per training run under independent and joint tuning, with additional calls for validation samples determined by the fixed split definitions. The set of historical runs mixes exploratory and final paper runs, and many solver runs terminate before the 600-second cap, so it does not support a single total elapsed time for solver execution. We report the environment, proposal budgets, and token usage.

Table 32: Largest instance-level differences between LDST and RBFOPT under best-so-far selection including the default configuration on the fixed set of 48 Case 3 instances. Here $\Delta = L_{LDST} - L_{\text{RBFOPT}}$. Positive Δ means that RBFOPT has lower loss; negative Δ means that LDST has lower loss. The table reports the original Case 3 loss without display magnification.

Instance	LDST	SMAC3	RBFOPT	RANDOM	Δ	$\lvert\Delta\rvert$
`miplib_irish-electricity`	0.9533	0.9218	0.6977	0.8991	$+0.2556$	0.2556
`miplib_neos-1354092`	0.9215	0.9863	1.0000	1.0000	-0.0785	0.0785
`cvrp_B-n51-k7`	0.9922	0.9804	0.9855	0.9860	$+0.0067$	0.0067
`cvrp_B-n45-k6`	0.9942	0.9893	0.9894	0.9895	$+0.0048$	0.0048
`cvrp_A-n62-k8`	0.9964	0.9918	0.9919	0.9918	$+0.0046$	0.0046
`cvrp_P-n50-k10`	0.9919	0.9959	0.9960	0.9960	-0.0041	0.0041
`cvrp_B-n67-k10`	0.9964	0.9920	0.9923	0.9919	$+0.0041$	0.0041
`cvrp_P-n65-k10`	0.9927	0.9958	0.9967	0.9954	-0.0041	0.0041
`cvrp_A-n63-k9`	0.9953	0.9993	0.9993	0.9993	-0.0040	0.0040
`cvrp_A-n69-k9`	0.9972	0.9941	0.9932	0.9936	$+0.0040$	0.0040

Table 33: Hardware, software, and token usage for the main study with seed 42.

Item	Value
Operating system	Linux (Ubuntu; kernel 6.14.0-33-generic; x86_64)
CPU	13th Gen Intel(R) Core(TM) i7-13700K (24 logical CPUs)
System memory	62 GiB total RAM
Python	3.12.4 (`python` and `python3`)
SCIP	9.2.4
SMAC3	2.3.1
Threads per solver run	1
Concurrent instance jobs	1
SCIP clock in the main bank	CPU time
Main-bank solver seed	`randomization/randomseedshift = 42`
Main-bank LLM model	GPT-5.2
Total LLM calls in the A1 bank	10,964
Total tokens in the A1 bank	62,958,576
Approx. uncached list-price cost	USD 320.08
Invalid-JSON warnings in the A1 bank	69

Seed convention. The original 215-instance main study fixes `randomization/randomseedshift` to 42, and its stochastic black-box baselines use optimizer seed 42. The prespecified five-seed follow-up is reported separately in Appendix H; it does not replace the full-suite seed-42 estimates. The LLM policy otherwise uses a fixed model and fixed API settings. Residual stochasticity in API responses remains outside the solver-seed comparison.

N BENCHMARK INSTANCES AND FORMULATIONS

We provide the family-specific formulations, source breakdowns, and benchmark instance lists underlying Appendix A. The evaluation bank contains 215 instances, comprising 90 CVRP, 12 GAP, 24 JSP, 30 UFL, and 59 MIPLIB instances. The preceding benchmark appendix states the source collections, construction procedure, regime assignment, and split protocol used by the experiments; this section provides the detailed family-level formulations and source tables.

N.1 CVRP: CAPACITATED VEHICLE ROUTING PROBLEM

Problem definition. The CVRP seeks a minimum-cost set of vehicle routes that start and end at a depot, serve each customer exactly once, and satisfy the vehicle-capacity limit Q. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

be a complete undirected graph with two depot copies, $\mathcal{V} = \{0, 1, \ldots, n, n{+}1\}$, and customer set $\mathcal{C} = \mathcal{V} \setminus \{0, n{+}1\}$. Each customer i has demand d_i, and K vehicles are available.

Formulation. We use the two-commodity flow formulation of Baldacci et al. (2004) with binary edge variables x_{ij} and continuous flow variables $y_{ij} \geq 0$, given by

$$
\begin{aligned}
\min \quad & \sum_{\{i,j\} \in \mathcal{E}} c_{ij}\, x_{ij} \\
\text{s.t.} \quad & \sum_{j \in \mathcal{V} \setminus \{i\}} \big(y_{ji} - y_{ij}\big) = 2d_i, \quad \forall i \in \mathcal{C}, \\
& \sum_{j \in \mathcal{C}} y_{0j} = d(\mathcal{C}), \\
& \sum_{j \in \mathcal{C}} y_{j0} = KQ - d(\mathcal{C}), \\
& \sum_{j \in \mathcal{C}} y_{n+1,j} = KQ, \\
& y_{ij} + y_{ji} = Q\, x_{ij}, \qquad \forall \{i,j\} \in \mathcal{E}, \\
& \sum_{\substack{j \in \mathcal{V} \\ j > i}} x_{ij} + \sum_{\substack{j \in \mathcal{V} \\ j < i}} x_{ji} = 2, \qquad \forall i \in \mathcal{C}, \\
& y_{ij} \geq 0,\ y_{ji} \geq 0, \qquad \forall \{i,j\} \in \mathcal{E}, \\
& x_{ij} \in \{0,1\}, \qquad \forall \{i,j\} \in \mathcal{E}.
\end{aligned}
$$

