

EPOCH: Coverage-Aware Evolution with Experiential Preference Optimization for LLM-based Automated Heuristic Design*

Oguzhan Gungordu, Ayse Tursucular, Siheng Xiong, Faramarz Fekri

Georgia Institute of Technology

{ogungordu3, atursucular3, sxiong45}@gatech.edu

faramarz.fekri@ece.gatech.edu

Abstract

LLM-based automated heuristic design (AHD) discovers heuristics for combinatorial optimization problems (COPs), where an LLM generates candidates in an evolutionary search. Existing frameworks freeze the generator or fine-tune it on a scalar fitness signal hiding which instances each heuristic covers. Reflection-based frameworks improve heuristics with natural-language feedback, but it stays in the prompt and never reaches the generator’s weights. We propose **EPOCH**, which makes instance coverage the objective of the evolutionary search and fine-tunes the generator to internalize reflection. Parent selection, operator allocation, and stagnation recovery all optimize how well the heuristics jointly cover the training instances, so the search returns a complementary heuristic ensemble. The generator learns from its own search by experiential preference optimization. A teacher LLM returns a natural-language insight on the generated heuristics, and the generator produces new heuristics with this insight. These improvements join the preference pairs for direct preference optimization (DPO). Since DPO only teaches the generator to rank these improvements, a reflective self-distillation (RSD) term makes it more likely to generate them on its own, and the gains accumulate over the long horizon of the search. Across COPs and open-source LLM backbones, **EPOCH**’s heuristic ensembles outperform inference-only and fine-tuned baselines and generalize across problem scales.

1 Introduction

Many operational decisions, from vehicle routing to scheduling, are COPs (Tam et al., 2024). Because these problems are typically NP-hard, practitioners rely on heuristics that trade optimality for speed (Kirkpatrick et al., 1983; Glover, 1990). Designing a strong heuristic by hand demands expertise, and a design for one problem rarely carries

over to its variants (Choong et al., 2018; Pillay and Qu, 2018). AHD makes the heuristic itself the object of search (Burke et al., 2013), classically with genetic programming, which cannot express heuristics beyond its human-defined function set (Langdon and Poli, 2013; Dufflo et al., 2019).

LLM-based AHD replaces that function set with a large language model (LLM) that generates heuristics in the evolutionary loop (Liu et al., 2023), demonstrated by FunSearch (Romera-Paredes et al., 2023) and EoH (Liu et al., 2024). ReEvo (Ye et al., 2024) adds reflection (Shinn et al., 2023), HSEvo (Dat et al., 2025) maintains population diversity, MCTS-AHD (Zheng et al., 2025) organizes heuristics in a Monte Carlo tree, and EoH-S (Liu et al., 2026) evolves a complementary set of heuristics. All keep the generator frozen. CALM (Huang et al., 2026) and EvoTune (Šurina et al., 2025) instead fine-tune the generator on the heuristics it generates, using group-relative policy optimization (GRPO) (Shao et al., 2024) with a manually designed, graded scalar reward and DPO (Rafailov et al., 2023) with fitness-ranked pairs, respectively. In both, a heuristic is reduced to a single number, so the update drives the generator toward one best-on-average heuristic, with no reflection on the generated heuristics. Appendix A gives an extended review of related work.

Despite this progress, two limitations remain. First, no heuristic is best on every instance of a distribution (Liu et al., 2026), and a scalar fitness hides which instances each heuristic wins, so the coverage of the training instances cannot guide the evolutionary search beyond survivor selection. Second, the natural-language feedback of the inference-only methods stays inside the prompt and never reaches the generator’s weights, and the fine-tuned methods employ no reflection. We therefore propose **Experiential Preference Optimization for Coverage-aware Heuristic Design (EPOCH)**, a framework that fine-tunes the generator during

*Preprint. Under review.

