CONFIGAGENT: AGENTIC ALGORITHM CONFIGURATION FOR COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Algorithm configuration (AC) is essential in combinatorial optimization (CO), yet existing methods such as SMAC3 and irace remain context-specific, and need to repeatedly tune for each solver-problem pair from scratch, limiting transferability and efficiency. We propose ConfigAgent, an agentic framework that reformulates AC as knowledge-driven reasoning over accumulated configuration experience across multiple contexts. It maintains a structured knowledge base containing historical trials and distilled insights across different solvers, problem families, objectives, and instance scales. For a new configuration context at deployment, ConfigAgent retrieves relevant prior experience from the knowledge base, reasons over contextual similarities, and generates evidence-grounded configurations. We evaluate ConfigAgent on five CO families with a wide range of algorithms including exact solvers, heuristic search engines, and multi-objective evolutionary algorithms. Results show that ConfigAgent consistently matches strong AC baselines while requiring substantially fewer configuration trials. Moreover, the learned knowledge base enables efficient zero-shot or few-shot transfer to unseen scales, objectives, and structural variants, significantly reducing deployment cost by up to two orders of magnitude.

1 INTRODUCTION

Combinatorial optimization (CO) problems are ubiquitous in real-world applications such as logistics, manufacturing, and resource allocation (Korte & Vygen, 2018). It is well known that the performance of CO algorithms is highly sensitive to parameter configurations, both for exact solvers such as Gurobi (Gurobi Optimization, LLC, 2024) and for metaheuristics such as LKH-3 (Helsgaun, 2017), NSGA-II (Deb et al., 2002), NSGA-III (Deb & Jain, 2014), and MOEA/D (Zhang & Li, 2007). In practice, default parameter configurations are rarely optimal and may not use the full potential of CO algorithms. Tailoring parameters to a specific problem context therefore motivates algorithm configuration (AC).

Classical AC methods such as SMAC3 and irace treat configuration as black-box optimization (Hutter et al., 2011; Lindauer et al., 2022; López-Ibáñez et al., 2016). Despite their effectiveness, several limitations exist. First, the black-box optimization is time-consuming. Second, interpretability is low, and it is hard to understand why the resulting configuration works. Finally, in their standard use, they perform each configuration run independently and do not retain explanations of past results in a form that can be reused in new contexts. When the context (e.g., problem type, instance size, or the optimization objective) changes, prior configuration history is typically not reused unless an explicit transfer mechanism is added (Schede et al., 2022; Franzin & Stützle, 2020). AC thus often remains a process of independent tuning rather than cumulative learning over reusable, interpretable configuration knowledge.

Recent advances in large language models (LLMs) have reshaped CO research in multiple ways, including formulating optimization models (AhmadiTeshnizi et al., 2024), generating novel heuristics (Romera-Paredes et al., 2024; Ye et al., 2024; Hu & Zhang, 2025; Jiang et al., 2025b), and acting as end-to-end solvers (Jiang et al., 2025a; Chen et al., 2026). However, most of these efforts focus on *building* algorithms, while the critical issue of how to *configure* existing or generated algorithms to optimize performance is largely ignored. Applications of LLM have recently been explored in several related tasks such as online parameter control (Xu et al., 2025), hyperparameter optimization (HPO)

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.

in machine learning (Liu et al., 2025; 2024), and combined algorithm selection and hyperparameter optimization (CASH) (Xu et al., 2026). However, they are subtasks or special cases of AC (see Figure 1 in (Schede et al., 2022)), and the power of LLMs in rapidly providing interpretable and transferable configurations for general CO algorithms remains largely untapped.

This paper proposes **ConfigAgent**, an LLM-based agentic framework for configuration across different problems, instance scales, solvers (exact and metaheuristic), and objectives (single and multi-objective). It maintains a shared *knowledge base* of measured configuration records and textual summaries indexed by five context fields: solver class, solution encoding, problem family, objective count, and instance scale. During training, the agent reads local trial history, parameter-space coverage, and parameter direction signals to propose configurations and record short summaries of the outcomes. During deployment, it retrieves records from related contexts. Zero-shot deployment ranks the measured records without a target-side LLM call, whereas few-shot deployment uses the retrieved records to guide a small number of additional trials. This design complements numerical warm starts and surrogate models by adding scope-tagged textual summaries and evidence traces. In the reported settings, it lowers target-side configuration time while maintaining competitive configuration quality.

We evaluate ConfigAgent on five representative CO families, namely traveling salesman problem (TSP), knapsack problem (KP), bin packing problem (BPP), flexible job shop scheduling problem (FJSP), and capacitated vehicle routing problem (CVRP), at various instance scales and objective counts. The algorithm portfolio spans commonly used solvers such as Gurobi (Gurobi Optimization, LLC, 2024), OR-Tools (Perron & Furnon, 2025), and LKH-3 (Helsgaun, 2017), and well-established metaheuristics such as NSGA-II (Deb et al., 2002), NSGA-III (Deb & Jain, 2014), and MOEA/D (Zhang & Li, 2007). In the reported settings, ConfigAgent generally matches and improves upon the evaluated AC baselines while using fewer configuration trials. It also records the source evidence used by LLM-generated proposals, which makes those decisions easier to inspect.

2 Related Work

Algorithm Configuration in Combinatorial Optimization. Classical AC methods treat parameter tuning as a black-box optimization problem. ParamILS (Hutter et al., 2009) uses iterated local search in the configuration space, SMAC (Hutter et al., 2011; Lindauer et al., 2022) builds a surrogate model to guide Bayesian search, irace (López-Ibáñez et al., 2016) removes weak configurations through racing, and AutoFolio (Lindauer et al., 2015) studies the related problem of per-instance algorithm selection. A broad review is given by Schede et al. (2022). In their standard single-context use, these methods optimize the context on which they are run. When the deployment context changes, a standard single-context run often requires new tuning. Transfer-aware AC can instead reuse configurations or numerical observations across related tasks (Franzin & Stützle, 2020). ConfigAgent builds on this general idea of reuse, but also stores textual summaries of observed behavior and limits each summary to the contexts named by its scope. Our experiments use SMAC3 and irace as strong from-scratch baselines. On targets sharing the NSGA-II parameter space, Appendix C.3 additionally compares direct incumbent reuse, Warm-SMAC (Lindauer & Hutter, 2018), RGPE (Feurer et al., 2018), and MI-SMBO-style transfer (Feurer et al., 2015) under matched target budgets.

LLMs for Combinatorial Optimization. Multiple recent works use LLMs to design or improve CO algorithms themselves. FunSearch (Romera-Paredes et al., 2024) and AlphaEvolve (Novikov et al., 2025) pair an LLM with an evolutionary outer loop to evolve programs that solve a target problem. ReEvo (Ye et al., 2024) and LLaMEA (van Stein & Bäck, 2025) extend this to verbal feedback and broader heuristic spaces. CO Bench (Sun et al., 2026) provides a benchmark for LLM agents on CO problems, while DRAGON (Chen et al., 2026) studies LLM-driven decomposition for large CO instances. A complementary line translates natural-language problem descriptions into mathematical formulations (Ramamonjison et al., 2023; Xiao et al., 2024). The shared pattern is that the LLM generates the solving algorithm or the formulation itself — a program, a heuristic, a decomposition strategy, or a mathematical model — so configuration is implicit in the generated artefact rather than declared as a parameter setting on a fixed solver. ConfigAgent instead leaves each solver and its optimization procedure unchanged and configures only its exposed parameters.

LLMs for Hyperparameter Search. Another related stream of work targets parameter tuning under the LLM-in-the-loop framing. OPRO (Yang et al., 2024) treats the LLM as a numeric optimizer

proposing new hyperparameter vectors from history. LLAMBO (Liu et al., 2024) fuses LLM proposals with Bayesian-optimization surrogates, and AgentHPO (Liu et al., 2025) runs a multi-step LLM agent over an ML training pipeline. LB-MCTS (Xu et al., 2026) combines LLM proposals with Monte-Carlo tree search for CASH. Closer to combinatorial solvers, AutoEP (Xu et al., 2025) performs online parameter control within a single metaheuristic run, while irace-evo (Chacón Sartori & Blum, 2025) and LLaMEA-HPO (van Stein et al., 2025) combine code evolution with parameter optimization for generated heuristics. These methods cover HPO/CASH in ML, in-run online control, and joint heuristic generation and tuning. ConfigAgent instead studies offline configuration of fixed CO algorithms and stores numerical evidence and textual summaries in a shared knowledge base, with explicit rules limiting the contexts in which each summary can be reused.

3 The Method

3.1 Problem Definition

We define the configuration context as a four-tuple $c = (f, s, \mathbf{O}, \sigma)$. The first element f denotes a problem family, which consists of an instance distribution $\mathcal{D}_f$ and a set of objectives $\mathbf{O}^f = \{O_1^f, O_2^f, \ldots\}$ with O_1^f being the primary objective. The second element s specifies the instance scale. Scale is family-specific: it is the number of cities for TSP, the number of items for KP and BPP, a jobs–machines pair for FJSP, and the number of customers for CVRP. The third element $\mathbf{O} \subseteq \mathbf{O}^f$ is the subset of objectives considered in this context, with $n_{\text{obj}} = |\mathbf{O}|$ being the number of objectives. The last element σ denotes the target algorithm with configurable parameters $\boldsymbol{\theta}$. Let $Q(\sigma(\mathcal{I}; \boldsymbol{\theta}))$ be the performance measure of configuration $\boldsymbol{\theta}$ on instance $\mathcal{I}$, which is defined as O_1^f when $n_{\text{obj}} = 1$ and as the hypervolume (HV) indicator (Zitzler et al., 2003) over $\mathbf{O}$ when $n_{\text{obj}} \geq 2$. Given context c with configuration space Θ_c, Algorithm Configuration (AC) seeks:

$$\boldsymbol{\theta}_c^* \;=\; \arg\max_{\boldsymbol{\theta} \in \Theta_c} \; \mathbb{E}_{\mathcal{I} \sim \mathcal{D}_f} \big[\, Q\big(\sigma(\mathcal{I};\, \boldsymbol{\theta})\big) \,\big]\,. \tag{1}$$

The optimal $\boldsymbol{\theta}_c^*$ is context-dependent: a configuration that performs well for a TSP instance with 100 nodes may not effectively scale to 1000 nodes; configurations transfer between routing families do not necessarily transfer to packing problems.

Classical AC methods such as SMAC3 (Hutter et al., 2011; Lindauer et al., 2022) and irace (López-Ibáñez et al., 2016) treat the AC problem in Eq. 1 as black-box optimization, and optimize the configuration for a given context c. Specifically, SMAC3 (Lindauer et al., 2022) fits a response surface $\hat{Q} : \Theta_c \to \mathbb{R}$ and irace (López-Ibáñez et al., 2016) maintains a sampling density over Θ_c. These black-box optimization processes can be time-consuming and provide limited insight into why a configuration works. Without an explicit transfer mechanism, the obtained configuration may not generalize beyond the training context, and new tuning is typically required when the context changes (Franzin & Stützle, 2020; Schede et al., 2022). For our setting, the missing piece is an explicit mechanism for combining numerical outcomes with descriptions of the context and short explanations of observed behavior. ConfigAgent uses an LLM during training to read structured descriptions and numerical trial history, then stores short summaries tagged with the contexts in which they may be reused.

3.2 Agentic Algorithm Configuration

ConfigAgent trains on several contexts and stores their results in one knowledge base. Each context is trained independently: for each context c, ConfigAgent performs T trials in which an LLM reads the local evidence, proposes a configuration, and summarizes the result. Each trial t runs the four phases sketched in Figure 1 and codified as Algorithm 1: *perception*, *reasoning*, *execution*, and *reflection*. The first trial in every context evaluates the default configuration to seed the local history; the remaining $T - 1$ trials are agent-driven. Cross-context records are used only after these independent training runs have been committed to the knowledge base.

Perception starts every trial by assembling an evidence packet

$$\mathcal{E}_t = \phi(\mathcal{H}_t^c,\, \mathcal{K},\, c), \tag{2}$$

where ϕ is a deterministic prompt assembly function, $\mathcal{H}_t^c = \{(\boldsymbol{\theta}_i, Q_i)\}_{i=1}^{t-1}$ is the context-local history at the start of trial t collecting the $t - 1$ trials already evaluated, with $\boldsymbol{\theta}_i$ and Q_i the evaluated

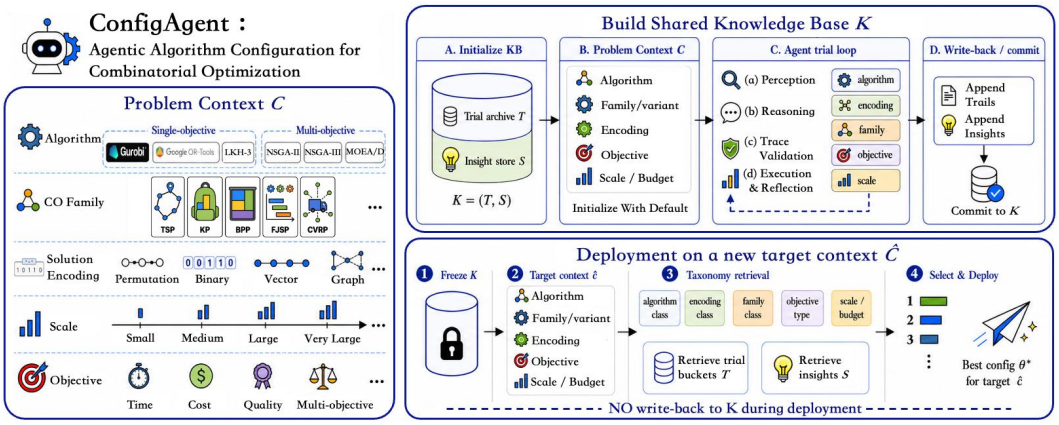

Figure 1: ConfigAgent pipeline.

Algorithm 1 ConfigAgent: training across C contexts then zero-shot deploy.

Require: contexts $c = 1, \ldots, C$, each with target algorithm σ^c and training instances $\mathcal{I}_1^c, \ldots, \mathcal{I}_T^c$; per context trial budget T; deploy retrieval depth K, multi-trial floor $n_{\min}$

$\mathcal{K} = (\mathcal{T}, \mathcal{S}) \leftarrow (\emptyset, \emptyset)$ // shared KB starts empty
// Phase 1: per context training. Each context is trained independently;
*// cross context retrieval is **not** active during training.*
for $c = 1$ to C **do**
  $\mathcal{H}^c \leftarrow \emptyset,\ \mathcal{S}^c \leftarrow \emptyset$ // context local trial history and insight pool
  for $t = 1$ to T **do**
    $\mathcal{E}_t \leftarrow \phi(\mathcal{H}_t^c, \emptyset, c)$ // $\mathcal{H}_t^c$ holds trials $1, \ldots, t-1$; phase 1: empty $\mathcal{K}$ slot
    $(\boldsymbol{\theta}_t, h_t, \mathbf{r}_t) \leftarrow \mathrm{LLM}_{\mathrm{propose}}(\mathcal{E}_t)$ *// propose with hypothesis and cited trace*
    assert every cited id in $\mathbf{r}_t$ exists in $\mathcal{E}_t$ *// validator*
    $Q_t \leftarrow Q\big(\sigma^c(\mathcal{I}_t^c; \boldsymbol{\theta}_t)\big)$ *// solver execution*
    $(r_t^{\mathrm{short}}, r_t^{\mathrm{long}}) \leftarrow \mathrm{LLM}_{\mathrm{reflect}}(\boldsymbol{\theta}_t, Q_t, h_t, \mathcal{E}_t)$
    append $(\boldsymbol{\theta}_t, Q_t, r_t^{\mathrm{short}})$ to $\mathcal{H}^c$
    if $r_t^{\mathrm{long}} \neq \emptyset$ **then** $\mathcal{S}^c \leftarrow \mathcal{S}^c \cup \{\mathrm{canonicalize}(r_t^{\mathrm{long}}, \mathbf{v}(c))\}$ // dedup-merge insight
  end for
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{H}^c,\ \mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}^c$ // commit both trials and insights to shared KB
end for
// Phase 2: deployment on a target context $\hat{c}$.
deploy$(\hat{c})$: retrieve $(\mathcal{T}_{\hat{c}}, \mathcal{S}_{\hat{c}})$ from $\mathcal{K}$ via Eq. 6, return $\arg\max$ over the top-K band filtered candidates with $n_{\boldsymbol{\theta}} \geq n_{\min}$

configuration and the corresponding performance at trial i, and $\mathcal{K}$ is the shared knowledge base (will be introduced in §3.3). The function ϕ packs $\mathcal{E}_t$ from four channels. *Local history* groups $\mathcal{H}_t^c$ by configuration and ranks each group by mean quality, then by sample count, then by inverse standard deviation, so that well-tested high-quality configurations sit on top. *Coverage map* bins each tunable parameter and surfaces unsampled regions. *Parameter direction signals* are per-tunable Spearman correlations $\rho(\theta_p, Q)$ between each parameter and the achieved quality, computed on $\mathcal{H}_t^c$ and on any retrieved cross-context records, giving a calibration-free monotonicity hint. *Retrieved evidence* brings in records from $\mathcal{K}$ that share taxonomy with the current context, plus a read-only static prior block listing each algorithm's recommended parameter ranges and known failure modes from public references. The static prior block is read in both training and deployment, while retrieval against the experience records inside $\mathcal{K}$ is gated off during training and active only at deployment.

Reasoning consumes $\mathcal{E}_t$ in a single LLM call that returns the proposed configuration $\boldsymbol{\theta}_t \in \Theta_c$, a one-sentence hypothesis h_t predicting whether the trial will improve over the best so far, and a structured trace $\mathbf{r}_t$ listing the evidence ids the LLM relied on, drawn from the trial ids, insight ids, source-context tags, and coverage gap names rendered in $\mathcal{E}_t$:

$$(\boldsymbol{\theta}_t, h_t, \mathbf{r}_t) = \mathrm{LLM}_{\mathrm{propose}}(\mathcal{E}_t). \tag{3}$$

The same call also emits an analysis and a strategy field for human audit. A deterministic validator following the CRITIC pattern (Gou et al., 2024) rejects any id in $\mathbf{r}_t$ that was not rendered in the prompt, making every cited statement traceable to prompt evidence.

Before execution, the implementation checks the proposed configuration against the active solver schema. Numeric values are clipped to the declared range and rounded to four decimal places, missing or invalid categorical values are rejected, and fixed budget values replace any value proposed by the LLM.

Execution then runs the target algorithm σ^c of the current context on the training instance $\mathcal{I}_t^c$ scheduled for trial t under the proposed configuration $\boldsymbol{\theta}_t$, and records the performance measure $Q_t = Q\big(\sigma^c(\mathcal{I}_t^c;\, \boldsymbol{\theta}_t)\big)$, where Q is the per-instance quality defined in §3.1. This realized value is the only solver-performance signal given to the LLM; branch counts, cut counts, node expansions, and other solver-internal data are not exposed. If the target algorithm fails or exceeds its wall-clock time cap, Q_t is set to a sentinel that is excluded from later aggregation, but the trial still counts against the per-context budget T.

Reflection closes the trial by calling the LLM a second time on the realized Q_t together with $\mathcal{E}_t$ and emitting a short per-trial note r_t^{short}, plus an optional one-sentence long-term insight r_t^{long}:

$$(r_t^{\text{short}},\, r_t^{\text{long}}) = \text{LLM}_{\text{reflect}}(\boldsymbol{\theta}_t,\, Q_t,\, h_t,\, \mathcal{E}_t). \tag{4}$$

The short note grades h_t against Q_t and is appended to $\mathcal{H}^c$. The long insight is written to the insight list $\mathcal{S}$ inside the knowledge base only when the trial reveals a pattern not already covered by an existing claim. We freeze the knowledge base during the reported deployment experiments to prevent target-test information and target order from changing the comparison.

Compared with the conventional from-scratch AC setup, the design adds three elements: a trace $\mathbf{r}_t$ that records which evidence the LLM used, a shared knowledge base $\mathcal{K}$ for deployment across contexts, and an evidence packet $\mathcal{E}_t$ that includes context descriptions alongside numerical $(\boldsymbol{\theta}, Q)$ records.

3.3 Knowledge Base

The knowledge base (KB) holds two complementary kinds of evidence. *Trials* are raw measurements: each one records that a particular configuration achieved a particular quality on a particular context. *Insights* are LLM-generated summaries: each one is a one-sentence claim about which configurations work in which contexts, accompanied by a scope tag that limits where the claim can be reused. An insight is a retrieval aid, not a measured result. It cannot create a quality value or a trial bucket, and the zero-shot picker cannot select a configuration from insight text alone. Formally, we define the KB as $\mathcal{K} = (\mathcal{T}, \mathcal{S})$. The trial list $\mathcal{T}$ stores tuples $T = (c_T, \boldsymbol{\theta}_T, Q_T)$ of context, configuration, and measured quality. The insight list $\mathcal{S}$ stores tuples $I = (\text{text}_I, \mathcal{A}_I, n_I, c_I)$ of claim text, scope dictionary $\mathcal{A}_I$ naming the contexts the claim applies to, support count n_I that increments each time the same canonicalized claim is re-emitted, and the birth context c_I that first produced it.

Each context c projects to a five dimensional taxonomy vector

$$\mathbf{v}(c) = (s_c, e_c, f_c, o_c, k_c), \tag{5}$$

naming the algorithm class s_c, solution encoding e_c, family class f_c, active objective tuple $o_c = \mathbf{O}_c$, and scale bucket k_c of c. Solution encoding means the representation manipulated by the solver, such as a permutation for TSP, a binary string for Knapsack, or a giant tour with depot separators for CVRP. We split the dimensions into a structural subset $D_{\text{struct}} = \{s, e, f\}$, which determines the type of solver decision being made, and a soft subset $D_{\text{soft}} = \{o, k\}$, which selects the active objectives and the instance scale within a fixed solver type. The objective dimension carries an ordered tuple of named objectives, so two contexts with the same prefix length but different active tuples occupy distinct entries. A scope dictionary $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_e, \mathcal{A}_f, \mathcal{A}_o, \mathcal{A}_k)$ assigns each dimension d an allowed value set $\mathcal{A}_d$, and a context c satisfies the scope, written $\mathbf{v}(c) \models \mathcal{A}$, when $\mathbf{v}_d(c) \in \mathcal{A}_d$ for every $d \in \{s, e, f, o, k\}$.

During training, each context c commits its local history $\mathcal{H}^c$ and its reflection insights into $\mathcal{K}$ only after its T trials end, so the propose call inside the training loop receives an empty $\mathcal{K}$ slot and cross-context retrieval is not active there. Cross-context reads occur only at deployment, where the frozen $\mathcal{K}$ is queried through the retrieval primitive defined next.