Here $d(\mathcal{C}) = \sum_{i \in \mathcal{C}} d_i$. We use all 90 CVRPLIB instances listed in Table 34.

Instance selection. We draw CVRP instances from the `A`, `B`, `E`, `F`, and `P` families in CVRPLIB (Lima et al., 2014), originating from Augerat et al. (1995) (sets A/B/P), Christofides & Eilon (1969) (set E), and Fisher (1994) (set F). CVRPLIB instance names follow the format `<set>-n<`N`>-k<`K`>`, where N is the number of nodes including the depot and K is the number of routes in the reference solution. In our implementation, for an instance with N nodes, we set $n := N - 1$ customers and duplicate the depot as nodes 0 and $n{+}1$, while fixing the fleet-size parameter to K. The full 90-instance set is summarized by the construction rules above and the numbers of instances in Table 34.

Table 34: Summary of CVRP instance families available in CVRPLIB.

Set	#	Range of N	Source
`A`	27	32–80	Augerat et al. (1995)
`B`	23	31–78	Augerat et al. (1995)
`E`	13	13–101	Christofides & Eilon (1969)
`F`	3	45–135	Fisher (1994)
`P`	24	16–101	Augerat et al. (1995)
Total	**90**		

N.2 GAP: Generalized Assignment Problem

Problem definition. Each job $j \in \mathcal{J}$ must be assigned to exactly one agent $i \in \mathcal{I}$. Assigning j to i consumes resource a_{ij} from capacity b_i and incurs cost c_{ij}. The objective is to minimize the total assignment cost (Ross & Soland, 1975).

Formulation.

$$\begin{aligned} \min \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}\, x_{ij} \\ \text{s.t.} \quad & \sum_{i\in\mathcal{I}} x_{ij} = 1, && \forall j \in \mathcal{J}, \\ & \sum_{j\in\mathcal{J}} a_{ij}\, x_{ij} \le b_i, && \forall i \in \mathcal{I}, \\ & x_{ij} \in \{0,1\}, && \forall i \in \mathcal{I},\ \forall j \in \mathcal{J}. \end{aligned}$$

We use 12 OR-Library instances from `gapc` and `gapd`.

Instance selection. We use the OR-Library GAP collection (Beasley, 1990), specifically the large-scale type-C and type-D instances introduced by Chu & Beasley (1997) and distributed as `gapc` and `gapd`. Each file contains six minimization instances with $|\mathcal{I}| \in \{5, 10, 20\}$ and $|\mathcal{J}| \in \{100, 200\}$. Instances are referenced by file name and within-file index (e.g., `gapc-3`). Table 35 summarizes the two GAP source files and their size grid.

Table 35: Summary of GAP instances used in our experiments.

File	#	Sizes ($\lvert\mathcal{I}\rvert \times \lvert\mathcal{J}\rvert$)	Source
`gapc`	6	$\{5, 10, 20\} \times \{100, 200\}$	Chu & Beasley (1997)
`gapd`	6	$\{5, 10, 20\} \times \{100, 200\}$	
Total	**12**		

N.3 JSP: Job Shop Scheduling Problem

Problem definition. Each job consists of an ordered sequence of operations. Operation o has processing time p_o on a required machine $\mu(o)$. Machines can process at most one operation at a time, operations are non-preemptive, and the objective is to minimize the makespan $C_{\max}$ (Manne, 1960).

Formulation. Let $\mathcal{P}$ be within-job precedence arcs and $\mathcal{D}$ the set of unordered machine-conflict pairs. With start times s_o and binary order variables $\delta_{oo'}$ for $(o, o') \in \mathcal{D}$, the standard disjunctive MILP takes the form

$$\begin{aligned} \min \quad & C_{\max} \\ \text{s.t.} \quad & s_{o'} \ge s_o + p_o, && \forall (o,o') \in \mathcal{P}, \\ & s_o + p_o \le s_{o'} + H\,(1 - \delta_{oo'}), && \forall (o,o') \in \mathcal{D}, \\ & s_{o'} + p_{o'} \le s_o + H\,\delta_{oo'}, && \forall (o,o') \in \mathcal{D}, \\ & C_{\max} \ge s_o + p_o, && \forall o, \\ & s_o \ge 0, && \forall o, \\ & \delta_{oo'} \in \{0,1\}, && \forall (o,o') \in \mathcal{D}. \end{aligned}$$

We set a valid horizon $H = \sum_o p_o$ and use 24 OR-Library instances from the `jobshop1` collection (families `abz`, `ft`, `la`, `orb`).