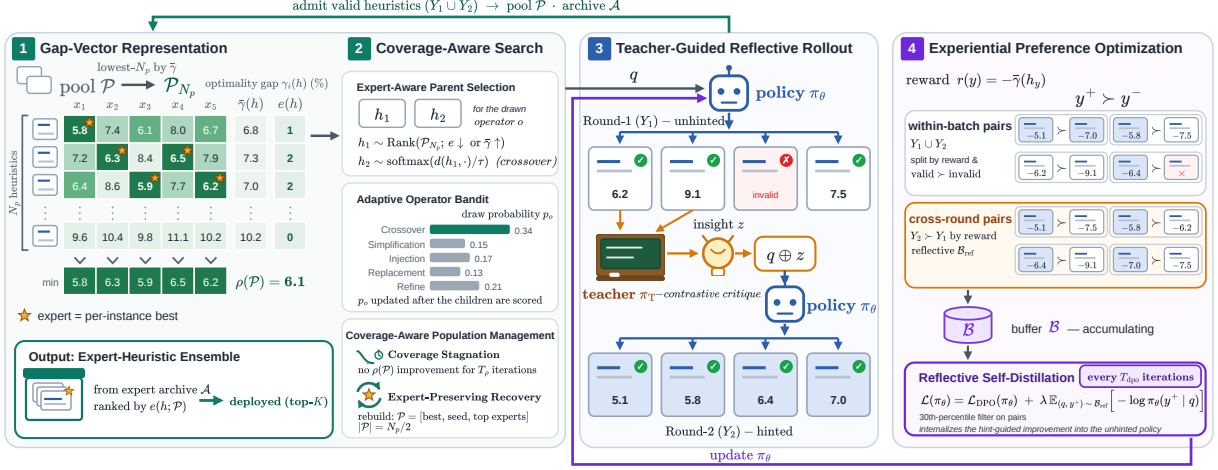


Figure 1: Overview of **EPOCH**, illustrating one training iteration. **(1)** Heuristics are represented by per-instance gap vectors, whose minima define expert heuristics and the coverage gap $\rho(\mathcal{P})$. **(2)** These drive parent selection, operator allocation, and coverage-aware population management. **(3)** The policy π_θ samples rollouts from the operator prompt q , the teacher π_T returns an insight z on the best and worst valid ones, and π_θ regenerates from $q \oplus z$. **(4)** Responses form within-batch and cross-round preference pairs; periodic DPO bursts add an RSD term internalizing the hint-guided improvement into the unhinted policy.

a coverage-aware evolutionary search and turns teacher-guided reflection into a training signal.

Our main contributions are as follows. **(1)** We develop a coverage-aware LLM-based AHD framework that fine-tunes the generator and threads each heuristic’s per-instance optimality gaps through parent selection, operator allocation, stagnation recovery, and the selection of the deployed heuristic ensemble, so the search optimizes instance coverage rather than average fitness. **(2)** We convert teacher-guided reflection into a training signal, where a teacher LLM serves only as a critic, the improvements it guides form preference pairs, and a *reflective self-distillation* term internalizes them so the gains accumulate over the search. **(3)** We design an adaptive operator bandit that favors the evolutionary operators improving instance coverage, and a coverage-based stagnation recovery with a persistent archive to preserve the best heuristics found on each instance. **(4)** Experiments across COPs and LLM backbones show that the heuristic ensembles discovered by EPOCH outperform inference-only and fine-tuned LLM-based AHD baselines and generalize across problem scales.

2 Preliminaries

Automated Heuristic Design Setup. We consider a COP with an instance space $\mathcal{X}$ and a solution space $\mathcal{S}$. Each *heuristic* $h \in \mathcal{H}$ is an executable program that constructs a feasible solution, $h : \mathcal{X} \rightarrow \mathcal{S}$, and a cost function $f : \mathcal{S} \rightarrow \mathbb{R}$

scores it. Performance is assessed on a training set $\mathcal{D}_{\text{train}} = \{x_1, \dots, x_m\}$. Conventional AHD summarizes a heuristic by the scalar fitness

$$P(h) = \frac{1}{m} \sum_{i=1}^m [-f(h(x_i))] \quad (1)$$

and, under an evaluation budget, returns the single best-on-average heuristic $h^* = \arg \max_{h \in \mathcal{H}} P(h)$. EPOCH instead searches for a set of complementary heuristics covering $\mathcal{D}_{\text{train}}$ and retains each heuristic’s full per-instance performance profile (Section 3.1) rather than the scalar of Eq. (1).

Direct Preference Optimization. EPOCH updates the generator with DPO (Rafailov et al., 2023), which trains a policy π_θ on triplets of a prompt q , a preferred response y^+ , and a dispreferred response y^- , relative to a reference policy π_{ref} . With the policy-side *implicit reward* $\hat{r}_\theta(y) = \beta \log(\pi_\theta(y | q) / \pi_{\text{ref}}(y | q))$, the loss

$$\mathcal{L}_{\text{DPO}}(\pi_\theta) = -\mathbb{E}_{(q, y^+, y^-)} \left[\log \sigma(\hat{r}_\theta(y^+) - \hat{r}_\theta(y^-)) \right] \quad (2)$$

widens the *margin* $\hat{r}_\theta(y^+) - \hat{r}_\theta(y^-)$, where σ is the logistic function and β regularizes toward π_{ref} . As the evolving policy itself generates our preference data, DPO runs in periodic bursts.

3 Methodology

EPOCH augments an LLM-based evolutionary loop with two coupled ideas. The search main-

tains a heuristic pool $\mathcal{P}$, initialized from a seed heuristic; its $N_p=10$ heuristics with the lowest average gap form the reproduction population $\mathcal{P}_{N_p}$. Each iteration draws an operator, builds a prompt from parents selected from this population, samples candidates from the policy π_θ , and admits the valid ones into $\mathcal{P}$. A *per-instance gap-vector representation* makes the search coverage-aware, and an *experiential preference-optimization* update, fed by teacher-guided reflection and an RSD term that internalizes its gains, fine-tunes the generator on its own evolutionary search. Sections 3.1–3.7 present the representation, parent selection, the evolutionary operators, the operator bandit, coverage stagnation and recovery, the reflective rollout, and experiential preference optimization, respectively. Figure 1 illustrates one training iteration and Algorithm 1 gives its pseudocode, with the complete pseudocode of its subalgorithms in Appendix C.