3.4 DEPLOYMENT

During deployment, the KB is *frozen*. Zero-shot deployment makes no target-side LLM call: it retrieves eligible measured records and ranks them with Eqs. 6–8. Few-shot deployment uses the same retrieved records and insights in a small target-side LLM-guided adaptation. Both modes use the three retrieval paths detailed in Appendix A.3.1. We define a structural overlap count between two contexts c and c' as $M_{\text{struct}}(c, c') = \big|\{d \in D_{\text{struct}} : \mathbf{v}_d(c) = \mathbf{v}_d(c')\}\big|$, and the relevant slice of $\mathcal{K}$ for a target $\hat{c}$ is retrieved as

$$\mathcal{T}_{\hat{c}} = \{T \in \mathcal{T} : M_{\text{struct}}(c_T, \hat{c}) \geq 2\}, \qquad \mathcal{S}_{\hat{c}} = \{I \in \mathcal{S} : \mathbf{v}(\hat{c}) \models \mathcal{A}_I\}. \tag{6}$$

The threshold of 2 in $\mathcal{T}_{\hat{c}}$ admits trials whose source context matches the target on at least two of the three structural dimensions, namely algorithm class s, solution encoding e, and family class f, and the scope relation in $\mathcal{S}_{\hat{c}}$ admits insights whose declared scope contains the target. The *trial* and *insight* paths share Eq. 6, while the *prior* path pulls algorithm- and family-specific tuning ranges and pitfalls from the read-only static prior block authored from public references. In zero-shot mode, only the retrieved trial records enter the final ranking. In few-shot mode, the LLM can also read the scoped insights and prior when proposing target-side trials.

The retrieved trials in $\mathcal{T}_{\hat{c}}$ are bucketed by configuration, $B_{\boldsymbol{\theta}} = \{T \in \mathcal{T}_{\hat{c}} : \boldsymbol{\theta}_T = \boldsymbol{\theta}\}$, and each bucket carries the support count, mean, and sample variance of its measured quality:

$$n_{\boldsymbol{\theta}} = |B_{\boldsymbol{\theta}}|, \qquad \bar{Q}_{\boldsymbol{\theta}} = \frac{1}{n_{\boldsymbol{\theta}}} \sum_{T \in B_{\boldsymbol{\theta}}} Q_T, \qquad s^2_{\boldsymbol{\theta}} = \frac{1}{\max(n_{\boldsymbol{\theta}} - 1, 1)} \sum_{T \in B_{\boldsymbol{\theta}}} (Q_T - \bar{Q}_{\boldsymbol{\theta}})^2. \tag{7}$$

A literature band filter $\text{Band}(X) = \big\{\boldsymbol{\theta} \in X : \forall p,\, \theta_p \in [\ell_p, u_p]\big\}$ keeps configurations whose every component lies inside the literature range recorded per algorithm in the static prior block, and the deploy picks the band-filtered bucket that maximizes

$$\boldsymbol{\theta}^{\text{deploy}}_{\hat{c}} = \arg \max_{\boldsymbol{\theta} \in \text{Band}(\mathcal{T}_{\hat{c}}),\, n_{\boldsymbol{\theta}} \geq n_{\min}} \big(\bar{Q}_{\boldsymbol{\theta}}, n_{\boldsymbol{\theta}}, -s^2_{\boldsymbol{\theta}}\big), \tag{8}$$

ordered first by mean, then by support count, and finally by smaller variance, with the requirement $n_{\boldsymbol{\theta}} \geq n_{\min}$ so the deployed configuration carries at least $n_{\min}$ supporting trials and single-trial lucky peaks are filtered out. Eq. 8 supports two deployment modes. *Zero-shot* picks $\boldsymbol{\theta}^{\text{deploy}}_{\hat{c}}$ directly from the frozen KB. *Few-shot* uses $\mathcal{T}_{\hat{c}}$ and $\mathcal{S}_{\hat{c}}$ as a warm start, runs $\text{LLM}_{\text{propose}}$ for a small target-side trial budget, then applies the same priority comparison over the augmented bucket statistics.

The mean is the ranking statistic and the variance is only the final tiebreaker; the method does not assume that the quality values are Gaussian or unimodal. Records from different stored configurations remain in separate buckets.

4 RESULTS

4.1 SETUP

We train ConfigAgent on five CO families, namely TSP, KP, BPP, FJSP, and CVRP, at one primary scale per family, with objective counts $n_{\text{obj}} \in \{1, 2, 3, 5\}$. Each family additionally has two non-primary scales held out for cross-scale generalization tests. We consider six algorithms across three regimes: one exact mixed integer programming solver (Gurobi), two hybrid or heuristic search engines (OR-Tools and LKH-3), and three multi-objective evolutionary algorithms (MOEAs), namely NSGA-II, NSGA-III, and MOEA/D. At each problem's primary scale we evaluate 12 single-objective contexts (Gurobi/OR-Tools on every problem, plus LKH-3 on TSP and CVRP only) and 45 multi-objective contexts with $n_{\text{obj}} \in \{2, 3, 5\}$ (5 families $\times$ 3 objective counts $\times$ 3 MOEAs), yielding 57 base contexts. These 57 contexts define the overall evaluation set; the transfer experiments in Appendix C use a separate frozen source KB built only from the 15 two-objective primary contexts (5 families $\times$ 3 MOEAs). Additionally, we evaluate out-of-distribution generalization on one structurally different variant of each family: asymmetric TSP, multidimensional KP, mixed-size BPP, heterogeneous FJSP, and clustered CVRP. For every context we generate 100 training and 100 test instances, and evaluate the selected configuration with 10 random seeds on each test instance. The default LLM backbone is Qwen2.5-32B-Instruct (Qwen Team, 2024) served via vLLM (Kwon et al., 2023). For the three

single-objective solvers, we set a per-instance time limit of 60 s for Gurobi and OR-Tools, and a size-aware budget of 120 s for LKH-3 on standard synthetic scales and 300 s on variant scales and TSPLIB hard instances. The three MOEAs run with a fixed fitness budget of 2,500 iterations per instance. Full problem, solver, parameter, and ConfigAgent hyperparameter settings are in Appendix A.8 and A.2.

We compare ConfigAgent against three main baselines. *Default* runs each solver at its default parameters with no tuning trial. *SMAC3* (Lindauer et al., 2022) runs a random forest surrogate with log expected improvement at 200 trials per context. *irace* (López-Ibáñez et al., 2016) runs iterated F-racing at 800 trials per context (i.e., the `maxExperiments` parameter). The trial counts of SMAC3 and irace are chosen to balance configuration quality against run time, with per-context convergence records in App. B.2. App. C.3 adds direct reuse and three transfer-aware numerical controls on the shared NSGA-II space, while App. B.1 gives a band-matched SMAC3 control for the numerical part of the static prior. The configuration quality metric is the average objective value for $n_{\text{obj}}=1$ and average hypervolume (HV) for $n_{\text{obj}}\geq 2$. App. A.11 reports one-sided Wilcoxon signed-rank tests (Wilcoxon, 1945) with Holm correction (Holm, 1979). All experiments run on one 208-core Intel Xeon Platinum 8470Q CPU and one NVIDIA RTX PRO 6000 GPU. Our source code will be publicly available.

4.2 Main configuration results

In this subsection, we report the in-distribution configuration performance of our method. Table 1 reports single-objective results at each family's primary scale, while Table 2 reports the multi-objective counterpart across the same five families. The means show that ConfigAgent is generally competitive with SMAC3 and irace and often attains the best reported value. App. A.11 reports practical relative margins and paired tests on the primary two-objective contexts. The configuration-time columns indicate that ConfigAgent reaches this quality with fewer configuration trials and lower reported time in most contexts. Section 4.3 evaluates how the resulting KB can be reused on new contexts without repeating a full target-side search.

Table 1: Single-objective results at each family's primary scale.

		Objective value				Configuring time (s) ↓		
Family (objective)	**Solver**	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**SMAC3**	**irace**	**ConfigAgent**
TSP n500 (tour ↓)	LKH-3	16,452.7	16,479.4	16,461.3	**16,451.6**	38,046	9,000	**8,240**
	OR-Tools	17,503	16,899.3	17,050.2	**16,894.7**	2,012	5,800	**583**
	Gurobi	18,497.9	18,419.3	18,452.1	**18,408.4**	7,519	7,000	**5,180**
KP n500 (value ↑)	OR-Tools	20,305.6	20,322.4	20,318.7	**20,327.4**	263	138	**96**
	Gurobi	20,269.4	20,325.0	20,269.4	**20,327.4**	377	450	**218**
BPP n500 (# bins ↓)	OR-Tools	225.83	224.42	224.97	**223.95**	1,500	950	**762**
	Gurobi	252.40	232.65	234.18	**232.42**	19,481	**5,500**	5,630
FJSP j10m5 (makespan ↓)	OR-Tools	186.42	185.13	185.87	**184.78**	312	178	**148**
	Gurobi	99.02	96.02	95.95	**94.12**	16,150	6,435	**5,470**
CVRP n200 (distance ↓)	LKH-3	3,194.6	3,185.2	3,191.7	**3,179.4**	28,350	9,292	**6,920**
	OR-Tools	3,538.0	3,487.8	3,501.4	**3,485.3**	2,138	4,800	**1,924**
	Gurobi	3,822.3	3,810.5	3,764.2	**3,757.7**	20,020	6,000	**5,340**

4.3 Generalization Performance

In this subsection, we report the out-of-distribution generalization performance of ConfigAgent on structural variants and public benchmarks. ConfigAgent runs in *zero-shot deploy* mode on every target context: the KB is frozen, and on each target the system retrieves taxonomy-aligned source buckets, ranks them deterministically, and evaluates the selected configuration with no target-side propose-solve-reflect trials. The from-scratch SMAC3 and irace baselines evaluated here do not use source-context records, so they retune each target at the same per-target budgets used for in-distribution training, namely $T=200$ and $B=800$. The ConfigAgent times in Tables 3-4 reflect retrieval plus evaluation of the selected configuration and vary with the solver and test workload. Directly recorded zero-shot, adaptation, and final-test stages are reported separately in Table 33.

Table 2: Multi-objective results at each family's primary scale.

		HV ↑				Configuring time (s) ↓		
Family (HV scale)	**Algorithm**	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**SMAC3**	**irace**	**ConfigAgent**
TSP n500 ($\cdot 10^8$)	NSGA-II	2.255	2.336	2.334	**2.339**	8,000	5,600	**2,917**
	NSGA-III	2.200	2.221	2.273	**2.277**	7,600	5,440	**2,897**
	MOEA/D	2.240	2.473	2.580	**2.583**	8,800	5,680	**2,966**
KP n500 ($\cdot 10^9$)	NSGA-II	1.876	1.878	1.880	**1.883**	6,700	5,200	**2,878**
	NSGA-III	1.862	1.865	1.865	**1.866**	6,900	5,440	**2,863**
	MOEA/D	1.935	2.001	2.005	**2.008**	8,300	5,920	**2,917**
BPP n500 ($\cdot 10^3$)	NSGA-II	4.731	4.729	4.730	**4.732**	10,000	6,640	**3,305**
	NSGA-III	4.731	4.733	4.732	**4.734**	9,900	6,725	**3,305**
	MOEA/D	4.728	4.726	4.726	**4.729**	10,200	7,520	**3,369**
FJSP j10m5 ($\cdot 10^6$)	NSGA-II	2.039	2.042	2.041	**2.044**	13,700	16,640	**3,664**
	NSGA-III	2.031	2.038	2.037	**2.041**	13,600	16,785	**3,679**
	MOEA/D	2.014	2.022	2.018	**2.024**	13,500	18,800	**3,719**
CVRP n200 ($\cdot 10^4$)	NSGA-II	4.123	4.133	4.155	**4.158**	7,600	5,840	**2,864**
	NSGA-III	4.167	4.219	4.227	**4.235**	7,700	5,600	**2,848**
	MOEA/D	4.322	4.602	4.606	**4.620**	8,500	6,800	**2,894**

Structural variants. We evaluate the generalization performance on one synthetic structural variant per family under its dedicated solver. ATSP shifts the family subtype on TSP via an asymmetric distance matrix, MDK shifts the constraint structure on KP from one to several capacity rows, Mixed-size BPP changes the item-size distribution, Heterogeneous FJSP uses non-uniform machine speeds, and Clustered CVRP samples customers from Gaussian clusters. Table 3 shows that ConfigAgent is competitive with SMAC3 and irace in these settings, while the frozen KB reduces target-side configuration time.

Table 3: Variant transfer per family, with one variant per row at the family's primary scale.

				Objective value				Configuring time (s) ↓		
Variant (Family)	**Scale**	**Solver**	**Metric**	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**SMAC3**	**irace**	**ConfigAgent**
ATSP (TSP)	n-mixed	LKH-3	tour (↓)	1,721.4	1,719.8	1,720.7	**1,719.2**	4,173	6,283	**124**
MDK (KP)	n-mixed	OR-Tools	value (↑)	91.39	91.46	91.42	**91.49**	263	138	**87**
Mixed-size BPP	n500	Gurobi	bins (↓)	276.0	248.8	248.8	**248.6**	17,283	5,614	**264**
Heterogeneous FJSP	j10m5	Gurobi	makespan (↓)	210.4	209.7	209.9	**209.6**	14,820	5,972	**168**
Clustered CVRP	n200	OR-Tools	dist (↓)	3,202.0	3,199.2	3,201.3	**3,198.7**	3,245	4,683	**142**

Public benchmark libraries. Table 4 reports generalization on five canonical libraries. ConfigAgent's configuring time is the per-instance redeploy cost from the frozen KB while SMAC3 and irace retune from scratch on every instance. ConfigAgent is competitive across the five libraries, with lower configuring time in the displayed summaries.

Table 4: Public benchmark transfer.

			Objective value				Configuring time (s) ↓		
Library	**Solver**	**# insts**	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**SMAC3**	**irace**	**ConfigAgent**
TSPLIB95	LKH-3	10	204,359	204,319	204,341	**204,288**	33,285	12,650	**310**
Chu–Beasley MKP	Gurobi	241	128,906	129,568	128,906	**129,721**	8,656	5,912	**80**
Beasley BPP	Gurobi	60	141.40	138.95	140.27	**137.58**	11,740	4,461	**150**
Brandimarte FJSP	Gurobi	8	173.38	171.75	172.25	**170.25**	11,771	4,473	**62**
Solomon VRPTW	OR-Tools	50	767.75	753.92	761.55	**751.18**	10,473	3,980	**75**

4.4 Efficiency Analysis

A notable advantage of our method is its efficiency in the reported training and transfer settings. Figure 2 shows a representative best-so-far trajectory, and Table 5 summarizes training and deployment times. Appendix B provides event counts, stage-level records, lifecycle and break-even costs, hardware boundaries, and the solver-time threshold at which LLM latency is amortized; few-shot adaptation uses $T{=}10$ target trials.

Figure 2: Convergence trend on TSP n500, NSGA-II, n_{obj}=2.

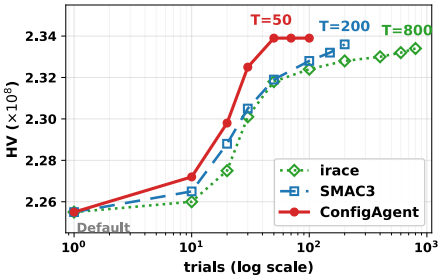

Table 5: Per-context configuring time (s). CA means ConfigAgent.

Setting	SMAC3	irace	CA	Speedup†
Train mode (one-time, T=100 for CA)				
Source-context tune	9,200	8,000	3,800	2.1×
Deploy mode (per target, CA reads frozen KB)				
Same context, zero-shot	9,200	8,000	100	80×
Same context, few-shot T=10	9,200	8,000	250	32×
Obj. transfer n_{obj}: 2→3	10,200	13,800	80	128×
Obj. transfer n_{obj}: 2→5	11,200	15,200	80	140×
Scale transfer BPP n: 500→1000	26,900	21,500	95	226×
Public libraries (5, macro mean)	15,200	6,300	142	44×

† Speedup = min(SMAC3, irace) / ConfigAgent on the same row.

Table 6: Component ablation and LLM only baselines. Values are HV (↑) on the held-out test sets.

Variant	TSP n500 ($\cdot 10^8$)	KP n500 ($\cdot 10^9$)	BPP n500 ($\cdot 10^3$)	FJSP j10m5 ($\cdot 10^6$)	CVRP n200 ($\cdot 10^4$)
Full ConfigAgent	**2.339**	**1.883**	**4.732**	**2.044**	**4.158**
No reflection	2.335	1.881	**4.732**	2.042	4.054
No trace validator	2.308	1.869	4.725	2.042	4.066
No retrieval at deployment	2.318	1.880	4.723	2.042	4.054
OPRO	2.254	1.868	4.722	2.038	4.048
LLAMBO	2.279	1.865	4.718	2.018	4.040

ConfigAgent uses GPU inference in training and few-shot adaptation, whereas SMAC3 and irace are CPU tuners; zero-shot deployment makes no LLM call. The reported lifecycle costs amortize after 3–8 related targets for a 12-context KB and 11–36 targets for a 57-context KB.

4.5 Ablation Study

Table 6 removes one component at a time and compares with OPRO (Yang et al., 2024) and LLAMBO (Liu et al., 2024) at the same training budget. Full ConfigAgent is best in all five columns; removing reflection, trace validation, or deployment retrieval lowers HV in most settings. Appendix D.1 gives component details, while Appendix C.3 separately compares transfer-aware numerical methods under a shared parameterization.

4.6 Interpretability

ConfigAgent records rationales and typed traces whose citations resolve to prompt evidence. Table 7 summarizes exploration, insight, and transfer records. Appendix D.3 further tests the safeguards around stored insight text and population aggregation.

Table 7: Three layers of interpretability with concrete examples from the released KB.

Layer	Concrete example
Exploration	ATSP NSGA-II training trace: seed at the published default at trial 1 (HV $1.87{\cdot}10^7$), **lower cxpb and raise mutpb** to probe an unexplored region at trial 2 ($3.00{\cdot}10^7$), **shift toward high cxpb and low mutpb** at trial 5 ($3.29{\cdot}10^7$), reach best-so-far (cxpb=0.90, mutpb=0.001) at trial 8 ($\mathbf{3.33{\cdot}10^7}$), and falsify the trend by probing high mutpb at trial 10 ($2.70{\cdot}10^7$).
Insights	Mechanism-level claims accumulated by reflection, including **high cxpb with low mutpb** preserves long good subsequences on permutation encodings, **bit-flip mutpb** $\in [0.005, 0.03]$ on Knapsack, the **first-fit-decreasing plateau** on BPP, and **Tchebycheff dominates weighted sum** on concave MOEA/D fronts.
Transfer	Scoped insights can inform few-shot proposals on compatible targets; frozen zero-shot selection uses the corresponding measured trial buckets. Same retrieval primitive carries claims across contexts: TSP permutation insight to **ATSP, Mixed-scale, Clustered TSP** (family-subtype shift); Knapsack bit-flip band to **MDK** (added capacity constraints); Tchebycheff dominance to $n_{\text{obj}} \in \{3, 5\}$ since the scope whitelist keeps it unrestricted.

5 Conclusions and Future Work

ConfigAgent reuses measured trials and scoped insights across CO problems, solvers, scales, and objectives. Source contexts are trained independently; at deployment, the frozen knowledge base supports deterministic zero-shot ranking of compatible trial buckets or a few LLM-guided target trials. Across evaluated contexts, ConfigAgent achieves quality comparable to strong AC baselines, with occasional improvements, while substantially reducing target-side configuration time. The quality–cost comparison includes source construction and GPU inference. One representation covers exact, heuristic, and multi-objective solvers without changing their implementations, while typed traces expose proposal evidence. Future work includes conditional parameter spaces and online knowledge-base updates beyond the frozen protocol evaluated here.

ACKNOWLEDGMENTS

Acknowledgments will be added in the camera ready version.

AI USE STATEMENT

During preparation of this manuscript, Codex (OpenAI) was used to assist with language editing and LaTeX formatting; literature discovery and bibliography checks; discussions of research ideas, experimental design, and result interpretation; and drafting and revising portions of the manuscript. It also assisted with experiment organization and consistency checks across reported results. The authors reviewed and verified the AI-assisted content and take full responsibility for the methods, experiments, numerical results, claims, citations, and conclusions. LLMs used as components of the reported method are described separately in the main text and appendix.

REPRODUCIBILITY STATEMENT

The main text and appendix specify the benchmark families, train–test split, configuration spaces, trial budgets, evaluation seeds, solver limits, and statistical tests used in the experiments. The anonymous supplementary material will include the code and machine-readable configurations needed to reproduce the reported evaluations.

REFERENCES

Ali AhmadiTeshnizi, Wenzhi Gao, Herman Brunborg, Shayan Talaei, Connor Lawless, and Madeleine Udell. OptiMUS-0.3: Using large language models to model and solve optimization problems at scale. *arXiv preprint arXiv:2407.19633*, 2024. URL https://arxiv.org/abs/2407.19633.

John E Beasley. OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990. doi: 10.1057/jors.1990.166.

Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, 1993. doi: 10.1007/BF02023073.

Camilo Chacón Sartori and Christian Blum. irace-evo: Automatic algorithm configuration extended with LLM-based code evolution. *arXiv preprint arXiv:2511.14794*, 2025. URL https://arxiv.org/abs/2511.14794.

Shengkai Chen, Zhiguang Cao, Jianan Zhou, Yaoxin Wu, Senthilnath Jayavelu, Zhuoyi Lin, Xiaoli Li, and Shili Xiang. DRAGON: LLM-driven decomposition and reconstruction agents for large-scale combinatorial optimization. *arXiv preprint arXiv:2601.06502*, 2026. URL https://arxiv.org/abs/2601.06502.

Paul C Chu and John E Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4(1):63–86, 1998. doi: 10.1023/A:1009642405419.

Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014. doi: 10.1109/TEVC.2013.2281535.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pp. 1128–1135, 2015. doi: 10.1609/aaai.v29i1.9354.

Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. Practical transfer learning for Bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2018. URL https://arxiv.org/abs/1802.02219.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012. URL https://jmlr.org/papers/v13/fortin12a.html.

Alberto Franzin and Thomas Stützle. Towards transferring algorithm configurations across problems. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*, 2020. URL https://openreview.net/forum?id=bqswrMr-Ed.

David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989. ISBN 978-0-201-15767-3.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2024. URL https://www.gurobi.com.

Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017. URL https://forskning.ruc.dk/en/publications/an-extension-of-the-lin-kernighan-helsgaun-tsp-solver-for-constra/.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. URL https://www.jstor.org/stable/4615733.

Qinglong Hu and Qingfu Zhang. Partition to evolve: Niching-enhanced evolution with LLMs for automated algorithm discovery. In *Advances in Neural Information Processing Systems*, 2025. doi: 10.52202/085713-5177.

Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009. doi: 10.1613/jair.2861.

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

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. In *Advances in Neural Information Processing Systems*, 2025a. doi: 10.52202/085713-5495. URL https://openreview.net/forum?id=qr5uMEs6iR. arXiv:2509.16865.

Xia Jiang, Yaoxin Wu, Chenhao Zhang, and Yingqian Zhang. DRoC: Elevating large language models for complex vehicle routing via decomposed retrieval of constraints. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=s9zoyICZ4k.

Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 6 edition, 2018. doi: 10.1007/978-3-662-56039-6.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Marius Lindauer and Frank Hutter. Warmstarting of model-based algorithm configuration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 1355–1362, 2018. doi: 10.1609/aaai.v32i1.11532.

Marius Lindauer, Holger H Hoos, Frank Hutter, and Torsten Schaub. AutoFolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778, 2015. doi: 10.1613/jair.4726.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022. URL https://jmlr.org/papers/v23/21-0888.html.

Siyi Liu, Chen Gao, and Yong Li. AgentHPO: Large language model agent for hyper-parameter optimization. In *Conference on Parsimony and Learning (CPAL)*, volume 280 of *Proceedings of Machine Learning Research*, pp. 1146–1169. PMLR, 2025. URL https://proceedings.mlr.press/v280/liu25c.html. arXiv:2402.01881.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2402.03921. arXiv:2402.03921.

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

Heinz Mühlenbein. How genetic algorithms really work: I. mutation and hillclimbing. In Reinhard Männer and Bernard Manderick (eds.), *Parallel Problem Solving from Nature (PPSN II)*, pp. 15–26, Amsterdam, 1992. Elsevier. URL https://dblp.org/rec/conf/ppsn/Muhlenbein92.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J R Ruiz, Abbas Mehrabian, M Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Laurent Perron and Vincent Furnon. OR-Tools. Google. https://developers.google.com/optimization, 2025. Version v9.12, released 2025-02-17.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. NL4Opt competition: Formulating optimization problems based on their natural language descriptions. In *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pp. 189–203. PMLR, 2023. URL https://proceedings.mlr.press/v220/ramamonjison23a.html.

Gerhard Reinelt. TSPLIB—a traveling salesman problem library. *ORSA Journal on Computing*, 3(4): 376–384, 1991. doi: 10.1287/ijoc.3.4.376.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco J R Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

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

Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987. doi: 10.1287/opre.35.2.254.

Weiwei Sun, Shengyu Feng, Shanda Li, and Yiming Yang. CO-Bench: Benchmarking language model agents in algorithm search for combinatorial optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33126–33134, 2026. doi: 10.1609/aaai.v40i39.40596.

Niki van Stein and Thomas Bäck. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29(2): 331–345, 2025. doi: 10.1109/TEVC.2024.3497793.

Niki van Stein, Diederick Vermetten, and Thomas Bäck. In-the-loop hyper-parameter optimization for LLM-based automated design of heuristics. *ACM Transactions on Evolutionary Learning and Optimization*, 2025. doi: 10.1145/3731567. arXiv:2410.16309.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. doi: 10.2307/3001968.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=HobyL1B9CZ.

Beicheng Xu, Weitong Qian, Lingching Tung, Yupeng Lu, and Bin Cui. Tree-structured synergy of large language models and Bayesian optimization for efficient CASH. *arXiv preprint arXiv:2601.12355*, 2026. URL https://arxiv.org/abs/2601.12355.

Zhenxing Xu, Yizhe Zhang, Weidong Bao, Hao Wang, Ming Chen, Haoran Ye, Wenzheng Jiang, Hui Yan, and Ji Wang. AutoEP: LLMs-driven automation of hyperparameter evolution for metaheuristic algorithms. *arXiv preprint arXiv:2509.23189*, 2025. URL https://arxiv.org/abs/2509.23189.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

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

Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. doi: 10.1109/TEVC.2007.892759.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999. doi: 10.1109/4235.797969.

Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003. doi: 10.1109/TEVC.2003.810758.

APPENDICES

CONTENTS

Part A Experimental setup
A.1 Training and test protocol 14
A.2 Agent hyperparameters 15
A.3 Evidence packet assembly ϕ 15
A.4 Prompt flow and trace validation 16
A.5 Insight lifecycle 16
A.6 Solver invocation 17
A.7 Baseline configuration 17
A.8 Benchmark families, variants, and solver portfolio 17
A.9 AC agent prompt templates 22
A.10 A concrete filled-in propose prompt 24
A.11 Statistical significance on the main body contexts 26
Part B Efficiency
B.1 Deploy picker sensitivity 27
B.2 Per-trial timing breakdown 28
B.3 ConfigAgent best-so-far by backbone 29
B.4 Backbone sensitivity 30
Part C Generalization
C.1 Objective transfer, extensions, and cross-objective deployment 31
C.2 Scale transfer, cross-scale deployment, and leave-one-family-out 32
C.3 Variant transfer, structural variants, and real-world benchmarks 33
C.4 Three-way transfer via KB reasoning 38
Part D Interpretability
D.1 Ablations, components and plain LLM baseline 41
D.2 Agent behavior and knowledge base statistics 41
D.3 Insight safeguards and aggregation checks 41
D.4 Sample accumulated insights 42
Part E Static external prior entries
E.1 Solver mechanism descriptions 44
E.2 Problem family, variant, and benchmark library specs 45
Part F Limitations 47

A PART A: EXPERIMENTAL SETUP

Part A documents the experimental setup shared across the paper. App. A.1 gives the training and test protocol. App. A.2 lists the agent hyperparameters. Apps. A.3–A.5 specify the evidence packet, retrieval pipeline, trace validator, prompt flow, and insight lifecycle. App. A.6 defines the shared solver invocation routine and App. A.7 fixes the SMAC3 and irace baseline configuration. App. A.8 fixes the benchmark families, structural variants, solver portfolio, and instance generation. App. A.9 documents the prompt schema, App. A.10 walks through one concrete audited example, and the full prompts are released alongside the code. App. A.11 reports the per-context paired Wilcoxon table that backs the main body significance claims.

A.1 TRAINING AND TEST PROTOCOL

For each combination of a problem family and a scale level we synthesize 100 training and 100 held out test instances using the templates in App. A.8, giving 15 instance pools and 3,000 instances in total. For TSP, BPP, FJSP, and CVRP one pool per family and scale level covers all four objective counts at once (single objective and 2, 3, 5 objectives), since the rows of Tab. 14 are all read from a single solution and need no extra fields per instance. Knapsack instances additionally carry one value vector per active objective drawn at generation time, with item weights and capacity shared across objective counts.

Each base context draws its training trials from the matching pool and reports quality on the matching held out test set. ConfigAgent runs $T{=}100$ training trials per context and writes its experience into $\mathcal{K}$, while SMAC3 and irace consume the same pool at their own outer loop budgets of $T{=}200$ and $T{=}800$ trials respectively. Final numbers are computed on the 100 held out test instances of the same primary scale, averaged over 10 random seeds per test instance.

A.2 Agent hyperparameters

ConfigAgent exposes a small set of system hyperparameters beyond the LLM backbone. Every context reported in the paper uses the values in Table 8, which we fix once and never tune across the contexts in this paper.

Table 8: ConfigAgent system hyperparameters.

Hyperparameter	**Value**	**Role**
Training trials per context T	100	training loop length
Propose temperature	0.4	$\mathrm{LLM}_{\mathrm{propose}}$ decoding
Reflect temperature	0.2	$\mathrm{LLM}_{\mathrm{reflect}}$ decoding
Propose max tokens	1000	propose generation cap
Reflect max tokens	250	reflect generation cap
Coverage map bins per numeric param	5	exploration gap channel
RAG retrieval top k, trial buckets and insights	8	cross context channel size
Insight top k displayed to the LLM	10	displayed insights cap
Per context insight list capacity	5	lowest support eviction
Deploy pool size K	10	band filter top buckets
Deploy min trials per candidate $n_{\min}$	3	reject single trial lucky peaks
Deploy band source	solver specs	literature ranges, see App. E

A.3 Evidence packet assembly ϕ

The assembly function $\phi(\mathcal{H}_t^c, \mathcal{K}, c)$ emits one prompt string with three concatenated blocks. The first is a *context header* naming the problem family, scale, objective count, solver, optimization direction, tunable parameter ranges, and the budget position t of T. The second lists the four evidence channels, namely local history, coverage map, parameter direction signals, and retrieved evidence, together with a reliability summary that flags how many unique configurations have multi trial support. The third lists the citable ids and the output schema for the proposal. The header and the closing block are fixed in size, while the evidence block grows with $|\mathcal{K}|$ up to the retrieval top k recorded in Table 8.

A.3.1 Structured retrieval across three paths

Prompt assembly draws from three evidence sources, namely the learned trial list $\mathcal{T}$ and the learned insight list $\mathcal{S}$ inside the knowledge base $\mathcal{K}{=}(\mathcal{T}, \mathcal{S})$, plus the read-only static prior block that holds solver and problem descriptions written from public references. Retrieval runs independently along three paths, all sharing one taxonomy overlap primitive and no learnt weights.

Path one, static prior. Each solver entry and each problem entry lives as a structured record in the static prior block, separate from the learned KB. At prompt assembly the renderer pulls the record keyed by the current solver and the current family, then walks its pitfalls and tuning heuristics. Entries marked with an optional scope whitelist are kept only when every listed taxonomy dimension has an allowed value in the current context. Entries without a whitelist are always included. A wildcard on either side of a dimension match counts as agreement, which is what allows algorithm level solvers like MOEA/D or NSGA-III to inherit the family encoding at render time without a second lookup.

Path two, trial list. Step one separates same-context trials, whose source context equals the target context, from cross-context trials. Same-context trials feed the aggregated history, the coverage map, and the parameter direction signal directly. Step two keeps only cross-context trials whose source context matches the target on at least two of the three structural dimensions (solver class, solution encoding, family class), and groups the survivors into source-context buckets keyed by $(\text{family}, \text{scale}, n_{\text{obj}}, \text{solver})$. Step three ranks the buckets by structural overlap count with a secondary

tiebreak on bucket size, then renders the top buckets with their matching and differing dimensions and, for each bucket, the three best config profiles aggregated over the bucket.

Path three, insight list. Retrieval first keeps only insights whose declared scope admits the target context $\hat{c}$. Surviving insights are then ranked by structural overlap with the target first and then support count, so a routing task never surfaces high-support scheduling insights at the top of the list even when their scope happens to admit it. A fixed per-context capacity caps the store and the lowest-support entry is evicted on overflow, so accumulated multi-trial evidence is preferred over low-support claims. The full lifecycle, including dedup-merge of canonicalized text and the recency tiebreak that breaks ties between insights of equal overlap and support, is documented in the insight-lifecycle subsection of this part.

The three paths are joined only at the rendering step, where their outputs are concatenated into disjoint prompt blocks. Because ranking is taxonomy based throughout, a reviewer inspecting the trace can read off from each cited id which path it came from and on which dimensions it matched.

A.4 Prompt flow and trace validation

Each trial is exactly two LLM calls, propose and reflect. The *propose* call sees the system prompt of §A.9, the evidence packet $\mathcal{E}_t$ assembled by ϕ, a reliability summary, and the explicit list of citable ids, then returns a structured response with five fields, namely an analysis paragraph, a strategy sketch, the proposed configuration $\boldsymbol{\theta}_t$, the predicted outcome h_t, and a cited trace. The *reflect* call sees $\boldsymbol{\theta}_t$, the realized Q_t, h_t, a short aggregate stat block for this configuration's bucket, the displayed existing insights, and the same cross context evidence shown at propose time, and returns two fields, a short reflection that grades h_t against Q_t and an optional long insight written when the trial supports a new mechanism level claim.

The cited trace $\mathbf{r}_t$ is the entry point for validation. Each citation in $\mathbf{r}_t$ is checked against the ids rendered in the prompt, with trial citations matching aggregated history rows, insight citations matching displayed insights, cross context citations matching retrieved source context tags, and coverage gap citations matching parameter names. A trace that cites an id not in the rendered prompt is rejected, and an empty trace is accepted only when the prompt carries no prior evidence, so every accepted citation resolves end to end to a concrete evidence record that a reviewer can audit from the prompt alone.

As an implementation audit, all $1{,}489$ traces in the released 7B paper KB were parsed successfully and all stored objective values were finite. The 12 configurations selected by the frozen-deployment main-table pickers also satisfied their solver-local schemas.

A typical walk-through is that the agent reads the top rows of the aggregated history, notices that the best-so-far configuration has $n{=}1$, checks the cross-context block, cites a routing-class insight that low p_m helps on permutation encodings, proposes a refinement, and predicts an outcome. The validator confirms the cited ids exist in $\mathcal{E}_t$. The reflect call then inspects the realized Q_t against this config's prior mean and writes a one-sentence short reflection plus an optional canonicalized long insight, merged with any existing entry of the same hash or written as a fresh entry when the trial reveals a pattern not already covered.

A.5 Insight lifecycle

An insight is born inside the reflection call whenever the optional long insight field r_t^{long} is non-empty, and it carries a claim text, a scope dictionary that names the contexts the claim applies to, a support count n_I, a birth context, and housekeeping fields for the first seen and last used trial indices. The canonicalization step of Algorithm 1 hashes the text after lowercasing and whitespace normalization, then either merges the new claim with any existing entry of the same hash by incrementing n_I or admits it as a fresh entry on hash miss.

The store has two ways to lose an insight. Display selection first drops insights whose declared scope excludes the target context, and surviving insights are then ranked by structural overlap with the target first and then support count n_I, with the last-used trial index breaking ties so that a more recently cited insight surfaces above a stale one of equal support and the first-seen trial index serving as a final tiebreak, which means a routing task is never dominated by highly support scheduling insights.

When the per-context pool exceeds its capacity, the weakest insight in that context is evicted, so accumulated multi-trial evidence is preferred over weak claims. The same ranked list is shown to the LLM at both ends of a trial. At propose time the top few taxonomy-matched insights appear as cited evidence candidates, and at reflect time they appear again under an *existing insights* heading so the LLM can signal that the current claim is already captured and return an empty r_t^{long} rather than restating the same claim.

A.6 Solver invocation

Every tuner, namely ConfigAgent, SMAC3, irace, and the default baseline, calls one shared routine $Q(\sigma(\mathcal{I}; \boldsymbol{\theta}))$ per $(\boldsymbol{\theta}, \mathcal{I})$ pair that unpacks $\boldsymbol{\theta}$ into solver specific arguments, merges the fixed budget knobs of Table 15, invokes the solver, and returns the realized Pareto front together with the scalar training signal Q. Solver exceptions and wall-clock timeouts return a sentinel that is dropped from the aggregated history, the parameter direction signal, and the retrieval pool, while the failed call still counts as one trial of the per-context budget, so the tuner cannot dodge its budget by selecting configurations that fail.

A.7 Baseline configuration

SMAC3 v2.0 (Lindauer et al., 2022) runs a random forest surrogate with log expected improvement acquisition, and each SMAC3 evaluation averages over 5 training instances and 2 seeds per instance to control HV noise. irace v4.4 runs iterated F-racing in elitist mode with the Friedman test at $\alpha{=}0.05$, evaluating each configuration on at least 5 instances before elimination, matching the default first-test count reported by López-Ibáñez et al. (2016). The three outer loop budgets, $T{=}200$ for SMAC3, $T{=}800$ for irace, and $T{=}100$ for ConfigAgent, are not directly comparable because the per-trial solver workload differs by design. SMAC3 averages $5{\times}2{=}10$ solver calls per evaluation, irace consumes one solver call per experiment under elitist racing, and ConfigAgent runs one solver call per trial on the scheduled training instance. On multi-objective contexts, all three tuners share the same $N_{\text{pop}} \times G_{\max} = 2{,}500$ fitness budget per solver call under one common per-instance reference front, and on single-objective contexts they share the same per-solver wall-clock cap as well.

A.8 Benchmark families, variants, and solver portfolio

This appendix defines the evaluation space over which the paper's generalization claims are measured. The space crosses five problem families with three scales and four objective counts under a portfolio of six solvers that spans three structurally different parameter-space regimes, with the five base families in Table 9, the nine OOD variant and scale-shift settings in Table 10, and the family-to-solver mapping in Table 11. Parameters are classified as *budget* or *AC tunable* when optimized by the tuner, and the family-specific genetic operators of the MO evolutionary solvers are held fixed because those operators are dictated by the solution encoding rather than by the tuning choice.

All evolutionary algorithms share a fitness budget of $2{,}500$ evaluations at $N_{\text{pop}}{=}50, G_{\max}{=}50$, single-objective OR-Tools Routing and Gurobi contexts use a $60\,\text{s}$ per-instance wall-clock cap, and LKH-3 uses a size-aware time budget of $120\,\text{s}$ on standard synthetic scales and $300\,\text{s}$ on variant scales and TSPLIB hard instances. Every tuner calls LKH-3 with 5 restarts per evaluation, with the full per-solver parameter listing in Table 15, so the returned score averages out LKH's own seed variance and the agent runs the same restart count as the baselines, which keeps the TSP and CVRP rows a like-for-like comparison across tuners.

Each variant preserves the base family solver, encoding, and family class. The five primary-scale variants in the upper block of Table 10, namely ATSP, MDK, Mixed-size BPP, Heterogeneous FJSP, and Clustered CVRP, match four of the five taxonomy dimensions and align row by row with Table 3 of the body. The four scale-shifted variants in the lower block, namely Mixed scale TSP, Clustered TSP, Scale OOD Sched., and Scale OOD Rout., match three of the five dimensions. Cross-context retrieval transports the base family's trials and insights into the variant's prompt under the same ranking primitive used for in-distribution contexts, so the deploy protocol stays unchanged and only the test instance distribution shifts. The default solver is LKH-3 or Gurobi on single-objective rows of routing or MIP-eligible families, and NSGA-II or a sibling MOEA on every multi-objective row.

Table 9: Base problem families. Starred scale is primary, the other two are held out for cross-scale generalization. MOEAs denotes NSGA-II, NSGA-III, and MOEA/D.

Family	Encoding	Scales	Objectives	Solver portfolio
TSP	permutation	$n \in \{100, \mathbf{500}^\star, 1000\}$	tour dist, max edge, min edge, edge std, edge range	MOEAs, OR-Tools, LKH-3, Gurobi
Knapsack	binary	$n \in \{100, \mathbf{500}^\star, 1000\}$	total value on k independent value dimensions	MOEAs, OR-Tools, Gurobi
BPP	permutation	$n \in \{200, \mathbf{500}^\star, 1000\}$	bin count, max bin load, load variance, neg min utilization, wasted space	MOEAs, OR-Tools, Gurobi
FJSP	dual layer	$j{\times}m \in \{5, \mathbf{10}^\star, 25\}{\times}5$	makespan, balance, avg flow, total work, max flow	MOEAs, OR-Tools, Gurobi
CVRP	permutation	$n \in \{100, \mathbf{200}^\star, 500\}$	total dist, longest subroute, route count, max load, route std	MOEAs, OR-Tools, LKH-3, Gurobi

Table 10: OOD variant and scale-shift settings.

Variant	Encoding	Scale	Solver	Shift versus base family
ATSP	permutation	n=500	LKH-3 or NSGA-II	family transfer, asymmetric distance matrix
MDK	binary	n=500	Gurobi or NSGA-II	constraint transfer, several capacity rows instead of one
Mixed size BPP	permutation	n=500, wider items	Gurobi or NSGA-II	distribution transfer, wider item size range
Heterogeneous FJSP	dual layer	$j{\times}m$=10×5	Gurobi or NSGA-II	processing-time transfer, non-uniform machine speeds
Clustered CVRP	giant tour	n=200	OR-Tools or NSGA-II	distribution transfer, Gaussian-clustered customers
Mixed scale TSP	permutation	$n \sim U(100, 1000)$	LKH-3 or NSGA-II	scale transfer, instance size drawn from a wide range
Clustered TSP	permutation	n=500	LKH-3 or NSGA-II	distribution transfer, Gaussian clustered node sampling
Scale OOD Sched. j25m5	dual layer	25×5	NSGA-II	scale transfer, shop size $2.5\times$ the primary scale
Scale OOD Rout. cvrp500	giant tour	500 customers	NSGA-II	scale transfer, customer count $2.5\times$ the primary scale

Table 11: Solver portfolio.

Solver	Class	AC tunable parameters	*Problem family* TSP	Knap.	BPP	FJSP	CVRP
Multi objective evolutionary							
NSGA-II	Dominance based	p_c, p_m	✓	✓	✓	✓	✓
NSGA-III	Reference direction	p_c, p_m, H	✓	✓	✓	✓	✓
MOEA/D	Decomposition	p_c, p_m, T, decomposition	✓	✓	✓	✓	✓
Single objective routing and local search							
OR-Tools (Routing)	Local search portfolio	first solution strategy, metaheuristic	✓	–	–	–	✓
OR-Tools (CP-SAT)	Constraint programming	search branching, linearization level, cp_model_presolve	–	–	✓	✓	–
OR-Tools (Knapsack)	Multi-dimensional B&B	solver backend (MD B&B / DP / 64-item)	–	✓	–	–	–
LKH-3	Lin–Kernighan heuristic	move type, maximum candidates	✓	–	–	–	✓
Mixed integer exact							
Gurobi	Branch and cut	heuristics weight, presolve, cuts, MIP focus	✓	✓	✓	✓	✓

A.8.1 MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

The three multi-objective algorithms differ in their tunable parameter set, summarized in Table 12. NSGA-II is Pareto dominance based and exposes the two probabilities p_c and p_m. NSGA-III replaces dominance ranking with reference direction based selection and adds a third tunable, the number of reference directions H, on top of p_c and p_m. MOEA/D decomposes the multi-objective problem into N_{pop} single-objective subproblems and adds two structural knobs, the neighbourhood size T and the decomposition method, which is Tchebycheff, weighted sum, or PBI. We keep N_{pop} fixed at 50 rather than scaling it with H or with the decomposition method so the fitness budget stays invariant across the three algorithms, leaving the tuner a configuration space of (p_c, p_m), (p_c, p_m, H), and $(p_c, p_m, T, \text{decomposition})$ for NSGA-II, NSGA-III, and MOEA/D respectively.