Instance selection. We use 24 benchmark JSP instances from the OR-Library `jobshop1` collection (Beasley, 1990), spanning the `abz`, `ft`, `la`, and `orb` families (Adams et al., 1988; Fisher & Thompson, 1963; Lawrence, 1984; Applegate & Cook, 1991). Table 36 gives the corresponding numbers of instances per family, size ranges, and sources.

Table 36: Summary of the JSP instances used in our experiments.

Family	#	Range of $(\lvert\mathcal{J}\rvert, \lvert\mathcal{M}\rvert)$	Source
`abz(5-9)`	5	(10–20, 10–15)	Adams et al. (1988)
`ft(10,20)`	2	(10–20, 5–10)	Fisher & Thompson (1963)
`la(01-12)`	12	(10–20, 5)	Lawrence (1984)
`orb(01-05)`	5	(10, 10)	Applegate & Cook (1991)
Total	**24**		

N.4 UFL: Uncapacitated Facility Location Problem

Problem definition. Facilities incur fixed opening costs, and each customer must be assigned to an open facility (Kuehn & Hamburger, 1963).

Formulation.

$$
\begin{aligned}
\min \quad & \sum_{j\in\mathcal{F}} f_j\, y_j \;+\; \sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{F}} c_{ij}\, x_{ij} && \\
\text{s.t.} \quad & \sum_{j\in\mathcal{F}} x_{ij} = 1, && \forall i \in \mathcal{C}, \\
& x_{ij} \le y_j, && \forall i \in \mathcal{C},\ \forall j \in \mathcal{F}, \\
& x_{ij} \in \{0,1\}, && \forall i \in \mathcal{C},\ \forall j \in \mathcal{F}, \\
& y_j \in \{0,1\}, && \forall j \in \mathcal{F}.
\end{aligned}
$$

We use 30 UflLib instances from the Uniform family.

Instance selection. We use uncapacitated facility location instances from UflLib (Hoefer, 2018). Following our benchmark protocol, we include the complete Uniform family by taking all 30 `*UnifS` instances. We exclude `*EuclS` instances, which are typically solved at the root under default settings and therefore provide limited tuning signal. Table 37 summarizes the UFL family retained in the evaluation bank.

Table 37: Summary of UFL instances used in our experiments.

Family	#	$(\lvert\mathcal{F}\rvert, \lvert\mathcal{C}\rvert)$	Source
`*UnifS`	30	(100, 100)	Hoefer (2018)
Total	**30**		

N.5 General MILP Benchmark: MIPLIB 2017

Benchmark description. To complement the domain-specific models, we use instances from the official MIPLIB 2017 benchmark set (Gleixner et al., 2021). MIPLIB 2017 is the sixth release of the library and was compiled through a data-driven selection process. From an initial pool of 5,721 submissions, it provides a collection of 1,065 instances and a curated benchmark subset of 240 instances for solver evaluation (`benchmark-v2`).

Instance selection. We uniformly sample 60 instances from the official MIPLIB 2017 benchmark subset with a fixed random seed (42). After testing, we exclude `k1mushroom` because, under default SCIP settings with a time limit of 600s, it does not complete presolving. The final experimental subset therefore contains 59 instances, listed in Table 38.

Table 38: Final 59-instance subset sampled from the official MIPLIB 2017 benchmark-v2 subset and used in our experiments, after removing `k1mushroom` because it does not finish presolving under the default 600-second run.

Final selected instances (`benchmark-v2`)		
50v-10	assign1-5-8	atlanta-ip
b1c1s1	beasleyC3	cbs-cta
chromaticindex512-7	cmflsp50-24-8-8	CMS750_4
cod105	comp21-2idx	cryptanalysiskb128n5obj16
dano3_3	drayage-25-23	dws008-01
fhnw-binpack4-48	gfd-schedulen180f7d50m30k18	glass4
glass-sc	gmu-35-50	h80x6320d
irish-electricity	lectsched-5-obj	momentum1
mushroom-best	mzzv11	n3div36
neos8	neos859080	neos-662469
neos-827175	neos-911970	neos-1354092
neos-1445765	neos-3046615-murg	neos-3083819-nubu
neos-3627168-kasai	neos-3988577-wolgan	neos-5052403-cygnet
nexp-150-20-8-5	nursesched-sprint02	p200x1188c
piperout-27	radiationm18-12-05	radiationm40-10-02
ran14x18-disj-8	rocI-4-11	rocII-5-11
roll3000	s100	snp-02-004-104
supportcase22	supportcase40	supportcase42
swath1	thor50dday	traininstance2
uct-subprob	unitcal_7	