3.1 Per-Instance Gap Vectors, Experts, and Coverage

Let f_i^* denote a reference optimum for training instance x_i , e.g., an optimal tour length for the Traveling Salesman Problem (TSP) or the L_1 lower bound on bin count for online Bin Packing (OBP). The *per-instance optimality gap* of h on x_i is

$$\gamma_i(h) = \frac{f(h(x_i)) - f_i^*}{|f_i^*|}, \quad (3)$$

which is 0 when h is optimal on x_i and grows with sub-optimality. We represent h by its *gap vector*

$$\begin{aligned} \gamma(h) &= (\gamma_1(h), \dots, \gamma_m(h)) \in \mathbb{R}^m, \\ \bar{\gamma}(h) &= \frac{1}{m} \sum_i \gamma_i(h), \end{aligned} \quad (4)$$

where $\bar{\gamma}(h)$ is the average gap over the m instances.

Given the current pool of heuristics $\mathcal{P}$, define the pool’s *best per-instance gap* as $\gamma_i^*(\mathcal{P}) = \min_{h \in \mathcal{P}} \gamma_i(h)$, the lowest gap any pool member attains on instance x_i . The heuristic achieving this minimum is the *expert heuristic* for instance x_i , and the *expert count*

$$e(h; \mathcal{P}) = |\{i : h = \arg \min_{h' \in \mathcal{P}} \gamma_i(h')\}| \quad (5)$$

counts the instances on which h is uniquely best in the pool; a heuristic with $e(h; \mathcal{P}) > 0$ is an expert heuristic. The quality of the pool as a *portfolio* is captured by the *coverage gap*

$$\rho(\mathcal{P}) = \frac{1}{m} \sum_{i=1}^m \gamma_i^*(\mathcal{P}) = \frac{1}{m} \sum_{i=1}^m \min_{h \in \mathcal{P}} \gamma_i(h), \quad (6)$$

obtained by routing each instance to its best heuristic in $\mathcal{P}$. Unlike the single-best average gap $\min_h \bar{\gamma}(h)$, the coverage gap $\rho(\mathcal{P})$ improves whenever any heuristic becomes a better expert on any instance, even while the single best heuristic is unchanged. This set-level objective has previously guided survivor selection within an inference-only search (Liu et al., 2026). EPOCH instead makes it the control signal of the entire search. Finally, the *complementarity* of two heuristics is the ℓ_1 distance between their gap vectors (Liu et al., 2026),

$$\begin{aligned} d(h, h') &= \|\gamma(h) - \gamma(h')\|_1 \\ &= \sum_{i=1}^m |\gamma_i(h) - \gamma_i(h')|, \end{aligned} \quad (7)$$

which is large when two heuristics behave differently across instances.

3.2 Expert-Aware Parent Selection

The evolutionary operator prompts are seeded with parent heuristics drawn by rank-based sampling from the reproduction population $\mathcal{P}_{N_p}$, the N_p heuristics of $\mathcal{P}$ with the lowest average gap; expert counts in this section are computed within that subset. A draw $h \sim \text{Rank}(\mathcal{P}_{N_p}; \zeta)$ selects h with probability proportional to $1/\text{rank}$ under the ordering key ζ , with $\downarrow$ and $\uparrow$ denoting descending and ascending order. The single-parent mutation operators draw

$$h_1 \sim \begin{cases} \text{Rank}(\mathcal{P}_{N_p}; e(\cdot; \mathcal{P}_{N_p}) \downarrow) & \text{w.p. } p_{\text{exp}}=0.5, \\ \text{Rank}(\mathcal{P}_{N_p}; \bar{\gamma}(\cdot) \uparrow) & \text{otherwise,} \end{cases} \quad (8)$$

where the expert-count branch, taken with probability p_{exp} , recruits expert heuristics for under-covered instances, and the average-gap branch refines strong all-rounders. The crossover operator draws $h_1 \sim \text{Rank}(\mathcal{P}_{N_p}; \bar{\gamma}(\cdot) \uparrow)$ and

$$h_2 \sim \begin{cases} \text{softmax}(d(h_1, \cdot)/\tau), \tau=0.5 & \text{w.p. } p_{\text{comp}}=0.5, \\ \text{Rank}(\mathcal{P}_{N_p}; \bar{\gamma}(\cdot) \uparrow) & \text{otherwise,} \end{cases} \quad (9)$$

where the complementarity branch, taken with probability p_{comp} , pairs the first parent with its most complementary counterpart under Eq. (7), and the average-gap branch pairs it with a strong all-rounder. The branch taken in Eq. (9) also determines the crossover variant of Section 3.3. Algorithm 2 implements this as SELECTPARENTS.

3.3 Evolutionary Operators

EPOCH evolves heuristics with five evolutionary operators, a two-parent crossover operator and four

single-parent mutation operators, all sharing a common prompt skeleton that presents each parent’s design idea, code, and average gap; the full templates and their sources are given in Appendix I.