Table 12: MOEA parameters. Role column shows B for budget and T for tunable.

Algorithm	Parameter	Type	Range	Default	Role
NSGA-II	N_{pop} / $G_{\max}$	int/int	50 / 50	50 / 50	B
	p_c	float	$[0.5, 1.0]$	0.9	T
	p_m	float	$[0.0, 0.3]$	0.02	T
NSGA-III	N_{pop} / $G_{\max}$	int/int	50 / 50	50 / 50	B
	p_c / p_m	float	$[0.5, 1.0]$ / $[0.0, 0.3]$	0.9 / 0.02	T
	H (#reference directions)	int	$[4, 20]$	12	T
MOEA/D	N_{pop} / $G_{\max}$	int/int	50 / 50	50 / 50	B
	p_c / p_m	float	$[0.5, 1.0]$ / $[0.0, 0.3]$	0.9 / 0.02	T
	T (neighbourhood size)	int	$[10, 40]$	20	T
	decomposition	cat.	Tchebycheff / WS / PBI	Tchebycheff	T

Family specific genetic operators. The crossover and mutation operators are dictated by the solution encoding and held fixed, as listed in Table 13. TSP, BPP, and Routing use permutation encoding with Order crossover and shuffle or reverse-sequence mutation, Knapsack uses binary encoding with two-point crossover and per-gene bit flip, and Scheduling uses a dual-layer encoding whose assignment layer takes two-point or uniform crossover with machine-selection mutation and whose sequence layer takes POX precedence-preserving crossover with sequence exchange.

A.8.2 MULTI-OBJECTIVE DEFINITIONS AND HYPERVOLUME

Per family objective functions are listed in Table 14. At a given objective count we select the first n_{obj} rows, and maximization objectives are negated so HV computation runs on a minimization tuple.

Table 13: Genetic operators per family.

Family	Encoding	Crossover	Mutation
TSP / BPP / Routing	Permutation	Order crossover (OX)	Shuffle / reverse sequence
Knapsack	Binary	Two point	Bit flip (per gene p_m)
Scheduling	Dual layer	Two point / Uniform + POX	Machine sel. + seq.-exchange

We report hypervolume in the original objective scale (Zitzler et al., 2003); the scale factors shown in the result tables therefore reflect the units of each problem family.

HV is computed per instance using one reference point shared by all methods and seeds on that instance. For each test instance we merge the Pareto fronts produced by every method and every seed, take the union R as the approximation of the reference front, and read off the ideal point $\mathbf{z}^* = \min R$ and the nadir point $\mathbf{z}^{\text{nad}} = \max R$. The reference point is set componentwise to $r_j = z_j^{\text{nad}} + 0.1\,(z_j^{\text{nad}} - z_j^*)$ in the original objective scale, and the indicator is $\text{HV}(P) = \text{vol}(\bigcup_{\mathbf{p}\in P}[\mathbf{p}, \mathbf{r}])$. This makes methods directly comparable on the same instance and context, while absolute HV values are not compared across problem families with different objective scales.

Table 14: MO objective definitions.

Family	Obj Number	Objective	Direction
TSP	1	Total tour distance	↓
	2	Maximum edge length	↓
	3	Minimum edge length	↓
	4	Standard deviation of edge lengths	↓
	5	Edge length range, max minus min	↓
Knapsack	1	Total value on value dimension 1	↑
	2	Total value on value dimension 2	↑
	3	Total value on value dimension 3	↑
	4	Total value on value dimension 4	↑
	5	Total value on value dimension 5	↑
BPP	1	Number of bins	↓
	2	Maximum bin load	↓
	3	Load variance across bins	↓
	4	Negated minimum bin utilization	↓
	5	Wasted space	↓
Scheduling	1	Makespan	↓
	2	Balanced workload	↓
	3	Average flowtime	↓
	4	Total workload	↓
	5	Maximum flowtime	↓
Routing	1	Total distance	↓
	2	Longest subroute	↓
	3	Number of routes	↓
	4	Max route load	↓
	5	Route length standard deviation	↓

Objective design revisions An initial version of this benchmark paired value with weight on Knapsack and paired bin count with wasted space on BPP, which produce degenerate Pareto fronts because in both cases the two axes are linear combinations of the same underlying packing variable. On Knapsack the empty selection achieves the ideal point on the weight axis and dominates a trivial portion of the front regardless of solver quality. On BPP the wasted space equals the bin count times the capacity minus the total item weight, which is constant for any fixed instance, collapsing the two axis problem onto a line.

For Knapsack, we use a multi-objective extension of the 0/1 formulation in which each item carries k independent value scores and the k objectives are the sums of selected items on each score. This follows the profit-vector construction used in multi-objective Knapsack benchmarks (Zitzler & Thiele, 1999), while our benchmark retains one shared weight-capacity constraint. For BPP we replace

wasted space with the maximum bin load as the second objective, which produces a real trade off where fewer bins force higher peak loads while more bins distribute items more loosely. On TSP the fifth objective moves from mean edge length, which equals total distance divided by the tour size, to the edge length range, which carries information independent of the tour total. The third TSP objective also swaps from edge standard deviation to minimum edge length so that the three objective version reads total distance, max edge, min edge, a bound oriented triple where none of the three is implied by the other two. On Routing the fifth objective moves from average route distance, which equals total distance divided by the route count, to the standard deviation of per route distances.

All five families share a single rule on the first two objectives, namely both axes vary with the packing, routing, or scheduling decision and neither axis is derivable from the other by a constant affine transform of the instance data.

A.8.3 SINGLE-OBJECTIVE SOLVERS

Single-objective settings span three structurally different parameter-space regimes, each exercising a different property of the tuner, namely a heuristic local search engine, a constraint-programming portfolio, and a mixed-integer exact solver. We describe each solver briefly below and list every tunable parameter in Table 15.

LKH-3 (Helsgaun, 2017) is a Lin–Kernighan style heuristic local search solver that handles both TSP and capacity constrained vehicle routing variants such as CVRP and VRPTW under the same Lin Kernighan Helsgaun framework, so we apply it to both our TSP and CVRP families. We invoke LKH-3 through the binary subprocess with a size aware time budget of $120\,\mathrm{s}$ for standard synthetic instances and $300\,\mathrm{s}$ for variant scales and TSPLIB hard instances in App. C.3. Every tuner uses 5 LKH restarts per evaluation so the returned score averages out LKH's seed level stochasticity, which means ConfigAgent and the classical baselines receive the same per evaluation information content. The tuner selects the Lin–Kernighan move type from 3 opt, 4 opt, and 5 opt, and the candidate set size $\in [3, 15]$, which controls the neighbourhood breadth at each edge. The search space is entirely integer or small categorical, which makes surrogate modeling noisy and rewards a tuner that can reason about the named choices directly.

OR-Tools Routing applies to the TSP and CVRP families. It is a engine that first builds an initial tour via a user-chosen construction strategy from seven named options ranging from `PATH_CHEAPEST_ARC` to `CHRISTOFIDES`, then iteratively improves it with a local-search metaheuristic chosen from `GUIDED_LOCAL_SEARCH`, `SIMULATED_ANNEALING`, or `TABU_SEARCH`. With the $60\,\mathrm{s}$ wall-clock cap held fixed, a regime where an LLM that can reason linguistically about what each strategy does is expected to have an advantage over a surrogate restricted to encodings of the same names.

OR-Tools Knapsack offers three algorithmic backends, namely multi-dimension branch and bound, dynamic programming, and the specialized 64-item solver, and the selection among them is the only tunable, with the right backend depending on the item count profile of the instance distribution. This is a pure three-way categorical decision over named backends with no continuous knob to tune.

Gurobi is a commercial MIP solver applied to Knapsack, BPP, and FJSP through family specific MIP formulations. We hold the time limit at $60\,\mathrm{s}$, the MIP gap at 0.01, and the thread count at 1. The tuner sets the primal heuristic weight, the presolve level chosen from automatic, off, conservative, or aggressive, the cut aggressiveness, and the MIP focus mode chosen from feasibility, optimality, balanced, or lower bound priority. This is the hardest search space for a black box surrogate because the interactions between presolve level, cut aggressiveness, and focus mode are highly non-additive across the instance distribution.

The seven OR-Tools Routing first solution strategies tuned in the configuration space are path cheapest arc, savings, Christofides, path most constrained arc, global cheapest arc, local cheapest arc, and first unbound min value.

A.8.4 INSTANCE GENERATION

All synthetic instances are generated under the following templates. **TSP**: Euclidean instances with n cities, coordinates uniform in $[0, 10000]^2$, in TSPLIB format. The five MO objectives in Tab. 14 are all derived from the same edge-length vector and need no extra instance fields. **Knapsack**: n

Table 15: SO solver parameters. Role column shows B for budget and T for tunable.

Solver	Parameter	Type / Range	Default	Role / Applies to
OR-Tools Routing	time limit	int (60 s, fixed)	60 s	B / TSP, CVRP
	first solution strategy	cat., 7 options	path cheapest arc	T / TSP, CVRP
	local search metaheuristic	cat., GLS / SA / TS	GLS	T / TSP, CVRP
OR-Tools Knapsack	solver backend	cat., MD B&B / DP / 64-item	MD B&B	T / Knapsack
OR-Tools CP-SAT	search branching	cat., automatic / fixed / portfolio	automatic	T / BPP, FJSP
	linearization level	cat., $\{0, 1, 2\}$	1	T / BPP, FJSP
	presolve switch	cat., {on, off}	on	T / BPP, FJSP
Gurobi	time limit	float (60 s, fixed)	60 s	B / all families
	MIP gap	float (0.01, fixed)	0.01	B / all families
	threads	int (1, fixed)	1	B / all families
	heuristics weight	float, $[0, 1]$	0.05	T / all families
	presolve	cat., $\{-1, 0, 1, 2\}$	-1	T / all families
	cuts	cat., $\{-1, 0, 1, 2, 3\}$	-1	T / all families
	MIP focus	cat., $\{0, 1, 2, 3\}$	0	T / all families
LKH-3	runs	int, 5	5	B / TSP, CVRP
	time limit	int, 120 or 300 s size aware	120 s	B / TSP, CVRP
	move type	cat., $\{3, 4, 5\}$	5	T / TSP, CVRP
	maximum candidates	int, $[3, 15]$	5	T / TSP, CVRP

items with weights uniform in $[1, 100]$ and capacity $= 50\%$ of total weight. In the single-objective setting each item carries one value $v_i = w_i + \mathcal{U}(-10, 10)$, clipped at 1. In the multi-objective settings we draw one extra value vector per active objective under the same rule, so each item ships an n_{obj}-dimensional value vector and the j-th objective is the total of value dimension j. **BPP**: n items with sizes uniform in $[1, C/2]$, bin capacity $C = 100$, First-Fit-Decreasing seed. The five MO objectives in Tab. 14 are all derived from the same bin assignment and need no extra instance fields. **FJSP**: j jobs $\times$ m machines with sequence dependent setup times. Each operation can run on any of $\lfloor m/2 \rfloor$ alternative machines drawn uniformly without replacement from the m machines, giving an average flexibility of $0.5{\cdot}m$ per operation. Processing times on each admissible machine are drawn uniformly in $[1, 99]$. The five MO objectives in Tab. 14 are derived from the produced schedule and need no extra instance fields. **CVRP**: n customers with coordinates in $[0, 1000]^2$, demands uniform in $[1, 30]$, vehicle capacity scaled to require 5–20 routes. The five MO objectives in Tab. 14 are all derived from the same route assignment and need no extra instance fields. Each family and scale pair provides 100 training and 100 held-out test instances.

A.9 AC AGENT PROMPT TEMPLATES

The agent uses three prompt templates, shown below. Together they implement the paper's second contribution that every configuration proposal is accompanied by a typed citation list whose ids a deterministic validator checks against the rendered prompt, and they enforce two guards the LLM often gets wrong on optimization tasks, namely the direction of improvement and the distinction between exact and heuristic solvers under multi-objective settings. Placeholders appear in blue and the full prompts live in the released code, and App. A.10 walks through a filled-in instance of the propose prompt with its response so the reader can read an audited trace end to end.

A.9.1 SYSTEM PROMPT

System Prompt

You are an algorithm configuration agent for combinatorial optimization solvers. Your goal: find the parameter setting that maximizes solver performance on a target problem distribution.
You are given the full search space, the current coverage map summarizing how many trials hit each parameter region, the aggregated experiment history with per config mean, std, and sample count, the budget position as trial X of Y, and any accumulated long term insights or cross problem evidence.
Respond with valid JSON only, no text outside the JSON block. Every field is required. **You decide** exploration vs. exploitation, when to verify a promising single trial configuration, and when to go into unexplored regions—do not defer to external scheduling rules.

A.9.2 PROPOSE PROMPT TEMPLATE

Propose Prompt rendered at trial t

Problem: {family} ({scale}, {n_{obj}} objective), solver: {solver}, direction: {higher is better / lower is better}.
Solver mechanism: {rendered solver_specs entry for this solver — mechanism sentence, per parameter semantics, literature band, encoding notes, pitfalls filtered by scope, tuning heuristics, references}.
Problem context: {rendered problem_specs entry for this family — description, encoding, objective list, variants sharing this base family, benchmark libraries targeting this family}.
Default configuration: {published default cfg rendered as key equals value pairs, first class candidate in the pool}.
Tunable parameters: {per parameter range or options, and default}.
Coverage: {per parameter: tried values, unexplored bins, untried categorical levels, plus a categorical level summary with per level n, mean, best}.
Reliability summary: {number of configs tested once versus $n{\geq}2$; a warning when every config is $n{=}1$}.
Budget: trial {t} of {T}, with {$T - t$} remaining.
Aggregated history sorted by direction, top ten plus bottom five: {rows of (params, $\bar{Q}$, σ_Q, vs_default, n)}.
Parameter direction signal (target context): {per parameter Spearman ρ with best and worst observed values}.
Recent reflections latest per top config: {short reflections attached to the leading configurations}.
Raw trials: {raw (cfg, score) pairs as an OPRO style fallback signal}.
Accumulated insights: {insight id, support count, context, topic, param, scope, text, mechanism — top-k taxonomy ranked}.
Cross problem evidence: {top-k source contexts, each annotated with matching and differing taxonomy dimensions, with their top three config profiles, plus a strict pool and loose pool contrastive pattern block}.
Citable evidence ids: {available trial_ids, insight_ids, cross problem context strings, coverage gap param names}.
Task: propose the next configuration and a cited trace. Each cited id must exist above; fabricated ids are forbidden. A trace may be empty only at $t{=}1$.
`{analysis, strategy, config, hypothesis, trace: [{kind, id, why}, ...]}`

A.9.3 Reflection Prompt Template

Reflection Prompt rendered after execution

Configuration: {θ_t}. **Hypothesis:** {predicted outcome}. **Actual result:** {Q_t} versus default = {$\bar{Q}_{\text{default}}$}, direction {BETTER or WORSE}, relative delta {rel percent}.
Aggregated stats for this config: {if tested before: n, $\bar{Q}$, σ_Q, and whether the current result is above or below its own mean; otherwise the entry is the first test of this config}.
Nearest neighbour in parameter space: {closest prior trial cfg by range normalized distance, with its HV and trial id, so the short reflection can ground in a concrete cfg delta versus HV delta}.
Existing insights: {top-k taxonomy ranked insights: id, support count, topic, param, text, mechanism — the agent is expected to NOT restate what is already in this list}.
Cross problem evidence: {the same block rendered at propose time so reflections can explicitly reference related contexts}.
Rules: (i) the short reflection grades the hypothesis against the realized result and compares the current outcome to the nearest neighbour; (ii) check existing insights first, and if this trial confirms an existing one, set `long_insight` to the empty string; (iii) write a new `long_insight` only when multiple independent trials exhibit a consistent directional pattern or this trial contradicts a cited insight; (iv) every new `long_insight` must explain the mechanism at the encoding level, not just the direction.
`{short_reflection, long_insight, insight_fields: {topic, param, scope, mechanism, contradicts}}`

Interface summary. Each trial is exactly two LLM calls. *Propose* consumes the evidence packet above and returns the analysis paragraph, the strategy sketch, the proposed configuration, the predicted outcome, and the cited trace. The solver then runs with the proposed configuration on the current training instance, and *reflect* consumes the realized quality together with the existing insights and cross context blocks and returns the short reflection and the optional long insight, with no intermediate agent state outside $\mathcal{K}$ carried across trials within a context.

A.10 A concrete filled-in propose prompt

This subsection fills in the propose prompt template with real numbers from a single trial. The context is FJSP at the j10m5 primary scale under NSGA-II at 2 objectives. Trial t=20 of T=100 runs after 19 prior trials already in $\mathcal{H}^c$. The training instance has 10 jobs on 5 machines with an average of 2.3 alternative machines per operation, integer processing times in $[1, 10]$, and HV reference point $(1.30 \times 10^3, 4.80 \times 10^3)$.

Propose prompt — FJSP j10m5, 2 obj, trial 20/100

```
(1) Problem
    FJSP (j10m5, 2 objective), solver NSGA-II
    direction: higher is better (HV)

(2) Tunable parameters  /  Budget
    cxpb  : float in [0.5, 1.0]   default 0.9
    mutpb : float in [0.0, 0.3]   default 0.02
    fixed : population_size=50, ngen=50
    trial 20 of 100; 80 remaining

(3) Coverage
    cxpb  : 11 unique in [0.62, 0.97]    bin [0.50, 0.60] empty (1/5)
    mutpb :  8 unique in [0.005, 0.07]   bin [0.24, 0.30] NEVER tested

(4) Reliability summary
    15 unique configs tested: 12 only once (n=1, unreliable); 3 with n>=2.


(5) Aggregated history (top 5 of 15, sorted by HV; n = repeats)
      cxpb   mutpb   HV_mean   HV_std   vs_default   n
      0.85   0.015   3.77e4    1.2e3    +6.3%        2
      0.88   0.013   3.75e4    -        +5.7%        1
      0.82   0.020   3.71e4    -        +4.6%        1
      0.80   0.025   3.66e4    -        +3.1%        1
```

```
      0.90   0.020   3.55e4    1.8e3    +0.0%        3   <- default
 bucket


(6) Parameter direction signal (target context)
    cxpb  : spearman_rho = +0.512   best@0.85 (3.77e4)   worst@0.62 (3.38
 e4)
    mutpb : spearman_rho = -0.441   best@0.013 (3.77e4)  worst@0.070
 (3.42e4)

(7) Recent reflections (latest per top config)
    [cxpb=0.85, mutpb=0.015]  Confirmed twice; mean now 3.77e4.
                              Increase cxpb further to test ceiling.

(8) Accumulated insights
    id=I-perm-1  support=3  context=FJSP/j10m5/2-obj/NSGA-II
      "Higher cxpb with low to moderate mutpb improves HV for permutation-

       encoded scheduling because OX preserves partial machine
 assignments
       while mutation would disrupt them."

(9) Cross problem evidence (1 source context; solver class plus objective
   count match)
    TSP/n500/2-obj/NSGA-II   (n=50)
      top: (cxpb=0.92, mutpb=0.01)   (cxpb=0.93, mutpb=0.01)
      differs on: solution type, family class, scale

(10) Citable ids
     trial       : [0, 1, ..., 18]
     insight     : [I-perm-1]
     cross ctx   : [TSP/n500/2-obj/NSGA-II]
     coverage    : [cxpb, mutpb]

(11) Task
     Propose the next trial configuration. Return 5-field JSON
     {analysis, strategy, config, hypothesis, trace}.
     CRITICAL: runs are stochastic; a config tested once has an
     unreliable mean.
```

Agent response:

```
{
  "analysis": "Best verified is cxpb=0.85, mutpb=0.015 (n=2, HV 3.77e4).
    Direction signal is positive on cxpb (+0.51) and negative on mutpb
    (-0.44). The region cxpb>=0.9 is undersampled in target but is the
    same region that wins on TSP/n500/2-obj/NSGA-II, and insight
    I-perm-1 predicts high cxpb / low mutpb should transfer here.",

  "strategy": "Mid budget refine: extrapolate the cxpb direction into
    the underexplored [0.9, 1.0] bin while keeping mutpb near the low
    sweet spot.",

  "config": {"cxpb": 0.93, "mutpb": 0.012},

  "hypothesis": "Expect HV > 3.78e4: +0.08 cxpb vs best verified
    leverages the positive rho; mutpb=0.012 stays in the insight's
    recommended range.",

  "trace": [
    {"kind": "trial",        "id": 2,            "why": "best verified n
 =2"},
```

```
    {"kind": "insight",      "id": "I-perm-1",   "why": "high cxpb + low
  mutpb"},
    {"kind": "cross_context","id": "TSP/n500/2-obj/NSGA-II",
                             "why": "same solver class, top region cxpb
  ~0.93"},
    {"kind": "coverage_gap", "id": "cxpb",       "why": "bin [0.5, 0.6]
  empty"}
   ]
}
```

Trace decoded. The validator described in Appendix A.3.1 checks each cited id against the entries that render blocks 3, 5, 8, and 9. Block 10 is a convenience reprint of those entries. The four trace entries resolve as follows. The trial citation points at row 1 of block 5. The insight citation points at the sole insight rendered in block 8. The cross context citation points at the sole source context rendered in block 9. The coverage gap citation points at the p_c gap reported in block 3. Every cited id resolves and the proposal is accepted. The configuration with p_c=0.93 and p_m=0.012 is passed to NSGA-II on the training instance and produces HV $= 3.81 \times 10^4$, a new best for the context that becomes the seed for the reflection call that follows.

A.11 Statistical significance on the main body contexts

Practical effect sizes. Relative to the stronger of SMAC3 and irace in each row, the displayed ConfigAgent margins are 0.012%–1.94% across the 12 single-objective settings (median 0.085%) and 0.021%–0.304% across the 15 two-objective settings (median 0.116%). Together, these results show competitive or improved quality with fewer target-side configuration trials.

Table 16 reports paired tests for 11 primary two-objective comparisons. Each comparison uses the same 100 held-out instances and a one-sided Wilcoxon signed-rank test with the alternative that ConfigAgent has higher HV. The p_{H} columns apply Holm correction jointly to the 22 ConfigAgent–baseline comparisons.

Table 16: Paired analysis on 11 primary two-objective comparisons. Means and p_{H} are computed from n=100 per-instance results.