- **Crossover** recombines two parents, with the variant set by the branch of Eq. (9):
 - *Exploitative crossover*, used when the second parent was chosen by average-gap rank, asks for a heuristic that outperforms both parents on the same instances.
 - *Complementary crossover*, used when the second parent was chosen by complementarity, asks for a heuristic that unites the parents’ strengths across all instances either of them already handles well.
- **Simplification** takes one parent and distills it into a more concise, elegant version, countering the build-up of redundant components.
- **Injection** takes one parent and adds a novel component to it while preserving its core mechanism.
- **Replacement** takes one parent, locates a component that scores competing choices nearly equally, and rewrites it to differentiate them.
- **Refine** takes one parent, identifies its single most important weakness, and rewrites it to fix that weakness while preserving the parts that already work well.

3.4 Adaptive Operator Bandit

EPOCH allocates operator probabilities adaptively, favoring operators that advance instance coverage. For each operator $o \in \mathcal{O}$ we keep a quality estimate $Q_o \in [0, 1]$, an exponential moving average of a per-draw reward. After o is drawn and its children are scored, let g_o be the fraction of the m instances on which any of its children improves the pool’s best per-instance gap measured before admission, ν_p the fraction of valid children whose average gap beats the best parent’s, and ν_v the valid fraction. The reward combines these as

$$R_o = w_e g_o + w_p \nu_p + w_v \nu_v, \quad (10)$$

with weights $(w_e, w_p, w_v) = (0.6, 0.25, 0.15)$, so the coverage gain g_o dominates. The quality is

updated as $Q_o \leftarrow (1 - \eta)Q_o + \eta R_o$ with $\eta=0.3$, and operators are drawn with probability

$$\begin{aligned} \omega_o &= w_o^0 \text{clip}(e^{\kappa(Q_o - \bar{Q})}, \frac{1}{2}, 2), \\ p_o &= \phi + (1 - k\phi) \omega_o / \sum_{o'} \omega_{o'}, \end{aligned} \quad (11)$$

where $\bar{Q}$ is the mean quality over the k eligible operators, w_o^0 a fixed prior, set to 3 for crossover and 1 otherwise, offsetting the split of the crossover draw between its two variants, $\kappa=6$ a steepness, and $\phi=0.05$ a probability floor. Centering on $\bar{Q}$ promotes above-average operators, while the clip and the floor keep every probability in a bounded band, so the bandit concentrates budget without starving any operator. After the recovery of Section 3.5 the quality estimates are stale, so all Q_o are reset to 0. Algorithms 3 and 4 implement the draw and the update as DRAWOPERATOR and UPDATEBANDIT.

3.5 Coverage Stagnation and Expert-Preserving Recovery

Self-reinforcing prompting can drive the pool onto minor variants of the current best heuristic, so EPOCH performs coverage-aware population management, detecting *coverage stagnation* and countering it with an *expert-preserving recovery* that shrinks the pool to $N_p/2$ heuristics. The stagnation criterion is keyed to the coverage gap rather than the single best heuristic, which can stay unchanged while the pool keeps gaining expert heuristics. Each iteration recomputes $\rho(\mathcal{P})$ and its best value ρ^* , registering an improvement when

$$\rho(\mathcal{P}) < \rho^* - \epsilon_\rho, \quad (12)$$

for an absolute margin $\epsilon_\rho=10^{-4}$. The recovery fires after $T_\rho=10$ iterations without improvement, or $T_{\text{cap}}=40$ iterations after the last recovery, the latter forcing a periodic reset in long improvement stretches. It rebuilds the pool from the best-by-average-gap heuristic, the seed, and the highest-expert-count heuristics, resets the bandit’s quality estimates, and restarts the stagnation window by setting ρ^* to the recovered pool’s coverage gap. A persistent *expert archive* $\mathcal{A}$, untouched by recovery, retains the best heuristics found on each training instance, so the pool can shrink aggressively without losing discovered expert heuristics. Algorithm 5 implements the recovery as RECOVER.

3.6 Teacher-Guided Reflective Rollout

Each iteration, the policy first generates the Round-1 rollouts Y_1 , $N_1=4$ responses to the operator

Algorithm 1 EPOCH: one training iteration

Input: policy π_θ ; teacher π_T ; pool $\mathcal{P}$; expert archive $\mathcal{A}$; preference buffer $\mathcal{B}$; iteration index t

Output: updated $\mathcal{P}$, $\mathcal{A}$, $\mathcal{B}$, π_θ

```
1:  $\backslash\backslash$  Coverage-aware population management
2:  $\mathcal{P} \leftarrow \text{RECOVER}(\mathcal{P})$   $\triangleright$  stagnation and rebuild, Alg. 5
3:  $\backslash\backslash$  Reflective rollout
4:  $o \leftarrow \text{DRAWOPERATOR}()$   $\triangleright$  bandit draw, Alg. 3
5:  $S \leftarrow \text{SELECTPARENTS}(\mathcal{P}, o)$   $\triangleright$  expert-aware, Alg. 2
6:  $q \leftarrow \text{Prompt}(o, S)$   $\triangleright$  operator prompt
7:  $Y_1 \leftarrow \{y_j \sim \pi_\theta(\cdot | q)\}_{j=1}^{N_1}$   $\triangleright$  Round 1, unhinted; evaluate  $\gamma$ 
8:  $z \leftarrow \pi_T(q, Y_1)$   $\triangleright$  natural-language insight
9:  $Y_2 \leftarrow \{y_j \sim \pi_\theta(\cdot | q \oplus z)\}_{j=1}^{N_2}$   $\triangleright$  Round 2, hinted; evaluate  $\gamma$ 
10:  $\backslash\backslash$  Admission, pairing, and bookkeeping
11:  $F \leftarrow (\gamma_1^*(\mathcal{P}), \dots, \gamma_m^*(\mathcal{P}))$   $\triangleright$  pre-admission snapshot
12:  $\mathcal{P} \leftarrow \mathcal{P} \cup \{h_y \mid y \in Y_1 \cup Y_2, y \text{ valid}\}$   $\triangleright$  admit
13:  $\mathcal{B} \leftarrow \mathcal{B} \cup \text{BUILDPAIRS}(q, Y_1, Y_2)$   $\triangleright$  pairs, Alg. 6
14:  $\text{UPDATEBANDIT}(o, Y_1, Y_2, F)$   $\triangleright$  reward, Alg. 4
15:  $\mathcal{A} \leftarrow \text{UpdateArchive}(\mathcal{A}, \mathcal{P})$   $\triangleright$  per-instance experts
16:  $\backslash\backslash$  Policy update
17: if  $t \bmod T_{\text{dpo}} = 0$  then
18:    $\pi_\theta \leftarrow \text{DPOBURST}(\pi_\theta, \mathcal{B})$   $\triangleright$  DPO + RSD, Eq. (14)
19: end if
20: return  $\mathcal{P}$ ,  $\mathcal{A}$ ,  $\mathcal{B}$ ,  $\pi_\theta$ 
```

prompt q . Each response states a design idea and implements it as an executable heuristic. An external *teacher* LLM π_T reads q and Y_1 and returns a natural-language *insight* z , and the policy generates the Round-2 rollouts Y_2 , $N_2=4$ responses to the insight-augmented prompt $\tilde{q} = q \oplus z$; we refer to $\tilde{q}$ and q as the *hinted* and *unhinted* prompts. The teacher contributes only this critique, queried once per iteration, and generates neither heuristics nor training data, so every heuristic and every preference pair comes from the policy itself. The teacher’s full prompt appears in Appendix I. The critique adapts to how many *valid* responses Round 1 produced, where a response y is valid when it follows the output format, parses into an executable heuristic h_y , and runs on the training instances without failure.

- *Contrastive critique.* With two or more valid heuristics, the teacher contrasts the best and worst, reporting the design choices that reduced the gap and an instance-grounded summary of the training instances each wins.
- *Repair critique.* With one valid heuristic, the teacher pairs it with a failed one and combines what worked with what to repair.
- *Diagnostic critique.* With no valid heuristic, the teacher inspects two distinct failure modes and proposes a runnable direction.

3.7 Experiential Preference Optimization

An iteration produces a pooled batch $Y_1 \cup Y_2$ of valid and invalid responses. Each valid response receives the reward $r(y) = -\bar{\gamma}(h_y)$, the negative average gap of its heuristic. Invalid responses require no reward, only a failure-type label that selects the repair and diagnostic critiques of Section 3.6. Two families of preference pairs are appended to a growing buffer $\mathcal{B}$ by BUILDPAIRS in Algorithm 6, all conditioned on the *unhinted* prompt q :

- **Within-batch pairs.** Following Šurina et al. (2025), the valid responses are split by reward into halves and randomly matched across them, each used at most once. Each invalid response is paired as dispreferred against a random valid response, teaching the policy to avoid unparseable or crashing code.
- **Cross-round pairs.** Each valid Round-2 hinted response $y^+ \sim \pi_\theta(\cdot | \tilde{q})$ is preferred over a strictly worse valid Round-1 unhinted response $y^- \in Y_1$ with $r(y^-) < r(y^+)$. These pairs encode the lift from reflection and form the *reflective subset* $\mathcal{B}_{\text{ref}}$.

The insight z biases Round 2 toward better heuristics but is only a sampling aid; every pair conditions on the unhinted q , so the policy is trained on the prompt distribution it generates from, and the optimality gap certifies every pair’s ordering.

Every $T_{\text{dpo}}=20$ iterations we run a preference-optimization burst, the DPOBURST step of Algorithm 1, over the accumulated buffer. A threshold filter drops pairs whose preferred reward falls below the 30th percentile of the valid-output rewards observed since the previous burst (Šurina et al., 2025), and the buffer accumulates across all iterations, which preserves diversity. The reference π_{ref} is the current model with its low-rank adaptation (LoRA) disabled (Hu et al., 2022).