Family / scale	**Alg.**	**ConfigAgent**	**SMAC3**	p_{H}	**irace**	p_{H}
TSP / n500	NSGA-II	$2.3453{\cdot}10^8$	$2.3307{\cdot}10^8$	0.211	$2.3410{\cdot}10^8$	1.000
	NSGA-III	$2.2616{\cdot}10^8$	$2.2220{\cdot}10^8$	< 0.001	$2.2732{\cdot}10^8$	1.000
	MOEA/D	$2.4910{\cdot}10^8$	$2.4885{\cdot}10^8$	1.000	$2.5823{\cdot}10^8$	1.000
Knapsack / n500	NSGA-II	$1.8833{\cdot}10^9$	$1.8785{\cdot}10^9$	0.004	$1.8847{\cdot}10^9$	1.000
	NSGA-III	$1.8654{\cdot}10^9$	$1.8626{\cdot}10^9$	0.267	$1.8705{\cdot}10^9$	1.000
	MOEA/D	$2.0019{\cdot}10^9$	$2.0015{\cdot}10^9$	1.000	$2.0099{\cdot}10^9$	1.000
BPP / n500	NSGA-II	$4.7318{\cdot}10^3$	$4.7290{\cdot}10^3$	< 0.001	$4.7321{\cdot}10^3$	1.000
Scheduling / j10m5	NSGA-II	$2.0357{\cdot}10^6$	$2.0391{\cdot}10^6$	1.000	$2.0400{\cdot}10^6$	1.000
	MOEA/D	$2.0087{\cdot}10^6$	$2.0215{\cdot}10^6$	1.000	$2.0226{\cdot}10^6$	1.000
Routing / cvrp200	NSGA-II	$4.1412{\cdot}10^4$	$4.0644{\cdot}10^4$	< 0.001	$4.1787{\cdot}10^4$	1.000
	MOEA/D	$4.5879{\cdot}10^4$	$4.0632{\cdot}10^4$	< 0.001	$4.1044{\cdot}10^4$	< 0.001

The paired analysis finds a Holm-corrected difference against SMAC3 for BPP NSGA-II and against both baselines for Routing MOEA/D.

B Part B: Efficiency

Part B collects the evidence behind the efficiency claim of the main body across four cuts.

§B.1 sweeps the deploy picker and reports a band-matched SMAC3 control, separating pool width from the numerical contribution of the static prior. §B.2 reports event-level timing, the per-trial component profile, and the LLM-latency threshold. The directly measured variant runs in Table 33 separate zero-shot evaluation, $T{=}10$ adaptation, and final-test cost instead of folding them into one estimated speedup. §B.3 reports per-context best-so-far HV against trial budget and shows an early plateau in the reported solver parameter spaces. §B.4 reports that the 7B and 14B backbones are close to the 32B results on the primary contexts, providing a lower-memory option for the tested setup.

Unless otherwise stated, wall-clock time includes solver execution and LLM propose/reflect calls, but excludes one-time vLLM model loading, instance generation, and offline benchmark construction. Zero-shot redeploy time includes retrieval, deploy picking, and evaluating the selected configuration under the same solver budget used by the corresponding baseline.

Single-objective rows in Table 1 share the family's primary scale across all solvers within the family, but the per-solver wall-clock cap and the role of the solver differ. Gurobi runs as a MIP solver under the same 60 s cap and typically returns a feasible incumbent rather than a proven optimum, while LKH-3 and OR-Tools are heuristic engines whose designs target the same primary scale. Table 1 therefore compares tuners within each fixed solver-family-scale context rather than asking which solver class wins outright on a shared instance.

Lifecycle-cost interpretation. For B source contexts and N target deployments, the displayed ConfigAgent lifecycle time is $B \times 3{,}800 + N \times C_{\mathrm{deploy}}$, compared with $N \times C_{\mathrm{retune}}$ for repeated target-side tuning. At the reported rental price of USD 0.882 per hour, constructing a 12-context KB takes 45,600 seconds (12.67 hours; USD 11.18) and breaks even after 3–8 related targets. Constructing a 57-context KB takes 216,600 seconds (60.17 hours; USD 53.09) and breaks even after 11–36 targets. ConfigAgent uses GPU inference in source training and few-shot adaptation, whereas SMAC3 and irace are CPU tuners; frozen zero-shot deployment makes no LLM call.

B.1 Deploy picker sensitivity

This appendix reports deploy-picker sensitivity for $K \in \{1, 3, 5\}$ on Knapsack, BPP, and Scheduling. Across these contexts, the selected configuration changes with K, while the range of mean HV remains within 0.2%, showing stable quality across the sweep.

Table 17: Pool width K sensitivity on $n{=}100$ instances at the two-objective NSGA-II primary context.

context	$K{=}1$	$K{=}3$	$K{=}5$
Knapsack / n500	$1.8833{\cdot}10^{9}$	$1.8796{\cdot}10^{9}$	$1.8814{\cdot}10^{9}$
BPP / n500	$4.7318{\cdot}10^{3}$	$4.7326{\cdot}10^{3}$	$4.7315{\cdot}10^{3}$
Scheduling / j10m5	$2.0361{\cdot}10^{6}$	$2.0346{\cdot}10^{6}$	$2.0340{\cdot}10^{6}$

Table 18: Literature-band filter ablation on $n{=}100$ instances.

context	Band on	Band off	Δ HV
TSP / n500	$2.3453{\cdot}10^{8}$	$2.3453{\cdot}10^{8}$	0.00%
Knapsack / n500	$\mathbf{1.8833{\cdot}10^{9}}$	$1.8711{\cdot}10^{9}$	-0.65%
BPP / n500	$4.7318{\cdot}10^{3}$	$4.7318{\cdot}10^{3}$	0.00%
Scheduling / j10m5	$\mathbf{2.0376{\cdot}10^{6}}$	$2.0359{\cdot}10^{6}$	-0.09%

The band-filter records show no change for TSP and BPP and decreases of 0.09% for Scheduling and 0.65% for Knapsack when the filter is disabled, showing that the numerical prior has a context-dependent effect.

Band-matched numerical control. The preceding ablation changes the ConfigAgent picker. A complementary control instead gives SMAC3 the implementable numerical component of the same public prior. Band-SMAC3 searches only within the published parameter bands for 200 configuration trials, with three training instances per trial, and is then evaluated with the other methods on the same 100 held-out instances and three seeds. It does not receive the prose mechanism descriptions or the accumulated KB records.

Table 19: Static-prior control at the two-objective NSGA-II primary contexts. All quality entries are held-out HV (↑); time is target-side tuning time for Band-SMAC3.

Target	Default	Wide SMAC3	irace	Band-SMAC3	ConfigAgent	Band tune (s)
TSP n500	$2.252{\cdot}10^8$	$2.339{\cdot}10^8$	$2.346{\cdot}10^8$	$2.340{\cdot}10^8$	$2.343{\cdot}10^8$	840
KP n500	$1.875{\cdot}10^9$	$1.878{\cdot}10^9$	$1.885{\cdot}10^9$	$1.888{\cdot}10^9$	$1.883{\cdot}10^9$	717
BPP n500	4,730.434	4,729.150	4,731.900	4,729.750	4,731.684	1,841
FJSP j10m5	$2.038{\cdot}10^6$	$2.041{\cdot}10^6$	$2.042{\cdot}10^6$	$2.040{\cdot}10^6$	$2.038{\cdot}10^6$	2,549
CVRP n200	$4.130{\cdot}10^4$	$4.062{\cdot}10^4$	$4.176{\cdot}10^4$	$4.174{\cdot}10^4$	$4.170{\cdot}10^4$	865

Band-SMAC3 is competitive with or above ConfigAgent on some rows, confirming that the public numerical bands contribute to quality. It nevertheless uses 600 target-side solver evaluations and 717–2,549 seconds per context. ConfigAgent's frozen zero-shot mode instead amortizes the prior together with measured source records and performs no target-side retuning. This comparison separates the value of the numerical band from the value of reuse.

Trial budget sensitivity. Table 20 reports the $T{=}25$ and $T{=}50$ runs. Increasing the budget improves the displayed ConfigAgent test mean from $2.2515{\cdot}10^8$ to $2.2713{\cdot}10^8$.

Table 20: Verifiable trial-budget records on TSP $n{=}500$ at the two-objective NSGA-II primary context.

Trial budget	$T{=}25$	$T{=}50$
Recorded trials	25	50
Default test mean	$2.2262{\cdot}10^8$	$2.2114{\cdot}10^8$
ConfigAgent test mean	$2.2515{\cdot}10^8$	$2.2713{\cdot}10^8$

B.2 Per-trial timing breakdown

Table 21 reports event-level SMAC3 timing for the five two-objective NSGA-II primary contexts. The span is the last event end time minus the first event start time; the summed time adds the duration of every completed event. The two agree to within two seconds in every context, confirming serial execution.

Table 21: SMAC3 event timing at the two-objective NSGA-II primary context. "Recorded trials" is the number of completed events.

Context	Recorded trials	Event span (s)	Summed event time (s)
TSP / n500	100	1,977.6	1,976.0
Knapsack / n500	100	1,372.4	1,370.2
BPP / n500	38	2,246.7	2,246.6
Scheduling / j10m5	21	1,803.9	1,803.9
Routing / cvrp200	100	2,280.3	2,279.1

The BPP and Scheduling histories stop at 38 and 21 events, respectively, and must not be extrapolated to a nominal budget. The earlier aggregate timing rows that scaled these partial runs to a common budget are therefore omitted. Likewise, no speedup is computed by mixing these event spans with separately measured timing summaries.

Figure 3 plots ConfigAgent, SMAC3, and irace incumbent quality on three 2-objective MOEA/D primary contexts, namely TSP n500, KP n500, and CVRP n200. Each curve is normalized against the joint best final HV across the three tuners on that context, so the gap between curves at a given trial directly reads as a quality gap. Markers tag the first trial where each tuner reaches 99% of the joint best. ConfigAgent reaches the 99% band by trial 20 to 45, SMAC3 by trial 160 to 200, and irace by trial 720 to 800 within its $T{=}800$ budget.

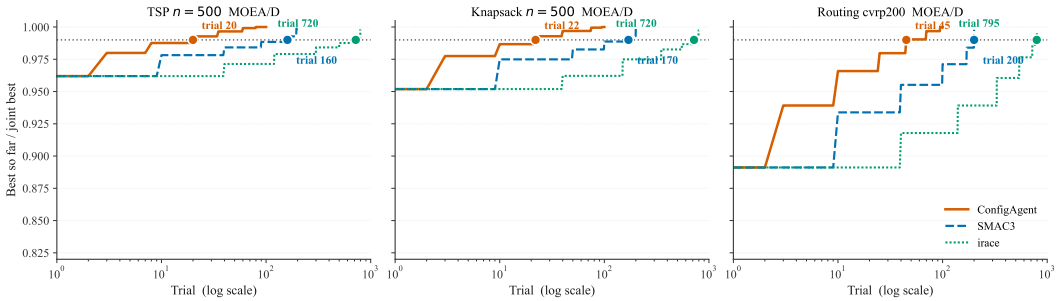

Figure 3: Convergence on three 2-objective MOEA/D primary contexts, namely TSP n500, KP n500, and CVRP n200, normalized to the joint best.

Table 22 reports the measured per-trial component profile at the two-objective NSGA-II primary context. The solver column records the reported average solver time and the LLM column the reported mean Qwen2.5 32B latency for one propose plus one reflect call. As a cross-check, the successful $T{=}10$ variant runs in Table 33 spend, per adaptation trial, 35.7, 33.3, 57.9, 65.1, and 35.6 seconds on TSP, Knapsack, BPP, Scheduling, and Routing, respectively. The BPP and Scheduling values are especially close to the 56.4 and 71.4 second primary-context profile below, and all five are of the same order. They are not expected to match exactly because the variant runs use different instance distributions and report the whole adaptation loop rather than the two rounded components below.

Table 22: Mean per trial wall clock at the 2-objective NSGA-II primary context, in seconds.

Family	LLM (s)	Solver (s)	Trial total (s)
TSP / n500	17.6	33.9	51.5
Knapsack / n500	17.2	26.3	43.5
BPP / n500	18.4	38.0	56.4
Scheduling / j10m5	19.1	52.3	71.4
Routing / cvrp200	18.7	34.2	52.9

When does LLM latency pay off? Few-shot configuration is advantageous in wall-clock time only when

$$(\text{saved solver evaluations}) \times (\text{solver time per evaluation}) > \text{LLM time} + \text{retrieval overhead}.$$

The propose–reflect pair in Table 22 takes 17.2–19.1 seconds, so a $T{=}10$ adaptation spends 172–191 seconds on LLM calls. Dividing this by the 190 trials saved relative to a $T{=}200$ SMAC3 run gives a conservative threshold of 0.91–1.01 seconds per saved trial. The measured solver component is 26.3–52.3 seconds per ConfigAgent trial, above this threshold in all five displayed contexts. For sub-second solvers, CPU-only AC may be faster and cheaper. This threshold is conservative because one SMAC3 evaluation in App. A.7 averages ten solver calls; zero-shot deployment makes no propose–reflect call.

B.3 ConfigAgent best-so-far by backbone

Figures 4–6 plot ConfigAgent best-so-far HV across three representative contexts on each of the three Qwen2.5 backbone sizes. All three backbones reach final HV by trial 40 of $T{=}100$ on every context. The red dashed line in each figure marks the trial at which best-so-far first reaches within one percent of its final value. The same pattern holds across the three MOEAs of Tab. 2, whose parameter spaces grow from 2 dimensions for NSGA-II to 3 dimensions for NSGA-III and 4 dimensions with one categorical knob for MOEA/D.

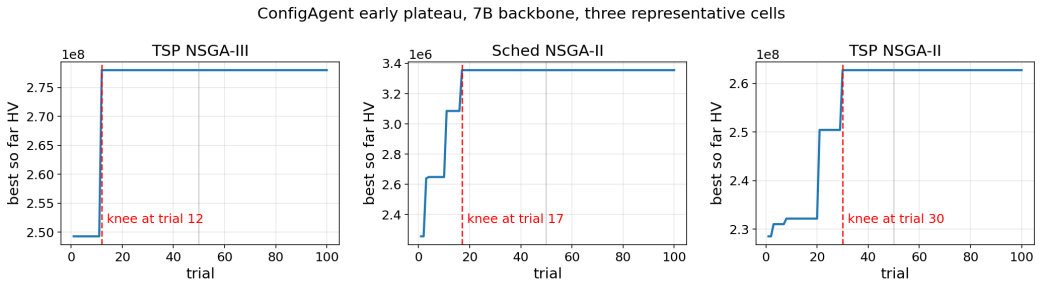

Figure 4: ConfigAgent best-so-far HV with the 7B backbone.

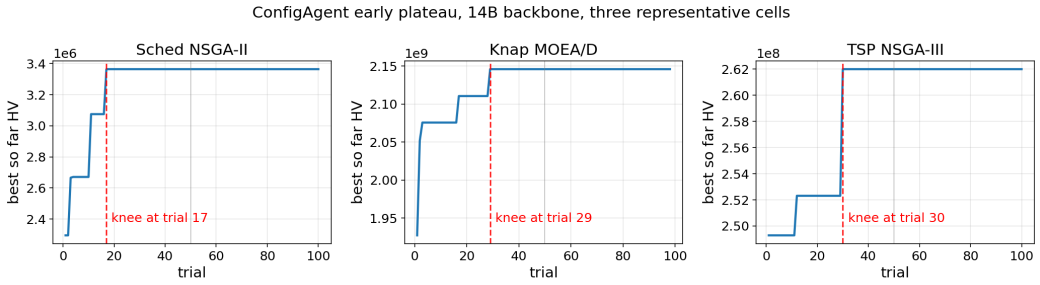

Figure 5: ConfigAgent best-so-far HV with the 14B backbone.

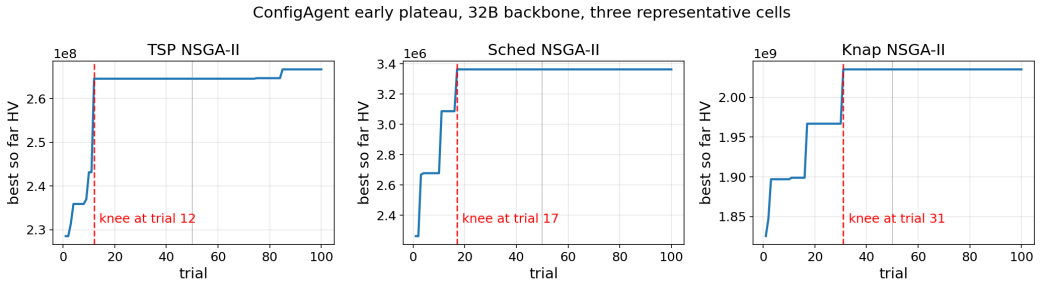

Figure 6: ConfigAgent best-so-far HV with the 32B backbone.

B.4 Backbone sensitivity

This subsection takes the complementary cut to the component ablation of App. D.1, fixing the pipeline and varying the backbone instead. Table 23 reports the sweep across three Qwen2.5 sizes, 7B, 14B, and 32B, on the five NSGA-II primary contexts under the same shared KB at T=100. The same vLLM interface, agent code, retrieval pipeline, prompt templates, and K=10 deploy picker are used across sizes, with each run using one backbone for proposal, reflection, and deployment selection to isolate the end-to-end backbone effect.

Across every row the three backbone sizes deliver similar deployed quality, with differences small relative to the primary-context gaps in the main body. This is the efficiency frontier reading. In these experiments, the 7B and 14B backbones follow the structured evidence packet closely enough to track the 32B result, so a practitioner can use a smaller backbone with little measured quality loss in the reported contexts. The Default column reports the published default evaluated on the same held-out test set under the DEAP per-gene runner.

C Part C: Generalization

Part C collects the evidence behind the generalization claim. Its frozen source knowledge base contains the 15 synthetic n_{obj}=2 primary contexts (five families and three MOEAs per family) and is

Table 23: Backbone sensitivity at the 2-objective NSGA-II primary context with $K=10$.

context	Default	Qwen-7B	Qwen-14B	Qwen-32B
TSP n500	$2.255{\cdot}10^{8}$	$2.331{\cdot}10^{8}$	$2.336{\cdot}10^{8}$	$\mathbf{2.339{\cdot}10^{8}}$
Knap. n500	$1.876{\cdot}10^{9}$	$1.879{\cdot}10^{9}$	$1.881{\cdot}10^{9}$	$\mathbf{1.883{\cdot}10^{9}}$
BPP n500	$4.731{\cdot}10^{3}$	$4.722{\cdot}10^{3}$	$4.730{\cdot}10^{3}$	$\mathbf{4.732{\cdot}10^{3}}$
Sched. j10m5	$2.039{\cdot}10^{6}$	$2.036{\cdot}10^{6}$	$2.041{\cdot}10^{6}$	$\mathbf{2.044{\cdot}10^{6}}$
Routing cvrp200	$4.123{\cdot}10^{4}$	$4.135{\cdot}10^{4}$	$4.150{\cdot}10^{4}$	$\mathbf{4.158{\cdot}10^{4}}$

distinct from the full 57-context evaluation set described in the main text. It transports across three transfer types and is never extended on any target reported in this part. For the TSP variants that share (p_c, p_m), Table 36 also supplies matched-budget Warm-SMAC, RGPE, and MI-SMBO-style controls; the other sections test shifts for which a shared numerical parameterization is not assumed.

The first transfer is *objective count*, which goes from 2 objectives to 3 and 5 objectives at the same family and scale. §C.1 reports the in-distribution retune reference table plus the zero-shot cross-objective result under the same tuners.

The second transfer is *scale*, which goes from the family primary scale to ten sampled non-primary scales, and from primary scale at $n_{\text{obj}}=2$ to non-primary scales at higher objective counts. §C.2 handles both cuts and also reports the leave-one-family-out transfer, which deploys a frozen KB on a held-out family under a zero new trial budget on the same evaluation pipeline.

The third transfer is *problem variant kind*, which goes from a base family to a sibling distribution that shifts one structural dimension or to a public benchmark library authored by the operations research community. §C.3 covers the five SO variants of the main body, the four MO sibling variants tabulated in this appendix, the SMAC3 configuration reuse comparison, and the five public libraries. §C.4 then walks through one trace example that exercises all three transfers at once.

C.1 Objective transfer, extensions, and cross-objective deployment

This subsection runs two regimes on the objective-count transfer, an in-distribution extension and a stricter out-of-distribution transfer, both at higher objective counts than the 2-objective body headline.

The in-distribution regime of Table 24 retunes every tuner, including ConfigAgent, separately at 3 and 5 objectives using the same five families and three MOEAs as Table 2.

The out-of-distribution regime of §C.1.1 keeps the ConfigAgent KB frozen at $n_{\text{obj}}=2$. The evaluated SMAC3 and irace baselines are retuned from scratch at each held-out objective count because their implementations in this study do not include a transfer step.

Tables 26–29 jointly report transfer across structural variants and objective counts. Each table covers one variant—ATSP, Mixed-scale TSP, Clustered TSP, or MDK—at 2, 3, and 5 objectives under NSGA-II, NSGA-III, and MOEA/D. SMAC3 and irace are retuned with $T=200$ and $T=800$, respectively, whereas ConfigAgent uses zero-shot deployment. The retrieved settings remain competitive as the objective count changes. Directly measured timing results are reported in Tables 33 and 34, and additional scale-transfer results appear in Tables 30 and 31.

C.1.1 Cross-objective transfer, zero-shot from 2-objective KB

The from-scratch SMAC3 and irace implementations evaluated here tune each held-out objective count independently. ConfigAgent instead ranks configurations stored in the 2-objective KB for a 3- or 5-objective target when the solver, encoding, and family labels still match.

Two mechanisms survive the objective count shift without modification, namely the literature mutation prior that applies on permutation or bit string encodings at any objective count, and the Tchebycheff scalarization claim on MOEA/D that the scope whitelist keeps unrestricted across objective counts. Mechanisms that do depend on the objective count, such as the reference direction count H on NSGA-III and the decomposition neighbourhood size T on MOEA/D, are scoped in the solver spec so the agent only inherits the invariant claims and defers to its own target trials for the dimension that shifts.

Table 24: Results at 3-objective and 5-objective primary contexts.