Reflective Self-Distillation. Optimizing the buffer with Eq. (2) teaches the policy to *rank* a reflective winner y^+ above the worse unhinted candidate y^- , but ranking is not generation. The loss enters a pair only through the margin $\hat{r}_\theta(y^+) - \hat{r}_\theta(y^-)$ and need not raise $\log \pi_\theta(y^+ | q)$; it is met largely by lowering $\hat{r}_\theta(y^-)$, and the reflective pairs are also a minority of the buffer with small margins, since both responses are valid heuristics. The unhinted policy thus learns to prefer the reflective winner without becoming

Method		TSP-20	TSP-50	TSP-100	TSP-200
<i>Classical heuristics</i>					
Nearest Neighbor		17.64	21.96	24.58	26.10
<i>LLM-based AHD: GPT-4o-mini (closed-source)</i>					
ReEvo	Top-1	10.55 $\pm$ 3.58	13.89 $\pm$ 4.48	15.43 $\pm$ 4.77	16.64 $\pm$ 5.00
	Top-3	9.18 $\pm$ 2.55	12.12 $\pm$ 2.66	13.66 $\pm$ 3.29	17.01 $\pm$ 4.28
	Top-10	8.49 $\pm$ 2.52	10.92 $\pm$ 2.29	12.48 $\pm$ 2.70	14.29 $\pm$ 2.98
HSEvo	Top-1	12.58 $\pm$ 4.18	16.37 $\pm$ 4.62	18.56 $\pm$ 4.94	20.86 $\pm$ 5.30
	Top-3	11.38 $\pm$ 5.16	15.12 $\pm$ 5.64	17.39 $\pm$ 5.91	20.14 $\pm$ 5.98
	Top-10	10.59 $\pm$ 5.81	14.28 $\pm$ 6.32	16.66 $\pm$ 6.50	19.65 $\pm$ 6.48
MCTS-AHD	Top-1	7.28 $\pm$ 0.71	10.77 $\pm$ 0.71	12.82 $\pm$ 0.67	14.75 $\pm$ 1.03
	Top-3	6.25 $\pm$ 1.07	9.50 $\pm$ 1.08	11.77 $\pm$ 1.14	14.00 $\pm$ 1.18
	Top-10	4.54 $\pm$ 1.20	7.94 $\pm$ 1.17	10.49 $\pm$ 1.25	12.85 $\pm$ 1.41
EoH-S	Top-1	8.56 $\pm$ 0.91	11.69 $\pm$ 0.71	13.80 $\pm$ 0.91	15.05 $\pm$ 0.98
	Top-3	5.05 $\pm$ 0.43	9.14 $\pm$ 0.61	11.80 $\pm$ 0.83	13.75 $\pm$ 0.85
	Top-10	3.22 $\pm$ 0.41	7.30 $\pm$ 0.38	10.06 $\pm$ 0.38	12.39 $\pm$ 0.42
<i>LLM-based AHD: Llama-3.1-8B</i>					
ReEvo	Top-1	10.58 $\pm$ 0.58	14.87 $\pm$ 1.36	18.38 $\pm$ 2.10	26.06 $\pm$ 12.93
	Top-3	10.28 $\pm$ 0.59	14.15 $\pm$ 2.15	16.19 $\pm$ 1.65	19.05 $\pm$ 1.37
	Top-10	10.25 $\pm$ 0.62	14.10 $\pm$ 2.24	15.96 $\pm$ 2.01	18.69 $\pm$ 1.54
HSEvo	Top-1	11.46 $\pm$ 1.69	15.85 $\pm$ 2.63	18.48 $\pm$ 4.51	21.34 $\pm$ 7.00
	Top-3	10.61 $\pm$ 1.28	14.83 $\pm$ 1.76	17.27 $\pm$ 2.82	19.47 $\pm$ 3.89
	Top-10	10.29 $\pm$ 1.54	14.45 $\pm$ 2.10	16.80 $\pm$ 3.24	18.99 $\pm$ 4.31
MCTS-AHD	Top-1	10.27 $\pm$ 0.37	14.23 $\pm$ 0.65	16.39 $\pm$ 1.48	17.87 $\pm$ 1.45
	Top-3	9.76 $\pm$ 0.40	13.40 $\pm$ 0.39	15.40 $\pm$ 0.81	16.92 $\pm$ 1.04
	Top-10	7.58 $\pm$ 0.63	11.73 $\pm$ 0.61	13.77 $\pm$ 0.91	15.51 $\pm$ 1.15
EoH-S	Top-1	10.69 $\pm$ 0.93	15.43 $\pm$ 1.58	17.86 $\pm$ 1.38	20.04 $\pm$ 1.46
	Top-3	6.05 $\pm$ 0.72	11.68 $\pm$ 1.06	14.59 $\pm$ 0.93	17.31 $\pm$ 1.05
	Top-10	4.70 $\pm$ 0.81	10.46 $\pm$ 1.02	13.60 $\pm$ 0.96	16.45 $\pm$ 1.07
CALM	Top-1	6.76 $\pm$ 0.26	10.02 $\pm$ 0.26	11.75 $\pm$ 0.27	13.47 $\pm$ 0.31
	Top-3	6.55 $\pm$ 0.23	9.68 $\pm$ 0.55	11.34 $\pm$ 0.51	13.33 $\pm$ 0.47
	Top-10	6.19 $\pm$ 0.54	9.31 $\pm$ 0.80	11.04 $\pm$ 0.67	13.30 $\pm$ 0.91
EPOCH (Ours)	Top-1	6.19 $\pm$ 0.30	9.61 $\pm$ 0.25	11.63 $\pm$ 0.21	13.57 $\pm$ 0.36
	Top-3	4.31 $\pm$ 0.30	7.62 $\pm$ 0.23	9.85 $\pm$ 0.36	11.98 $\pm$ 0.49
	Top-10	2.81 $\pm$ 0.22	5.73 $\pm$ 0.15	8.15 $\pm$ 0.19	10.74 $\pm$ 0.38

Table 1: TSP optimality gap (%) to LKH-3 optima; each cell is the mean over runs with standard deviation, lower is better. Size $n=50$ is ID (underlined); $n \in \{20, 100, 200\}$ are OOD. Within each backbone block the best method per metric and test set is **shaded**; each test set’s overall best result is shown in **bold**.

more likely to generate it, although every Round-1 rollout is drawn from that policy.