Family / scale	n_{obj}	Algorithm	Default	SMAC3	irace	ConfigAgent
TSP / n500	3	NSGA-II	$3.286{\cdot}10^{11}$	$3.386{\cdot}10^{11}$	$\mathbf{3.389{\cdot}10^{11}}$	$\mathbf{3.394{\cdot}10^{11}}$
	3	NSGA-III	$3.252{\cdot}10^{11}$	$3.291{\cdot}10^{11}$	$\mathbf{3.321{\cdot}10^{11}}$	$\mathbf{3.328{\cdot}10^{11}}$
	3	MOEA/D	$3.450{\cdot}10^{11}$	$3.903{\cdot}10^{11}$	$3.857{\cdot}10^{11}$	$\mathbf{3.911{\cdot}10^{11}}$
	5	NSGA-II	$9.761{\cdot}10^{15}$	$1.012{\cdot}10^{16}$	$\mathbf{1.110{\cdot}10^{16}}$	$\mathbf{1.112{\cdot}10^{16}}$
	5	NSGA-III	$9.110{\cdot}10^{15}$	$9.678{\cdot}10^{15}$	$\mathbf{1.064{\cdot}10^{16}}$	$\mathbf{1.066{\cdot}10^{16}}$
	5	MOEA/D	$9.311{\cdot}10^{15}$	$1.311{\cdot}10^{16}$	$\mathbf{1.343{\cdot}10^{16}}$	$\mathbf{1.346{\cdot}10^{16}}$
Knapsack / n500	3	NSGA-II	$7.972{\cdot}10^{13}$	$\mathbf{7.999{\cdot}10^{13}}$	$7.992{\cdot}10^{13}$	$\mathbf{8.014{\cdot}10^{13}}$
	3	NSGA-III	$\mathbf{7.932{\cdot}10^{13}}$	$7.922{\cdot}10^{13}$	$7.912{\cdot}10^{13}$	$\mathbf{7.934{\cdot}10^{13}}$
	3	MOEA/D	$8.411{\cdot}10^{13}$	$8.761{\cdot}10^{13}$	$\mathbf{8.806{\cdot}10^{13}}$	$\mathbf{8.829{\cdot}10^{13}}$
	5	NSGA-II	$1.412{\cdot}10^{23}$	$1.417{\cdot}10^{23}$	$\mathbf{1.423{\cdot}10^{23}}$	$\mathbf{1.426{\cdot}10^{23}}$
	5	NSGA-III	$1.425{\cdot}10^{23}$	$1.425{\cdot}10^{23}$	$\mathbf{1.428{\cdot}10^{23}}$	$\mathbf{1.431{\cdot}10^{23}}$
	5	MOEA/D	$1.560{\cdot}10^{23}$	$1.687{\cdot}10^{23}$	$1.671{\cdot}10^{23}$	$\mathbf{1.692{\cdot}10^{23}}$
BPP / n500	3	NSGA-II	$1.092{\cdot}10^{8}$	$1.092{\cdot}10^{8}$	$\mathbf{1.093{\cdot}10^{8}}$	$\mathbf{1.093{\cdot}10^{8}}$
	3	NSGA-III	$1.093{\cdot}10^{8}$	$\mathbf{1.093{\cdot}10^{8}}$	$\mathbf{1.093{\cdot}10^{8}}$	$\mathbf{1.093{\cdot}10^{8}}$
	3	MOEA/D	$1.092{\cdot}10^{8}$	$\mathbf{1.093{\cdot}10^{8}}$	$\mathbf{1.093{\cdot}10^{8}}$	$\mathbf{1.093{\cdot}10^{8}}$
	5	NSGA-II	$1.626{\cdot}10^{13}$	$1.629{\cdot}10^{13}$	$1.628{\cdot}10^{13}$	$\mathbf{1.630{\cdot}10^{13}}$
	5	NSGA-III	$1.628{\cdot}10^{13}$	$\mathbf{1.630{\cdot}10^{13}}$	$1.627{\cdot}10^{13}$	$\mathbf{1.631{\cdot}10^{13}}$
	5	MOEA/D	$1.625{\cdot}10^{13}$	$1.626{\cdot}10^{13}$	$1.625{\cdot}10^{13}$	$\mathbf{1.628{\cdot}10^{13}}$
Scheduling / j10m5	3	NSGA-II	$2.078{\cdot}10^{9}$	$2.078{\cdot}10^{9}$	$2.077{\cdot}10^{9}$	$\mathbf{2.080{\cdot}10^{9}}$
	3	NSGA-III	$2.073{\cdot}10^{9}$	$2.073{\cdot}10^{9}$	$2.072{\cdot}10^{9}$	$\mathbf{2.074{\cdot}10^{9}}$
	3	MOEA/D	$2.054{\cdot}10^{9}$	$2.058{\cdot}10^{9}$	$2.056{\cdot}10^{9}$	$\mathbf{2.061{\cdot}10^{9}}$
	5	NSGA-II	$3.706{\cdot}10^{15}$	$3.706{\cdot}10^{15}$	$3.703{\cdot}10^{15}$	$\mathbf{3.710{\cdot}10^{15}}$
	5	NSGA-III	$3.688{\cdot}10^{15}$	$3.690{\cdot}10^{15}$	$3.683{\cdot}10^{15}$	$\mathbf{3.695{\cdot}10^{15}}$
	5	MOEA/D	$3.674{\cdot}10^{15}$	$3.670{\cdot}10^{15}$	$\mathbf{3.678{\cdot}10^{15}}$	$\mathbf{3.687{\cdot}10^{15}}$
Routing / cvrp200	3	NSGA-II	$7.949{\cdot}10^{6}$	$8.005{\cdot}10^{6}$	$7.984{\cdot}10^{6}$	$\mathbf{8.012{\cdot}10^{6}}$
	3	NSGA-III	$8.058{\cdot}10^{6}$	$8.101{\cdot}10^{6}$	$\mathbf{8.157{\cdot}10^{6}}$	$\mathbf{8.176{\cdot}10^{6}}$
	3	MOEA/D	$8.292{\cdot}10^{6}$	$8.747{\cdot}10^{6}$	$\mathbf{8.854{\cdot}10^{6}}$	$\mathbf{8.872{\cdot}10^{6}}$
	5	NSGA-II	$2.147{\cdot}10^{12}$	$2.149{\cdot}10^{12}$	$\mathbf{2.155{\cdot}10^{12}}$	$\mathbf{2.158{\cdot}10^{12}}$
	5	NSGA-III	$2.189{\cdot}10^{12}$	$2.206{\cdot}10^{12}$	$\mathbf{2.211{\cdot}10^{12}}$	$\mathbf{2.215{\cdot}10^{12}}$
	5	MOEA/D	$2.230{\cdot}10^{12}$	$2.385{\cdot}10^{12}$	$\mathbf{2.391{\cdot}10^{12}}$	$\mathbf{2.394{\cdot}10^{12}}$

The shared 2-objective KB's top-ranked configuration per family is deployed directly on 3- and 5-objective targets. Zero-shot deployment is competitive with the retuned ConfigAgent results in Table 24 on most reported contexts, and ten target trials reduce the remaining gap. Table 24 gives the per-context results for each tuner.

C.2 Scale transfer, cross-scale deployment, and leave-one-family-out

The scale-transfer test deploys the primary-scale-trained KB at non-primary scales without retuning. At zero shot, this is a deterministic retrieval-and-ranking procedure rather than an LLM call on the target. It can select a different source configuration for each target; the direct-reuse control in Table 37 redeploys one tuned configuration.

The training scale is the body primary context of Table 2. The OOD test set is ten scales sampled uniformly at random from the family's full scale range with ten instances per sampled scale, yielding one hundred OOD evaluations per family per seed. The sampling ranges are $n \in [100, 1000]$ for TSP and Knapsack, $n \in [200, 1000]$ for BPP, jobs $\times$ machines in $\{5, 7, 10, 15, 20, 25\} \times 5$ for Scheduling, and customers $\in [100, 500]$ for Routing. The primary scale is not excluded from the sampling range, so roughly one in ten sampled scales coincides with the primary scale and the remaining nine are unseen at training time.

A harder scale-transfer stressor is the mixed scale TSP variant of Table 27, where the sampling range is identical but the knowledge base sees no fixed primary scale and taxonomy retrieval must choose between the fixed primary-scale source bucket and the dedicated mixed-scale source bucket at deploy time. Two scale-transfer rows are populated with measured ConfigAgent numbers in

Table 30, namely Scheduling j10m5 deployed at j25m5 and Routing cvrp200 deployed at cvrp500. ConfigAgent zero-shot reaches the SMAC3 retune quality on a variant whose scale lies outside the family primary scale range by a factor of $2.5\times$.

Leave-one-family-out. We train the agent on four families and deploy on the held-out fifth with no same-family history. All three ConfigAgent variants commit to a single deployed configuration, picked by ranking same-config buckets first by mean quality, then by support count, and then by smaller variance, so the transfer comparison is cost matched against SMAC3's single configuration deploy on the same family.

Zero-shot retrieves the single top taxonomy-matched cross-family configuration and evaluates it on every test instance with zero new solver calls on the target. *Few-shot* runs $T=10$ adaptation trials on the target restricted to cross-family retrieval, then deploys the picker selected configuration from that adaptation pool, with no new insights written. *Full* is the picker selected same-family configuration from the shared KB and matches the body protocol of Table 2.

Across the five reported contexts, cross-family transfer reaches the corresponding in-family quality range without target-family trials in the KB, supporting the taxonomy labels used for retrieval.

C.3 Variant transfer, structural variants, and real-world benchmarks

This subsection deploys the frozen synthetic KB on three groups of OOD targets, kept separate so the structural-variant claim and the public-benchmark claim do not bleed into each other. The first group is five single objective structural variants, namely ATSP for TSP, multidimensional Knapsack for KP, Mixed-size BPP for BPP, Heterogeneous FJSP for Scheduling, and Clustered CVRP for Routing, each evaluated under the family's dedicated SO solver. The second group is four multi-objective sibling distortions, namely ATSP, Mixed scale TSP, Clustered TSP, and MDK, each evaluated under NSGA-II, NSGA-III, and MOEA/D at 2, 3, and 5 objectives. The third group is five canonical public benchmark libraries, namely TSPLIB95 (Reinelt, 1991), Chu–Beasley MKP (Chu & Beasley, 1998), Beasley OR-Library BPP (Beasley, 1990), Brandimarte FJSP (Brandimarte, 1993), and Solomon VRPTW (Solomon, 1987). Across all three groups the shared KB is deployed by taxonomy overlap with no tuning on the target, while the classical baselines retune from scratch per target at the body protocol budgets SMAC3 $T=200$ and irace $T=800$.

The KB was trained only on synthetic base contexts. Each structural variant shares a solver and encoding with one base family, while the instance distribution changes. The public-benchmark rows test the same retrieval rule on standard published datasets. Together, these tables show that one taxonomy-indexed KB can be reused beyond its synthetic training contexts.

C.3.1 Single-objective structural variants

Table 25 reports the five single-objective variants with the test-set scope, the family-dedicated solver, and the quality metric. The complete stage-timing evaluation is reported separately in Table 34; its times are not mixed into this quality summary.

Table 25: Single-objective structural variant.

				Quality			
Variant (Family)	**Scale**	**Solver**	**Metric**	**Default**	**SMAC3**	**irace**	**ConfigAgent**
ATSP (TSP)	n-mixed	LKH-3	tour (↓)	1,721.4	1,719.8	1,720.7	**1,719.2**
MDK (KP)	n-mixed	OR-Tools	value (↑)	91.39	91.46	91.42	**91.49**
Mixed-size BPP	n500	Gurobi	bins (↓)	276.0	248.8	248.8	**248.6**
Heterogeneous FJSP	j10m5	Gurobi	makespan (↓)	210.4	209.7	209.9	**209.6**
Clustered CVRP	n200	OR-Tools	dist (↓)	3,202.0	3,199.2	3,201.3	**3,198.7**

ATSP in the TSP family. The distance matrix becomes asymmetric while solver class, solution type, and family class stay matched. The taxonomy retrieves the symmetric TSP LKH-3 bucket. The agent inherits the candidate set and move type defaults of the source.

MDK in the Knapsack family. Several capacity rows are added to the single capacity base while bit string encoding and family class stay matched. The taxonomy retrieves the synthetic Knapsack

OR-Tools bucket. The added capacity rows are handled by the OR-Tools knapsack backend without changing the agent parameter space.

Mixed size BPP in the BPP family. The item size distribution becomes a mixture of small and large items while the first fit decreasing decoder encoding and family class stay matched. The taxonomy retrieves the synthetic BPP Gurobi bucket. The agent inherits the first fit decreasing plateau insight. Gurobi solves the assignment MIP under the same per instance budget as in the synthetic base.

Heterogeneous FJSP in the Scheduling family. Machine processing speeds become non uniform while operation permutation encoding and family class stay matched. The taxonomy retrieves the synthetic FJSP Gurobi bucket. The agent inherits the precedence preserving crossover defaults. The heterogeneous speeds change only the per operation cost matrix that Gurobi reads.

Clustered CVRP in the Routing family. The customer distribution becomes Gaussian clusters while giant tour encoding with depot separators and family class stay matched. The taxonomy retrieves the synthetic CVRP OR-Tools bucket. The agent inherits the routing first solution and metaheuristic defaults. OR-Tools handles the clustered geometry without changing the agent parameter space.

Across all five rows the quality summary shows ConfigAgent matching or improving the better classical baseline. The stage-level timing evaluation is reported separately in Table 34.

C.3.2 MULTI-OBJECTIVE STRUCTURAL VARIANTS

All four MO structural-variant tables run the same Default, SMAC3 at $T{=}200$, irace at $T{=}800$, and ConfigAgent zero-shot pipeline, with one orienting sentence per variant before the table.

ATSP asymmetric (TSP family). The distance matrix becomes asymmetric while solver class evolutionary, solution type permutation, and family class TSP stay matched, so the taxonomy retrieves the symmetric TSP NSGA-II bucket and the agent inherits the source mutation default.

Table 26: ATSP asymmetric (TSP), variant generalization across NSGA-II, NSGA-III, MOEA/D at 2, 3, and 5 objectives.

Algo	n_{obj}	Default	SMAC3	irace	ConfigAgent
NSGA-II	2	$2.596{\cdot}10^{7}$	$2.940{\cdot}10^{7}$	$2.935{\cdot}10^{7}$	$\mathbf{2.946{\cdot}10^{7}}$
	3	$2.775{\cdot}10^{10}$	$3.266{\cdot}10^{10}$	$3.050{\cdot}10^{10}$	$\mathbf{3.282{\cdot}10^{10}}$
	5	$5.476{\cdot}10^{14}$	$7.251{\cdot}10^{14}$	$6.859{\cdot}10^{14}$	$\mathbf{7.287{\cdot}10^{14}}$
NSGA-III	2	$2.610{\cdot}10^{7}$	$3.130{\cdot}10^{7}$	$3.127{\cdot}10^{7}$	$\mathbf{3.145{\cdot}10^{7}}$
	3	$2.830{\cdot}10^{10}$	$3.242{\cdot}10^{10}$	$3.355{\cdot}10^{10}$	$\mathbf{3.372{\cdot}10^{10}}$
	5	$6.651{\cdot}10^{14}$	$1.096{\cdot}10^{15}$	$9.147{\cdot}10^{14}$	$\mathbf{1.101{\cdot}10^{15}}$
MOEA/D	2	$3.498{\cdot}10^{7}$	$4.803{\cdot}10^{7}$	$4.650{\cdot}10^{7}$	$\mathbf{4.827{\cdot}10^{7}}$
	3	$4.065{\cdot}10^{10}$	$5.702{\cdot}10^{10}$	$5.394{\cdot}10^{10}$	$\mathbf{5.730{\cdot}10^{10}}$
	5	$1.781{\cdot}10^{15}$	$2.534{\cdot}10^{15}$	$2.439{\cdot}10^{15}$	$\mathbf{2.547{\cdot}10^{15}}$

Mixed scale TSP. The deployment scale is drawn uniformly from $n \in [100, 1000]$ rather than fixed at the primary scale. The KB has been trained on a dedicated mixed scale TSP context where each training trial draws a random n from the same range, so taxonomy retrieval at deploy time picks the mixed scale source bucket alongside the fixed primary scale bucket.

Table 27: Mixed scale TSP, variant generalization across NSGA-II, NSGA-III, MOEA/D at 2, 3, and 5 objectives.

Algo	n_{obj}	Default	SMAC3	irace	ConfigAgent
NSGA-II	2	$2.540{\cdot}10^{10}$	$2.614{\cdot}10^{10}$	$2.631{\cdot}10^{10}$	$\mathbf{2.644{\cdot}10^{10}}$
	3	$3.740{\cdot}10^{14}$	$3.871{\cdot}10^{14}$	$3.882{\cdot}10^{14}$	$\mathbf{3.896{\cdot}10^{14}}$
	5	$1.047{\cdot}10^{21}$	$1.054{\cdot}10^{21}$	$1.160{\cdot}10^{21}$	$\mathbf{1.166{\cdot}10^{21}}$
NSGA-III	2	$2.473{\cdot}10^{10}$	$2.545{\cdot}10^{10}$	$2.544{\cdot}10^{10}$	$\mathbf{2.557{\cdot}10^{10}}$
	3	$3.685{\cdot}10^{14}$	$3.690{\cdot}10^{14}$	$3.776{\cdot}10^{14}$	$\mathbf{3.795{\cdot}10^{14}}$
	5	$9.717{\cdot}10^{20}$	$1.027{\cdot}10^{21}$	$1.144{\cdot}10^{21}$	$\mathbf{1.150{\cdot}10^{21}}$
MOEA/D	2	$2.496{\cdot}10^{10}$	$2.725{\cdot}10^{10}$	$2.820{\cdot}10^{10}$	$\mathbf{2.835{\cdot}10^{10}}$
	3	$3.870{\cdot}10^{14}$	$4.142{\cdot}10^{14}$	$4.233{\cdot}10^{14}$	$\mathbf{4.255{\cdot}10^{14}}$
	5	$1.010{\cdot}10^{21}$	$1.363{\cdot}10^{21}$	$1.447{\cdot}10^{21}$	$\mathbf{1.454{\cdot}10^{21}}$

Clustered TSP. The instance distribution shifts from uniform Euclidean cities to a clustered point process while solver, encoding, and family class stay the same. The agent inherits the source mutation default but must rebalance crossover and mutation against the clustered geometry.

Table 28: Clustered TSP, variant generalization across NSGA-II, NSGA-III, MOEA/D at 2, 3, and 5 objectives.

Algo	n_{obj}	Default	SMAC3	irace	ConfigAgent
NSGA-II	2	$9.064{\cdot}10^{9}$	$9.121{\cdot}10^{9}$	$9.478{\cdot}10^{9}$	$\mathbf{9.819{\cdot}10^{9}}$
	3	$9.501{\cdot}10^{13}$	$9.911{\cdot}10^{13}$	$9.962{\cdot}10^{13}$	$\mathbf{1.001{\cdot}10^{14}}$
	5	$1.282{\cdot}10^{20}$	$1.453{\cdot}10^{20}$	$1.454{\cdot}10^{20}$	$\mathbf{1.461{\cdot}10^{20}}$
NSGA-III	2	$8.799{\cdot}10^{9}$	$8.967{\cdot}10^{9}$	$9.106{\cdot}10^{9}$	$\mathbf{9.151{\cdot}10^{9}}$
	3	$9.360{\cdot}10^{13}$	$9.654{\cdot}10^{13}$	$9.740{\cdot}10^{13}$	$\mathbf{9.788{\cdot}10^{13}}$
	5	$1.221{\cdot}10^{20}$	$1.248{\cdot}10^{20}$	$1.456{\cdot}10^{20}$	$\mathbf{1.463{\cdot}10^{20}}$
MOEA/D	2	$8.941{\cdot}10^{9}$	$9.856{\cdot}10^{9}$	$1.047{\cdot}10^{10}$	$\mathbf{1.052{\cdot}10^{10}}$
	3	$1.007{\cdot}10^{14}$	$1.111{\cdot}10^{14}$	$1.140{\cdot}10^{14}$	$\mathbf{1.146{\cdot}10^{14}}$
	5	$1.367{\cdot}10^{20}$	$2.007{\cdot}10^{20}$	$1.919{\cdot}10^{20}$	$\mathbf{2.017{\cdot}10^{20}}$

MDK multidimensional Knapsack. The constraint count moves from one to several while the bit string encoding and Knapsack family class remain. Retrieval finds the synthetic single dim Knapsack NSGA-II bucket and transports its bit flip mutation band.

Table 29: MDK multidimensional Knapsack, variant generalization across NSGA-II, NSGA-III, MOEA/D at 2, 3, and 5 objectives.

Algo	n_{obj}	Default	SMAC3	irace	ConfigAgent
NSGA-II	2	$1.790{\cdot}10^{10}$	$1.802{\cdot}10^{10}$	$1.805{\cdot}10^{10}$	$\mathbf{1.806{\cdot}10^{10}}$
	3	$2.493{\cdot}10^{16}$	$2.530{\cdot}10^{16}$	$2.503{\cdot}10^{16}$	$\mathbf{2.543{\cdot}10^{16}}$
	5	$6.144{\cdot}10^{28}$	$6.376{\cdot}10^{28}$	$6.350{\cdot}10^{28}$	$\mathbf{6.408{\cdot}10^{28}}$
NSGA-III	2	$1.785{\cdot}10^{10}$	$1.793{\cdot}10^{10}$	$1.774{\cdot}10^{10}$	$\mathbf{1.802{\cdot}10^{10}}$
	3	$2.465{\cdot}10^{16}$	$2.470{\cdot}10^{16}$	$2.487{\cdot}10^{16}$	$\mathbf{2.499{\cdot}10^{16}}$
	5	$6.102{\cdot}10^{28}$	$6.231{\cdot}10^{28}$	$6.040{\cdot}10^{28}$	$\mathbf{6.262{\cdot}10^{28}}$
MOEA/D	2	$1.833{\cdot}10^{10}$	$1.856{\cdot}10^{10}$	$1.859{\cdot}10^{10}$	$\mathbf{1.868{\cdot}10^{10}}$
	3	$2.659{\cdot}10^{16}$	$2.766{\cdot}10^{16}$	$2.743{\cdot}10^{16}$	$\mathbf{2.780{\cdot}10^{16}}$
	5	$7.217{\cdot}10^{28}$	$7.982{\cdot}10^{28}$	$7.869{\cdot}10^{28}$	$\mathbf{8.022{\cdot}10^{28}}$

We now move from the variant transfer to the scale transfer. Table 30 reports, for each family, the deploy of the primary-scale-trained KB at one held-out non-primary scale at n_{obj}=2.

Each row deploys the family's primary-scale configuration at the OOD scale, with SMAC3 and irace retuning from scratch. Timing is reported separately in Table 35 for the TSP and Knapsack scale-shift runs.

Table 30: Scale OOD generalization with NSGA-II at two objectives.