To close this gap we distill the hinted policy into the unhinted one. Following [Zhao and Li \(2025\)](#), their discrepancy is the forward KL divergence

$$\mathcal{L}_{\text{RSD}}(\pi_\theta) = \mathbb{E}_{q,z} \text{KL}(\pi_\theta(\cdot | \tilde{q}) \| \pi_\theta(\cdot | q)), \quad (13)$$

which we estimate, as a reward-filtered surrogate, from the Round-2 winner y^+ , since y^+ is a draw from $\pi_\theta(\cdot | \tilde{q})$ and freezing that hinted distribution reduces the gradient of Eq. (13) to that of the negative log-likelihood $-\log \pi_\theta(y^+ | q)$. This RSD term, applied to $\mathcal{B}_{\text{ref}}$ alone, gives the burst objective

$$\begin{aligned} \mathcal{L}(\pi_\theta) &= \mathcal{L}_{\text{DPO}}(\pi_\theta) \\ &+ \lambda \mathbb{E}_{(q, y^+) \sim \mathcal{B}_{\text{ref}}} [-\log \pi_\theta(y^+ | q)], \end{aligned} \quad (14)$$

with $\lambda=0.1$ in all main runs and Appendix D giving the derivation. Where DPO is relative, RSD is absolute, directly raising the probability that the unhinted policy generates the reflective winner. Restricting the term to $\mathcal{B}_{\text{ref}}$, whose pairs contain by construction a valid and better hint-guided heuristic, distills genuine reflective gains. Because the Round-1 rollouts are drawn from $\pi_\theta(\cdot | q)$, the internalized improvement makes them better on their own, and the gain accumulates over the search.

4 Experiments

We evaluate EPOCH on three COPs, the TSP, OBP, and Knapsack Problem (KP), solved within a step-by-step construction framework. We assess the single best heuristic, the expert-heuristic ensemble, and generalization from the in-domain (ID) training size to out-of-distribution (OOD) sizes. Problem definitions, datasets, and evaluation protocol are in Appendix B, and the KP results in Appendix E.1.

Benchmarks and metrics. We report mean optimality gaps in percent, where lower is better. For TSP this is the gap of the tour length to the LKH-3 ([Helsgaun, 2017](#)) optimum averaged over 300 test instances per size, and for OBP the gap of the bin count to the L_1 lower bound averaged over the Weibull test instances. Each method saves a set of heuristics, scored at two granularities. *Top-1* evaluates the saved heuristic with the best training fitness, while *top-K* evaluates the saved set as a portfolio, taking on each test instance the lowest cost among the K heuristics. We use $K \in \{1, 3, 10\}$ for TSP and $K \in \{1, 4\}$ for OBP. EPOCH’s saved set is the expert-heuristic ensemble from its archive $\mathcal{A}$, ranked by expert count; Appendix B states how each baseline ranks its saved set.

Backbones and settings. We run EPOCH with two open-source backbones, Qwen2.5-7B and