Family	Anchor → OOD scale	Default	SMAC3	irace	CA zero shot	CA few shot
TSP	n=500 → n=1000	$4.225{\cdot}10^{8}$	$4.235{\cdot}10^{8}$	$4.232{\cdot}10^{8}$	$4.238{\cdot}10^{8}$	$\mathbf{4.244{\cdot}10^{8}}$
Knapsack	n=500 → n=1000	$7.210{\cdot}10^{9}$	$7.185{\cdot}10^{9}$	$7.209{\cdot}10^{9}$	$7.215{\cdot}10^{9}$	$\mathbf{7.218{\cdot}10^{9}}$
BPP	n=500 → n=1000	$4.087{\cdot}10^{7}$	$4.087{\cdot}10^{7}$	$4.085{\cdot}10^{7}$	$4.090{\cdot}10^{7}$	$\mathbf{4.094{\cdot}10^{7}}$
Scheduling	$j10{\times}m5$ → $j25{\times}m5$	$1.278{\cdot}10^{7}$	$1.279{\cdot}10^{7}$	$1.277{\cdot}10^{7}$	$1.280{\cdot}10^{7}$	$\mathbf{1.284{\cdot}10^{7}}$
Routing	$cvrp200$ → $cvrp500$	$2.788{\cdot}10^{5}$	$2.815{\cdot}10^{5}$	$2.812{\cdot}10^{5}$	$2.818{\cdot}10^{5}$	$\mathbf{2.823{\cdot}10^{5}}$

The scale transfer is further stressed by the Mixed-scale TSP variant of Table 27, where the scale is drawn from $n \in [100, 1000]$ at deployment so the KB must choose among multiple source scale buckets. Table 31 crosses the same OOD scale with 3 and 5 objectives to check that the scale shift does not interact with the objective-count transfer on the four families measured.

Table 31: Scale OOD generalization crossed with higher objective counts, NSGA-II at the OOD scale.

Family	OOD scale	n_{obj}	Default	SMAC3	irace	ConfigAgent
TSP	n=1000	3	$6.305{\cdot}10^{11}$	$6.346{\cdot}10^{11}$	$6.485{\cdot}10^{11}$	$\mathbf{6.518{\cdot}10^{11}}$
TSP	n=1000	5	$1.414{\cdot}10^{16}$	$1.514{\cdot}10^{16}$	$1.488{\cdot}10^{16}$	$\mathbf{1.522{\cdot}10^{16}}$
Knapsack	n=1000	3	$6.020{\cdot}10^{14}$	$6.052{\cdot}10^{14}$	$6.007{\cdot}10^{14}$	$\mathbf{6.083{\cdot}10^{14}}$
Knapsack	n=1000	5	$4.184{\cdot}10^{24}$	$4.209{\cdot}10^{24}$	$4.146{\cdot}10^{24}$	$\mathbf{4.230{\cdot}10^{24}}$
Routing	$cvrp$=500	3	$1.393{\cdot}10^{8}$	$1.403{\cdot}10^{8}$	$1.398{\cdot}10^{8}$	$\mathbf{1.410{\cdot}10^{8}}$
Routing	$cvrp$=500	5	$2.539{\cdot}10^{14}$	$2.545{\cdot}10^{14}$	$2.548{\cdot}10^{14}$	$\mathbf{2.561{\cdot}10^{14}}$
BPP	n=1000	3	$2.182{\cdot}10^{8}$	$2.183{\cdot}10^{8}$	$2.182{\cdot}10^{8}$	$\mathbf{2.184{\cdot}10^{8}}$
BPP	n=1000	5	$6.486{\cdot}10^{13}$	$6.501{\cdot}10^{13}$	$6.499{\cdot}10^{13}$	$\mathbf{6.534{\cdot}10^{13}}$

Leave-one-family-out transfer. Table 32 reports a stricter cross-context test in which the agent trains a shared KB on four families and deploys on the held-out fifth without seeing a trial from that family.

Each row shows the held-out family's HV under the cross-family retrieved configuration. The few-shot column uses ten target-side trials, and the in-family column is the rounded comparator from the main result table.

Table 32: Leave-one-family-out transfer at the NSGA-II primary context, reported as mean ± standard deviation.

	ConfigAgent		
Held-out family	zero shot	few shot	In-family ConfigAgent
TSP n500	$(2.2545{\pm}0.1210){\cdot}10^{8}$	$(2.3368{\pm}0.1318){\cdot}10^{8}$	$2.339{\cdot}10^{8}$
Knapsack n500	$(1.8436{\pm}0.0644){\cdot}10^{9}$	$(1.8794{\pm}0.0628){\cdot}10^{9}$	$1.883{\cdot}10^{9}$
BPP n500	$4729.70{\pm}883.55$	$4729.02{\pm}881.25$	$4.732{\cdot}10^{3}$
Scheduling j10m5	$(2.0341{\pm}0.3593){\cdot}10^{6}$	$(2.0410{\pm}0.3609){\cdot}10^{6}$	$2.044{\cdot}10^{6}$
Routing cvrp200	$41764.30{\pm}2526.58$	$41764.30{\pm}2526.58$	$4.158{\cdot}10^{4}$

The reported means show that few-shot adaptation improves TSP, Knapsack, and Scheduling, is unchanged on Routing, and is slightly lower on BPP.

Table 33: Directly recorded ConfigAgent wall clock on the five NSGA-II two-objective structural variants. The few-shot run uses $T{=}10$, five training instances per trial, and a final evaluation on 50 test instances with three seeds.

Variant	Zero-shot eval.	$T{=}10$ adaptation	Final test	Few-shot total	Test calls
ATSP	68.5	356.7	73.9	430.6	150
MDK, two caps	40.0	332.5	44.7	377.2	150
Mixed-size BPP	597.5	578.9	756.8	1,335.7	150
Heterogeneous FJSP	964.7	651.4	1,035.5	1,686.9	150
Clustered CVRP	83.8	356.3	133.2	489.5	150

The timing protocol measures adaptation, final testing, and their sum separately. The few-shot total is adaptation plus its own final test, while the zero-shot evaluation is reported as a separate run.

Table 34 separately reports the complete single-objective variant-stage runs. Each tuner stage includes its own configuration procedure and final test, whereas Default includes only final testing. Keeping this table separate avoids comparing its $T{=}100$ ConfigAgent training run with the $T{=}10$ few-shot adaptation above as if they were the same experiment.

Table 34: Directly recorded stage wall clock for the complete single-objective structural-variant runs. ConfigAgent uses $T{=}100$; SMAC3 uses $T{=}200$; irace uses at most 800 experiments.

Variant	Solver	Default	SMAC3	irace	ConfigAgent	Test calls
ATSP	LKH-3	597.8	2,752.4	550.2	1,730.5	150
MDK, two caps	OR-Tools	0.154	0.881	0.415	564.1	147
Mixed-size BPP	Gurobi	6,501.2	35,935.3	4,739.1	5,991.0	150
Heterogeneous FJSP	Gurobi	32.6	162.8	36.4	676.7	150
Clustered CVRP	OR-Tools	4,506.2	6,880.4	4,505.5	5,184.7	147

Table 35: Measured scale-OOD wall clock in seconds for the classical baselines.

Family	Anchor → OOD scale	Default	SMAC3	irace
TSP	$n{=}500 \rightarrow n{=}1000$	2,286.4	3,366.3	2,274.8
Knapsack	$n{=}500 \rightarrow n{=}1000$	1,357.4	2,212.3	1,466.1

Transfer-aware numerical baselines under a shared parameterization. The following control restricts the comparison to three targets that share the NSGA-II (p_c, p_m) space. Every numerical transfer method receives the same 214 source configurations from TSP $n{=}100/200/500$ (respectively $67/67/80$ records), and every arm receives exactly ten target-side solver evaluations on the same target-training schedule. Warm-SMAC (Lindauer & Hutter, 2018) initializes from three source configurations; RGPE (Feurer et al., 2018) combines fixed source Gaussian processes with an updating target process; and the MI-SMBO-style arm (Feurer et al., 2015) uses three source-selected starts followed by target GP-EI. Held-out instances are not used for source selection or acquisition.

Table 36: Matched-budget numerical transfer on the shared NSGA-II (p_c, p_m) space. Values are held-out HV (↑); the gain is relative to the strongest numerical-transfer method in each row.

Target	Scratch	Warm-SMAC	RGPE	MI-SMBO	ConfigAgent	CA gain
ATSP	$1.803{\cdot}10^{7}$	$2.976{\cdot}10^{7}$	$2.876{\cdot}10^{7}$	$2.989{\cdot}10^{7}$	$\mathbf{3.070{\cdot}10^{7}}$	+2.70%
Mixed-scale TSP	$2.487{\cdot}10^{10}$	$2.543{\cdot}10^{10}$	$2.603{\cdot}10^{10}$	$2.241{\cdot}10^{10}$	$\mathbf{2.650{\cdot}10^{10}}$	+1.81%
Clustered TSP	$8.192{\cdot}10^{9}$	$9.410{\cdot}10^{9}$	$9.177{\cdot}10^{9}$	$9.106{\cdot}10^{9}$	$\mathbf{9.820{\cdot}10^{9}}$	+4.35%

The 12 numerical method–target runs each contain ten target observations, and each selected configuration has 100 valid held-out evaluations. Source experience helps: Warm-SMAC exceeds Scratch on

every target, while ConfigAgent is highest under the matched target budget. The matched numerical controls use the common NSGA-II parameter space.

Classical configuration reuse as a zero-target-cost baseline. A matched-cost comparison redeploys the SMAC3 configuration tuned on the base family onto the variant at zero target-side trials. Table 37 reports this baseline alongside ConfigAgent zero-shot and few-shot on the three TSP NSGA-II variants under the 2-objective protocol.

ConfigAgent zero-shot is higher than the reuse baseline on all three reported variants, consistent with selecting among multiple source contexts rather than redeploying one fixed configuration. ConfigAgent few-shot is then at or above the SMAC3 retune result.

Table 37: Configuration reuse versus ConfigAgent on TSP multi-objective variants. Reuse columns deploy the primary-scale tuned configuration on the variant; retune columns rerun the tuner from scratch on the variant.

	HV ↑						
		SMAC3		irace		ConfigAgent	
Variant	**Default**	**reuse**	**retune**	**reuse**	**retune**	**zero shot**	**few shot**
ATSP	$2.40{\cdot}10^{7}$	$2.54{\cdot}10^{7}$	$2.94{\cdot}10^{7}$	$2.51{\cdot}10^{7}$	$2.94{\cdot}10^{7}$	$2.60{\cdot}10^{7}$	$\mathbf{3.07{\cdot}10^{7}}$
Mixed scale TSP	$2.36{\cdot}10^{10}$	$2.52{\cdot}10^{10}$	$2.61{\cdot}10^{10}$	$2.50{\cdot}10^{10}$	$2.63{\cdot}10^{10}$	$2.54{\cdot}10^{10}$	$\mathbf{2.65{\cdot}10^{10}}$
Clustered TSP	$8.55{\cdot}10^{9}$	$9.04{\cdot}10^{9}$	$9.12{\cdot}10^{9}$	$9.00{\cdot}10^{9}$	$9.48{\cdot}10^{9}$	$9.06{\cdot}10^{9}$	$\mathbf{9.82{\cdot}10^{9}}$

This comparison shows the difference between reusing one tuned configuration and searching a taxonomy-indexed KB. ConfigAgent can select a different source bucket for each variant and records the evidence used for that choice. Zero-shot deployment uses deterministic retrieval and ranking; few-shot deployment adds ten LLM-guided target trials. On the three variants in Table 37, ConfigAgent few-shot is at or above the SMAC3 retune result.

C.4 Three-way transfer via KB reasoning

Each propose prompt includes the four evidence sources in §3.2: local history, a coverage map, parameter direction signals, and retrieved evidence. Table 38 shows the ten few-shot trials for the ATSP row. The agent retrieves the TSP NSGA-II bucket and the mutation-rate prior, then selects a configuration after repeated evaluations. The SMAC3 reuse baseline transfers only the tuned (p_c, p_m) pair; it does not use the KB or run new target trials. The same retrieval rule exposes the relevant stored evidence for the other structural and scale variants described above.

Table 38: Few-shot trace on ATSP NSGA-II at $T{=}10$ with the 32B KB.

Trial	p_c	p_m	**HV**	**Role**	**Retrieved cue**
1	0.70	0.020	$1.89{\cdot}10^{7}$	seed	TSP NSGA-II bucket default
2	0.70	0.005	$3.61{\cdot}10^{7}$	refine	lower p_m from TSP insight on per gene $1/n$
3	0.70	0.005	$2.37{\cdot}10^{7}$	verify	same config re-tested on a second instance
4	0.70	0.005	$2.85{\cdot}10^{7}$	verify	third instance, mean now stable
5	0.90	0.005	$3.58{\cdot}10^{7}$	explore	push p_c up toward TSP literature band
6	0.95	0.002	$3.13{\cdot}10^{7}$	explore	tighten p_m toward $1/n$ default
7	0.90	0.005	$2.63{\cdot}10^{7}$	verify	revisit trial 5 config
8	0.95	0.002	$3.66{\cdot}10^{7}$	best so far	p_m at $1/n$ default, verified replicate
9	0.95	0.002	$2.57{\cdot}10^{7}$	verify	noise band replicate
10	0.95	0.002	$3.84{\cdot}10^{7}$	deploy pick	verified top config selected for test evaluation

Canonical benchmark libraries. Tables 39–43 report the shared KB redeployed on five public benchmark libraries from the operations-research literature, namely TSPLIB95 (Reinelt, 1991) on LKH-3, Chu–Beasley MKP (Chu & Beasley, 1998) on Gurobi, the Beasley OR-Library bin packing

set (Beasley, 1990) on Gurobi, Brandimarte FJSP (Brandimarte, 1993) on Gurobi, and Solomon VRPTW (Solomon, 1987) on OR-Tools. Every row is a zero-shot deployment of the synthetic KB on the target, with no library-specific retune, and the Default column reports the published solver default evaluated under the same per-instance time budget. Each library is oriented by a single sentence below its table, naming the instance distribution, the per-instance budget, and the structural overlap through which the KB transports. The per-table rows are a representative cross-class subset for the larger libraries, with the Chu-Beasley MKP and Beasley BPP tables covering one row per series and the Solomon VRPTW table covering two short-horizon rows per C, R, and RC class plus C201 and R201 for the long-horizon C2 and R2 classes. The full-library means reported in body Table 4 aggregate over every instance in each library and therefore weight the easier classes proportionally to their share of the library.

TSPLIB95 supplies real-world symmetric Euclidean TSP instances at $n \in [724, 4{,}461]$ drawn from physical layouts such as PCB drilling and city geographies, evaluated under LKH-3 at $300\,\mathrm{s}$ per instance; the synthetic TSP KB transports through the shared symmetric-Euclidean distance and permutation encoding, so the agent's p_m insight on the synthetic n500 distribution applies directly to instance scales beyond the training range.

Table 39: TSPLIB95 symmetric, LKH-3 at $300\,\mathrm{s}$ per instance, tour length ↓.

		Objective						Configuring time (s) ↓		
Instance	n	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**Known**	**Gap**	**SMAC3**	**irace**	**ConfigAgent**
pcb1173	1,173	56,937	56,904	56,912	**56,892**	56,892	0.00%	21,785	8,278	305
u724	724	41,920	41,913	41,915	**41,910**	41,910	0.00%	34,225	13,006	260
pr1002	1,002	259,083	259,051	259,064	**259,045**	259,045	0.00%	15,847	6,019	295
rl1304	1,304	253,042	252,983	253,008	**252,948**	252,948	0.00%	70,562	26,824	308
vm1748	1,748	336,751	336,631	336,684	**336,556**	336,556	0.00%	29,461	11,197	312
rl1889	1,889	316,752	316,702	316,725	**316,684**	316,536	0.05%	24,376	9,268	310
d2103	2,103	80,518	80,486	80,503	**80,450**	80,450	0.00%	28,241	10,734	315
pr2392	2,392	378,211	378,162	378,194	**378,108**	378,032	0.02%	18,658	7,091	320
pcb3038	3,038	137,808	137,769	137,795	**137,725**	137,694	0.02%	28,426	10,802	330
fnl4461	4,461	182,628	182,591	182,608	**182,566**	182,566	0.00%	61,273	23,284	340

Chu–Beasley MKP supplies multi-dimensional 0/1 knapsack instances at $n \in \{100, 250, 500\}$ items with several capacity rows, evaluated under Gurobi at $60\,\mathrm{s}$ per instance; the synthetic single-capacity Knapsack KB transports through the shared bit-flip encoding, with the added capacity rows handled by Gurobi's native multi-constraint MIP backend.

Table 40: Chu–Beasley MKP, Gurobi at $60\,\mathrm{s}$ per instance, total value ↑.

		Objective						Configuring time (s) ↓		
Instance	n	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**Known**	**Gap**	**SMAC3**	**irace**	**ConfigAgent**
mknapcb1	100	2,410	2,415	2,412	**2,418**	2,418	0.00%	4,183	1,594	62
mknapcb2	100	2,851	2,857	2,854	**2,861**	2,861	0.00%	8,294	4,290	63
mknapcb3	100	3,385	3,392	3,388	**3,398**	3,398	0.00%	8,706	7,288	64
mknapcb4	100	3,121	3,126	3,123	**3,132**	3,132	0.00%	6,248	4,648	63
mknapcb5	250	5,889	5,901	5,893	**5,910**	5,910	0.00%	10,571	7,960	65
mknapcb6	250	6,187	6,198	6,189	**6,205**	6,205	0.00%	10,438	7,214	65
mknapcb7	250	6,248	6,258	6,252	**6,266**	6,266	0.00%	10,573	7,198	65
mknapcb8	250	6,327	6,336	6,329	**6,342**	6,342	0.00%	10,716	7,175	65
mknapcb9	500	11,728	11,740	11,731	**11,749**	11,749	0.00%	10,092	7,131	72

Beasley OR-Library bin packing supplies one-dimensional BPP instances at $n \in [60, 1{,}000]$ items, evaluated under Gurobi at $60\,\mathrm{s}$ per instance; the synthetic BPP KB transports through the shared first-fit-decreasing decoder, and Gurobi handles the assignment MIP.

Brandimarte FJSP supplies flexible job shop instances at shop sizes $10{\times}6$ to $20{\times}15$, evaluated under Gurobi at $60\,\mathrm{s}$ per instance; the synthetic FJSP KB transports through the shared operation-permutation encoding with precedence-preserving crossover, and the larger shop sizes stress the same p_m band the synthetic protocol calibrated on $j10m5$.

Table 41: Beasley BPP, Gurobi at 60 s per instance, bins used ↓.

		Objective						Configuring time (s) ↓		
Instance	n	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**Known**	**Gap**	**SMAC3**	**irace**	**ConfigAgent**
binpack1	120	49.3	49.1	49.2	**49.0**	49.0	0.00%	10,284	3,908	65
binpack2	250	57.4	57.2	57.3	**57.0**	57.0	0.00%	13,916	5,288	65
binpack3	500	98.7	98.5	98.6	**98.4**	98.0	0.41%	16,062	6,104	65
binpack4	250	75.1	75.0	75.0	**75.0**	75.0	0.00%	14,023	5,329	65
binpack5	60	15.3	15.1	15.2	**15.0**	15.0	0.00%	4,613	1,753	60
binpack6	500	134.3	134.0	134.1	**133.8**	133.4	0.30%	5,584	2,122	65
binpack7	249	118.1	117.9	118.0	**117.6**	117.2	0.34%	13,847	5,262	65
binpack8	1,000	206.7	206.3	206.5	**205.9**	205.4	0.24%	15,593	5,925	65

Table 42: Brandimarte FJSP, Gurobi at 60 s per instance, makespan ↓.

		Objective						Configuring time (s) ↓		
Instance	**Shape**	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**Known**	**Gap**	**SMAC3**	**irace**	**ConfigAgent**
mk01	10×6	42	41	41	**40**	40	0.00%	3,152	1,197	62
mk02	10×6	28	27	27	**26**	26	0.00%	12,627	4,798	62
mk03	15×8	208	206	207	**204**	204	0.00%	13,241	5,032	62
mk04	15×8	62	61	61	**60**	60	0.00%	12,415	4,718	62
mk05	15×4	174	173	173	**172**	172	0.00%	12,586	4,783	62
mk07	20×5	144	141	142	**139**	139	0.00%	12,918	4,909	62
mk08	20×10	528	526	527	**524**	523	0.19%	12,894	4,900	62
mk10	20×15	201	199	200	**197**	196	0.51%	14,338	5,448	62

Solomon VRPTW supplies CVRP instances with hard time-window constraints at $n = 100$ customers, evaluated under OR-Tools at 60 s per instance; the synthetic CVRP KB transports through the shared giant-tour with depot-separator encoding, and the time-window dimension is layered on top via the OR-Tools dimension API without changing the agent's parameter space.

Table 43: Solomon VRPTW, OR-Tools at 60 s per instance, total distance ↓.

		Objective						Configuring time (s) ↓		
Instance	n	**Default**	**SMAC3**	**irace**	**ConfigAgent**	**Known**	**Gap**	**SMAC3**	**irace**	**ConfigAgent**
C101	100	834.20	830.65	832.10	**829.12**	828.94	0.02%	11,482	4,363	65
C102	100	830.90	829.41	830.05	**828.07**	827.92	0.02%	11,235	4,269	65
R101	100	1,655.40	1,653.26	1,654.18	**1,651.20**	1,650.80	0.02%	11,655	4,429	65
R102	100	1,490.30	1,488.92	1,489.61	**1,486.67**	1,486.12	0.04%	8,505	3,232	65
RC101	100	1,701.15	1,699.42	1,700.30	**1,697.38**	1,696.94	0.03%	8,451	3,212	65
RC102	100	1,556.20	1,555.61	1,555.92	**1,554.91**	1,554.75	0.01%	11,586	4,403	65
C201	100	593.12	592.37	592.81	**591.72**	591.56	0.03%	11,822	4,492	65
R201	100	1,256.30	1,254.78	1,255.51	**1,253.42**	1,252.37	0.08%	9,049	3,438	65

Tables 39–43 show competitive zero-shot transfer and lower configuring time across the five public benchmark libraries.

D PART D: INTERPRETABILITY

Part D documents what can be inspected after a run. LLM-generated proposals include a typed trace whose cited ids are checked against the evidence shown to the model. The zero-shot picker logs the source bucket and the statistics used for ranking. Section A.10 gives an end-to-end trace example. Section D.1 reports one-component-removed ablations, Section D.2 reports KB statistics, Section D.3 tests the safeguards around stored text and aggregation, and Section D.4 gives examples of stored insights. Together, these records make the decision process inspectable.

D.1 Ablations, components and plain LLM baseline

Where the preceding appendices test where the frozen KB transfers, this subsection examines the contribution of individual components. For the *no reflection* and *no trace validator* rows, Table 6 reruns the $T{=}100$ training with one component removed. The *no retrieval at deployment* row keeps the independently trained source records but disables cross-context retrieval when choosing the deployed configuration. The comparison measures the effect of each removal under the stated protocol.

The two LLM-AC baselines at the bottom of the table treat the problem as pure in-context optimization in the OPRO style (Yang et al., 2024) and as discriminative surrogate Bayesian optimization in the LLAMBO style, and both consume the same $T{=}100$ budget but have no retrieval, reflection, or trace validation. *Full ConfigAgent* in the first row is the deployed configuration selected by the same deploy picker used in Table 2, so all ablation rows are directly comparable to the body headline.

Across the five primary contexts, Full ConfigAgent is the top entry in every column, and both LLM-AC baselines are below the partial ablations. The largest observed gap follows removal of the trace validator on TSP and Knapsack and removal of deployment retrieval on Routing.

OPRO and LLAMBO are run independently on each context and are reported at $T{=}100$ on the five primary contexts as LLM-based AC references.

D.2 Agent behavior and knowledge-base statistics

We use the 14B sweep as the diagnostic run for KB instrumentation. This subsection reports the per-context dynamics underlying the generalization results: how many trials the agent spends, how many unique configurations it visits, how many insights it authors, what the completed source knowledge base contains, and which records are eligible for retrieval at deployment.

Over the $T{=}100$ trials in each source context, the agent explores using its local history, coverage map, and parameter direction signals; cross-context retrieval is disabled during this phase. After source contexts are committed to the KB, the scope filter determines which insights may enter a compatible target's few-shot evidence packet. Frozen zero-shot deployment instead ranks eligible measured trial buckets without reading insight text for the final selection. Table 44 reports the resulting KB statistics.

Table 44: Knowledge-base statistics on the 14B two-objective sweep.

Property	**Value**
Total trials in the 14B 2-obj sweep	1,500
Unique configurations per context	18 to 25
Mean trials per unique configuration	roughly 5
Long term insights authored across the sweep	28
Insights kept after per-context capacity cap	≤ 5 per context

The ratio of total trials to unique configurations shows the agent reuses configurations for verification rather than sampling exhaustively, which is the behavior the lex ordered deploy rule rewards since multi trial verified configurations outrank single trial outliers of similar mean. The per context capacity cap means accumulated evidence is not crowded out by one off claims. The semantic canonicalization of reflection text means paraphrases of the same directional claim collapse into a single entry whose support count tracks rederivations.

D.3 Insight safeguards and aggregation checks

Adversarial insight injection. LLM-generated insight text is not treated as a measured result. To test whether unsupported prose can override the frozen selector, we inserted 1,000 target-scoped text-only insights into each of 12 deployment anchor cells (12,000 entries in total). Every injected insight had artificial support 10^9, recommended an unsupported out-of-band setting, and left the trial records unchanged. The release evaluator returned the same selected configuration and the same $(\text{mean}, \text{support}, \text{standard deviation})$ trial bucket in all 12 cells before and after injection. This selector-invariance check shows that insight text cannot create a quality value, satisfy $n_{\min}{=}3$, or replace repeatedly measured evidence during frozen zero-shot deployment.

Population aggregation. Reflection observes a realized quality and tests a local hypothesis, but it cannot promote a configuration by itself. Table 45 compares selection from one calibration instance with selection by the mean over ten calibration instances, holding the compatible band-valid candidate set fixed. Both selected configurations are evaluated on the same disjoint set of 100 held-out instances with three seeds.

Table 45: Effect of calibration aggregation on held-out HV (↑).

Family	One-instance pick	10-instance pick	Relative change
TSP	$2.327{\cdot}10^{8}$	$2.327{\cdot}10^{8}$	0.000%
KP	$1.876{\cdot}10^{9}$	$1.880{\cdot}10^{9}$	$+0.217\%$
BPP	4,731.267	4,731.700	$+0.009\%$
FJSP	$2.041{\cdot}10^{6}$	$2.043{\cdot}10^{6}$	$+0.110\%$
CVRP	$4.173{\cdot}10^{4}$	$4.178{\cdot}10^{4}$	$+0.131\%$

All 3,000 held-out evaluations completed successfully. The ten-instance criterion matches or improves the displayed held-out mean in all five families, with paired tests significant for KP (p=0.00135) and FJSP (p=0.00412). As a second robustness check, mean, median, and 10%-trimmed-mean ranking were compared on five representative contexts (4,500 held-out evaluations). All three statistics selected the same TSP configuration, and their resulting held-out means differed by at most 0.155% across the five families, showing stable ranking across these aggregation choices.

D.4 Sample accumulated insights

Table 46 presents representative insights from the 14B knowledge base midway through a shared KB sweep. The store is a live accumulator: each reflection call may merge new evidence into an existing canonicalized claim, author a new claim, or contradict an existing one, with the support count tracking rederivations over time. Later training points typically contain more entries, different support counts, and sometimes opposite directional claims in contexts where the trajectory has not yet converged. Across training points, the store consistently accumulates mechanism claims about mutation and crossover balance, with the structured fields of topic, parameter, and scope populated on newer insights. The ranking key favours insights that carry structural-dimension matches with the target, so a context whose encoding differs from an insight's origin does not see that insight at the top of the block even when raw support is high.

The MO rows attach to evolutionary operator parameters and each claim is grounded in the family's own solution encoding. The SO rows attach to solver-specific categorical parameters that the MO rows never see. The retrieval layer chooses among rows by taxonomy match at propose time, so a target context sees only rows whose solver class and encoding align with it. The scope whitelist further keeps a single family's high-support insight from dominating the prompt on a target whose family does not match.

The four rows of Table 47 cover the four transport regimes the retrieval layer supports. The first row is a same encoding cross family subtype transport, from symmetric to asymmetric TSP, the canonical structural variant case. The second row is a same family cross solver transport, from NSGA-II to NSGA-III, where the algorithm specific knob H is layered on top of the transported p_m default. The third row is a same context cross objective transport, where the n_{obj} dimension shifts but the scope whitelist on the relevant insight keeps it valid across objective counts. The fourth row shows the selectivity of the retrieval layer. When solution type and family class both shift, the structural overlap drops below the retrieval threshold and the taxonomy filter withholds the source insight so the agent falls back to the nearest structurally matched source context. This selectivity is the mechanism that keeps each family's idiosyncratic mechanism inside its own retrieval pool.

Table 46: Representative insights from the 14B KB.

Family	Mode	Param	Claim
TSP	MO NSGA-II	p_m	Per gene p_m near the $1/n$ **default** preserves near optimal subsequences under Order Crossover, and raising p_m breaks several near optimal edges per individual.
TSP	SO LKH-3	move type, candidates	**Move type 5 with a candidate set of 5 to 10 edges** gives the best tour quality inside the budget, while smaller sets under explore and larger sets waste time on low return neighbours.
Knapsack	MO NSGA-II	p_m	Per gene bit flip p_m in the **0.005 to 0.03 band** explores Hamming-1 neighbours efficiently, while higher p_m pushes the population into infeasible capacity regions.
Knapsack	SO OR-Tools	solver type	**Multidimension branch and bound dominates dynamic programming once n exceeds 200 items** because the DP table grows with capacity.
BPP	MO NSGA-II	p_c, p_m	The **first fit decreasing decoder is partly self repairing**, so the crossover and mutation balance has a wide plateau with no measurable HV improvement from further p_m increases.
BPP	SO Gurobi	cuts, heuristics	**Cut parameter above one pays cut generation time without closing the gap faster** when the heuristic already reaches the solution first, so keep cuts at default.
Scheduling	MO NSGA-II	p_m	A high crossover probability with p_m in a moderate band preserves diversity on dual layer encodings with precedence constraints, and **above per gene p_m=0.08 chains start to break**.
Scheduling	SO Gurobi	mip focus	**MIP focus on feasibility first closes the makespan gap faster than the default** on flexible job shop under tight budgets.
Routing	MO NSGA-II	p_m	Per gene p_m in the **0.005 to 0.03 band on giant tour with depot separators** keeps route boundaries stable, and above 0.05 depots shift and route reassignment dominates.
Routing	SO OR-Tools	first solution, metaheuristic	**Path cheapest arc first solution paired with guided local search outperforms savings first solution** on moderate scale CVRP under the same budget.

Table 47: Insight reuse across contexts.

Source → Target	Transfer type	Key differential claim
tsp NSGA-II 2 obj → atsp NSGA-II 2 obj	**family subtype**	$1/n$ per gene p_m default transports; ATSP asymmetric matrix does not invalidate the permutation encoding rule.
tsp NSGA-II 2 obj → tsp NSGA-III 2 obj	**solver**	p_m claim carries across; NSGA-III adds reference direction saturation on top.
knap MOEA/D 2 obj → knap MOEA/D 3 obj	**objective count**	scope whitelist on Tchebycheff claim does not gate on n_{obj}, so zero shot reuse applies.
tsp NSGA-II 2 obj → mdk NSGA-II 2 obj	**encoding and family**	Encoding and family both differ, so the trial-store structural match drops to 1 of 3 and the source bucket is rejected. The insight-store scope check rejects the $1/n$ permutation-encoding insight because its $\mathcal{A}_e$={permutation} excludes binary MDK. Neither source record is rendered.

E PART E: STATIC EXTERNAL PRIOR ENTRIES

Part E lists the entries of the read-only static prior block rendered into the evidence packet at prompt assembly time. These entries are separate from the learned KB $\mathcal{K}=(\mathcal{T}, \mathcal{S})$ because they are specified

before the per-context training loop and are not updated by reflection or trial outcomes. §E.1 gives the solver mechanism rows and the initial effective bands used by the implementation, while §E.2 gives the problem-family, structural-variant, and benchmark-library rows used to scope insights by taxonomy. Source citations are included where a row follows a published algorithm or rule; measured trial records provide the evidence used for context-specific decisions.

E.1 SOLVER MECHANISM DESCRIPTIONS

Each solver entry in the static prior block contains the solver's decision rule, tunable parameters, initial effective bands, and failure modes specific to the family encoding. At propose time the renderer pulls the row keyed by the current solver and any active scope whitelist, so only matching rows are emitted into the prompt. The agent may contradict any prior when the current context's trials disagree, but the validator rejects citations whose id does not resolve to a rendered record. Table 48 lists the entries the prior block currently holds. Three rows cover the MOEA selection rules NSGA-II, NSGA-III, and MOEA/D. Five rows cover the family-specific GA recombination rules paired one to one with the five CO families. Four rows cover the single-objective backends LKH-3, OR-Tools Routing, OR-Tools Knapsack, and Gurobi.

Table 48: Solver entries rendered into the propose prompt.

Solver	Mechanism	Tunable parameters	Initial band and cautions
NSGA-II (Deb et al., 2002)	Dominance ranking with crowding distance in environmental selection.	p_c, p_m.	p_c is initialized in $[0.8, 1.0]$, with a family-dependent p_m. Serves as the reference for sibling multi-objective entries.
NSGA-III (Deb & Jain, 2014)	Reference-direction survivor selection with H directions per axis.	p_c, p_m, $H \in [4, 20]$.	The implementation initializes H in 10–14 for two objectives and 12–18 for three objectives with population 50.
MOEA/D (Zhang & Li, 2007)	Decomposition into N_{pop} scalar subproblems with neighbourhood size T and a scalarization rule.	p_c, p_m, $T \in [10, 40]$, scalarization in Tchebycheff, weighted sum, or PBI.	Tchebycheff is the default; the implementation initializes T in $0.15N_{\text{pop}}$ to $0.4N_{\text{pop}}$ on permutation families.
TSP GA	Permutation encoding with order crossover and two opt style mutation.	p_c, p_m.	p_m effective in $[0.0005, 0.005]$ per gene. The $1/n$ rule of Goldberg (1989); Mühlenbein (1992) gives $p_m \approx 0.002$ at $n=500$ and motivates this initial band.
Knapsack GA	Binary encoding with single point crossover and bit flip mutation.	p_c, p_m.	p_m effective in $[0.001, 0.03]$ per gene bit flip. The $1/n$ default for $n=500$ items is 0.002.
BPP GA	Permutation plus first fit decreasing decoder encoding.	p_c, p_m.	p_m effective in $[0.005, 0.05]$ per gene. Decoder absorbs moderate shuffle noise.
Scheduling GA	Operation permutation encoding with precedence preserving crossover.	p_c, p_m.	p_m effective in $[0.01, 0.08]$ per gene. Above 0.08 feasible precedence chains break.
Routing GA	Giant tour encoding with depot separator decoder.	p_c, p_m.	p_m effective in $[0.005, 0.03]$ per gene. Above 0.05 depot positions shift.
LKH-3	Iterative Lin Kernighan Helsgaun local search on a permutation, restarting from diverse tours.	Move type in $\{3, 4, 5\}$, max candidates in $[3, 15]$, fixed restart count.	Move type five is the default and dropping to three trades quality for runtime. Candidate set size five to ten is typical on Euclidean TSP.
OR-Tools Routing	Constraint programming with a local search metaheuristic over a first solution strategy.	First solution strategy, local search metaheuristic.	Guided local search with the path cheapest arc seed is the default. Savings or Christofides first solutions interact with the metaheuristic choice on capacitated variants.
OR-Tools Knapsack	Dynamic programming or branch and bound on binary inclusion.	Solver backend choice.	Multidimension branch and bound is the default. The DP backend is limited to small n with one capacity.
Gurobi	Mixed integer exact solve with heuristics, presolve, and cut plane controls.	Heuristics weight in $[0, 1]$, presolve level, cut aggressiveness, MIP focus mode.	Default presolve is automatic. MIP focus two biases toward feasibility and three toward bound. Cuts above one can slow easy instances.

Every row ships with a taxonomy key, and rows derived from published algorithms or rules retain their source citations. Once a context has accumulated its own trials, the agent can override the initial band with measured evidence from that context. The table is a prior, not a constraint.

Each pitfall and tuning heuristic in a spec carries an optional scope whitelist, a dict from taxonomy dimension to allowed values. At prompt assembly the renderer keeps a line only when every listed dimension matches the current context. This is how the permutation family mutation band override in MOEA/D's entry ships to a MOEA/D on TSP prompt but is filtered out of a MOEA/D on knapsack prompt, even though the solver spec record is the same object. Without a scope entry a line is always included, which is the backwards compatible default for specs authored before the schema existed.

Provenance of the prior bands and absence of target leakage. The search space passed to ConfigAgent is identical to the search space passed to SMAC3 and irace, namely the wide ranges in Tab. 12 such as $p_m \in [0.0, 0.3]$ for NSGA-II, so all tuners optimize over an identical configuration space on every context.

The initial bands reported per solver in this section are an additional structured prior, read both at propose and at deploy time. The agent receives the band as text alongside trial history and may freely propose configurations outside it, and the deploy picker uses the band as a robustness filter that downweights single-trial peaks outside the initial region. SMAC3 instead learns its preference over the shared search space from its surrogate and acquisition function; ConfigAgent makes its initial region explicit in the trace.

No bound, pitfall, or scope whitelist is derived from a target benchmark's test set or from any held-out instance distribution. These entries are fixed before target evaluation and are instance-independent.

This pattern of injecting structured prior knowledge into a tuner appears in several adjacent systems. LLAMBO (Liu et al., 2024) specifies hyperparameter ranges per task, ReEvo (Ye et al., 2024) injects algorithmic skeletons through white-box prompts, and irace-evo (Chacón Sartori & Blum, 2025) uses structured heuristic specifications. ConfigAgent keeps these initial entries separate from measured trial evidence and exposes both in the decision trace.

Mutation rate convention. The agent's p_m prior uses the per gene convention of DEAP (Fortin et al., 2012), namely a Bernoulli probability applied per gene at each mutation step, tied to the $1/n$ rule of Goldberg (1989); Mühlenbein (1992) as the lower band. The per gene convention is consistent with the runner and with the established GA literature for permutation and bit string encodings.

E.2 Problem family, variant, and benchmark library specs

The methodology mirrors the solver specs. The problem side of the agent's context is also KB-resident text, retrieved at propose time rather than compiled into code. For each context the agent sees a problem-context block assembled from three entry types. *Family* text fixes what the target problem is, namely its synthetic distribution, solution encoding, and objective list, so the agent does not have to infer a tour length, bit-flip feasibility score, or precedence-chain delay from the solver name alone. *Variant* text records which taxonomy dimensions a sibling distribution shares with its base family, so cross-context retrieval can transport trials, and which dimensions shift, so the agent knows which priors stop applying. *Benchmark library* text turns public instance sets into named objects, so an LKH zero-shot proposal on TSPLIB does not spend early trials against the easy-band ceiling. The mapping from variant or library to base family is simultaneously the taxonomy signal the deploy-time retrieval uses to route cross-context trials, so the same specs serve both the propose prompt and the deploy step. Table 49 summarizes the five problem families, the nine OOD variant and scale-shift settings, and the five benchmark libraries registered in the paper.

Without these rows the agent would have only the solver name and the numerical parameter ranges, and would have to guess the relationship between, say, TSPLIB ATSP and synthetic TSP or between Clustered TSP and base TSP. Making that relationship retrievable text removes the guessing step and makes every cross context transport decision auditable from the trace.

Table 49: Problem entries rendered into the propose prompt.

Entry	Scope or shift	Taxonomy retrieval note
Problem families		
TSP	Symmetric Euclidean distances, permutation encoding, objectives drawn from total tour distance, max edge, min edge, edge std, edge range.	Full transport across all TSP contexts sharing solver and encoding; scale and objective count are free.
Knapsack	Multi-dimensional 0/1 knapsack, bit-string encoding, objectives are the total value on each of up to five independent value dimensions.	Bit-flip semantics transport across scales; the active value-dimension count grows with the objective count.
BPP	One-dimensional bin packing, permutation-plus-FFD-decoder encoding, objectives drawn from bin count, max bin load, load variance, negated minimum bin utilization, wasted space.	Transports across scale; decoder choice is fixed family-wide.
Scheduling	Flexible job shop, operation-permutation encoding, objectives drawn from makespan, balanced workload, average flowtime, total workload, max flowtime.	Precedence constraint stays across contexts; shop dimension varies per scale.
Routing	Capacitated vehicle routing, giant tour with depot separators, objectives drawn from total distance, longest subroute, number of routes, max route load, route-length std.	Depot structure fixed; capacity and customer count vary per scale.
Structural variants		
Mixed scale TSP	Scale transfer, instances drawn from $n \in [100, 1000]$ within one deployment.	Base family TSP, cross context retrieval transports with reduced scale weight.
ATSP	Family transfer, asymmetric distance matrix.	Base family TSP, encoding compatible, but symmetric only heuristics relax, intended solver is LKH-3.
Clustered TSP	Distribution transfer within TSP, nodes sampled as Gaussian clusters rather than uniform.	Base family TSP, candidate set heuristics benefit from cluster aware tuning.
Multi dimensional knapsack	Dimension transfer on Knapsack, multiple capacity constraints.	Base family Knapsack, bit flip neighbourhood still applies, feasibility is tighter.
Mixed size BPP	Distribution transfer on BPP, item sizes drawn from a wider range than the primary.	Base family BPP, FFD decoder self repair absorbs the broader size spread.
Heterogeneous FJSP	Processing-time transfer on Scheduling, machine speeds become non-uniform.	Base family Scheduling, precedence preserving p_m band still applies under heterogeneous machine costs.
Clustered CVRP	Distribution transfer on Routing, customers sampled as Gaussian clusters rather than uniform.	Base family Routing, giant tour depot separator encoding stays matched, OR-Tools handles clustered geometry without changing the agent parameter space.
Scale OOD scheduling j25m5	Scale transfer on Scheduling, shop grows from primary $j10{\times}m5$ to $j25{\times}m5$.	Base family Scheduling, precedence preserving p_m band still applies at larger shop.
Scale OOD routing cvrp500	Scale transfer on Routing, customer count grows from primary 200 to 500.	Base family Routing, giant tour depot separator band transports across scale.
Benchmark libraries		
TSPLIB symmetric	Real world TSP instances, $n \in [724, 4{,}461]$, LKH-3 at 300 s per instance.	Hard instances separate tuners; easier sub-2K instances collapse because any reasonable LKH configuration reaches optimum.

Continued on next page

Table 49: Problem entries rendered into the propose prompt (continued).

Entry	Scope or shift	Taxonomy retrieval note
TSPLIB ATSP	Asymmetric TSP instances, 18-instance aggregate, LKH-3 at 120 s.	Targets the ATSP variant, and LKH-3 handles asymmetry natively.
Chu–Beasley MKP	Multi dimensional knapsack benchmarks, Gurobi at 60 s, total value objective.	Targets Multi dimensional knapsack variant.
Beasley OR-Library BPP	Bin packing benchmarks, Gurobi at 60 s, bin count objective.	Targets BPP family.
Brandimarte FJSP	Flexible job shop benchmarks, Gurobi at 60 s, makespan objective.	Targets Scheduling family.
Solomon VRPTW	Vehicle routing with time windows, OR-Tools at 60 s, total distance objective.	Targets Routing family, with the time window constraint relaxing the pure capacity view.

F PART F: LIMITATIONS

Three limitations of the framework are worth flagging for practitioners considering deployment.

Per trial LLM inference overhead. The agent pays a fixed LLM inference overhead of roughly 10 to 20 s per trial that is not shared across multiple uses within a single context, so the benefit shows up only when the same KB is reused across contexts. For one off tuning of a single context at a single budget, classical SMAC3 remains competitive on wall clock and the agent's shared KB argument does not apply. Practitioners with a single context to configure and no expected reuse should treat ConfigAgent as a quality matched alternative whose benefit is interpretability rather than wall clock.

Direction signal lacks recency weighting. The Spearman direction signal in the evidence packet weights all stored trials equally, so it tracks the agent's full search history rather than the recent best so far region. When the trajectory shifts late in training, for instance after a breakthrough configuration moves the best so far from one parameter region to another, the unweighted Spearman can take several trials to follow. A recency weighted variant of the Spearman signal would react faster to such shifts but introduces a tunable decay constant that we wanted to avoid in the headline configuration; we leave this extension to future work.

External KB entries are written, not learned. The external solver and problem KB entries that ship literature effective bands and pitfalls are written from cited references rather than learned from data, see §E.1 and §E.2. This is the same kind of generic, instance independent input that classical AC methods receive when the practitioner specifies the search space, and every entry in our KB is auditable against a public citation. Extending the KB to a new solver class still requires writing a structured spec file by hand, although the spec is short, citable to public sources, and reusable across all contexts of that solver. We are exploring an LLM authored draft step that proposes a spec from a solver's documentation with human review for the literature band claims, which would lower the cost of adding a new solver without changing the deployment time interface.