Method		1k_100	1k_500	5k_100	5k_500	10k_100	10k_500	Avg.
First-Fit		5.02	<i>Classical heuristics</i>					
Best-Fit		4.77	0.25	4.65	0.55	4.36	0.50	2.55
			0.25	4.31	0.55	4.05	0.47	2.40
<i>LLM-based AHD: GPT-4o-mini (closed-source)</i>								
ReEvo	Top-1	3.74 $\pm$ 0.94	0.40 $\pm$ 0.30	3.09 $\pm$ 1.12	0.69 $\pm$ 0.30	2.84 $\pm$ 1.11	0.53 $\pm$ 0.12	1.88 $\pm$ 0.46
	Top-4	3.61 $\pm$ 1.01	0.40 $\pm$ 0.30	2.98 $\pm$ 1.21	0.56 $\pm$ 0.13	2.74 $\pm$ 1.22	0.44 $\pm$ 0.03	1.79 $\pm$ 0.53
HSEvo	Top-1	2.85 $\pm$ 1.01	0.40 $\pm$ 0.30	2.19 $\pm$ 1.15	0.56 $\pm$ 0.02	2.01 $\pm$ 1.15	0.50 $\pm$ 0.04	1.42 $\pm$ 0.52
	Top-4	2.82 $\pm$ 1.01	0.20 $\pm$ 0.10	2.17 $\pm$ 1.17	0.47 $\pm$ 0.12	2.01 $\pm$ 1.15	0.42 $\pm$ 0.10	1.35 $\pm$ 0.56
MCTS-AHD	Top-1	2.59 $\pm$ 0.45	0.15 $\pm$ 0.12	1.88 $\pm$ 0.47	0.42 $\pm$ 0.17	1.71 $\pm$ 0.41	0.38 $\pm$ 0.12	1.19 $\pm$ 0.27
	Top-4	2.48 $\pm$ 0.51	0.10 $\pm$ 0.12	1.79 $\pm$ 0.53	0.36 $\pm$ 0.17	1.61 $\pm$ 0.48	0.33 $\pm$ 0.13	1.11 $\pm$ 0.31
EoH-S	Top-1	2.62 $\pm$ 0.33	0.25 $\pm$ 0.16	1.74 $\pm$ 0.73	0.42 $\pm$ 0.15	1.46 $\pm$ 0.83	0.35 $\pm$ 0.13	1.14 $\pm$ 0.33
	Top-4	2.40 $\pm$ 0.38	0.15 $\pm$ 0.12	1.41 $\pm$ 0.56	0.24 $\pm$ 0.16	1.08 $\pm$ 0.58	0.18 $\pm$ 0.15	0.91 $\pm$ 0.30
<i>LLM-based AHD: Llama-3.1-8B</i>								
ReEvo	Top-1	4.34 $\pm$ 0.70	0.25 $\pm$ 0.00	3.27 $\pm$ 1.25	0.41 $\pm$ 0.17	3.04 $\pm$ 1.36	0.36 $\pm$ 0.16	1.95 $\pm$ 0.60
	Top-4	4.10 $\pm$ 0.87	0.20 $\pm$ 0.10	3.23 $\pm$ 1.24	0.41 $\pm$ 0.17	2.98 $\pm$ 1.34	0.35 $\pm$ 0.16	1.88 $\pm$ 0.64
HSEvo	Top-1	3.39 $\pm$ 0.93	0.25 $\pm$ 0.00	2.42 $\pm$ 1.26	0.42 $\pm$ 0.18	2.11 $\pm$ 1.33	0.36 $\pm$ 0.15	1.49 $\pm$ 0.58
	Top-4	3.36 $\pm$ 0.97	0.25 $\pm$ 0.00	2.42 $\pm$ 1.26	0.40 $\pm$ 0.16	2.10 $\pm$ 1.33	0.36 $\pm$ 0.15	1.48 $\pm$ 0.59
MCTS-AHD	Top-1	4.14 $\pm$ 0.89	0.17 $\pm$ 0.12	3.54 $\pm$ 1.09	0.46 $\pm$ 0.12	3.30 $\pm$ 1.07	0.40 $\pm$ 0.11	2.00 $\pm$ 0.57
	Top-4	4.11 $\pm$ 0.94	0.17 $\pm$ 0.12	3.54 $\pm$ 1.09	0.46 $\pm$ 0.12	3.30 $\pm$ 1.07	0.40 $\pm$ 0.11	1.99 $\pm$ 0.57
EoH-S	Top-1	3.92 $\pm$ 0.78	0.21 $\pm$ 0.09	2.89 $\pm$ 0.87	0.51 $\pm$ 0.07	2.59 $\pm$ 1.00	0.45 $\pm$ 0.05	1.76 $\pm$ 0.38
	Top-4	3.56 $\pm$ 0.73	0.08 $\pm$ 0.12	1.46 $\pm$ 0.57	0.16 $\pm$ 0.09	1.14 $\pm$ 0.69	0.14 $\pm$ 0.10	1.09 $\pm$ 0.27
CALM	Top-1	3.71 $\pm$ 0.52	0.59 $\pm$ 0.82	1.32 $\pm$ 0.40	0.30 $\pm$ 0.14	0.82 $\pm$ 0.41	0.27 $\pm$ 0.14	1.17 $\pm$ 0.22
	Top-4	3.69 $\pm$ 0.51	0.59 $\pm$ 0.82	1.29 $\pm$ 0.43	0.29 $\pm$ 0.15	0.80 $\pm$ 0.43	0.26 $\pm$ 0.13	1.15 $\pm$ 0.23
EPOCH (Ours)	Top-1	2.99 $\pm$ 0.70	0.30 $\pm$ 0.36	1.66 $\pm$ 0.90	0.32 $\pm$ 0.21	1.33 $\pm$ 0.97	0.28 $\pm$ 0.24	1.15 $\pm$ 0.39
	Top-4	2.34 $\pm$ 0.43	0.10 $\pm$ 0.12	0.74 $\pm$ 0.06	0.20 $\pm$ 0.07	0.36 $\pm$ 0.06	0.13 $\pm$ 0.04	0.64 $\pm$ 0.07

Table 2: OBP optimality gap (%) to the L_1 lower bound on the six held-out Weibull test sets; each cell is the mean over runs with standard deviation, lower is better. Columns are (items, capacity), e.g. 1k_100 is 1,000 items at capacity 100; Avg. is the mean over the six. Item scales 1k, 5k are ID (underlined); 10k is OOD. Within each backbone block the best Avg. per metric is shaded; each test set’s overall best result is shown in bold.

Llama-3.1-8B, fine-tuned online during the search, with GPT-5-nano as the teacher (Appendix G.2 shows the gains persist with other teachers, including its own backbone). Every method uses a budget of 1,000 heuristic generations, which for EPOCH corresponds to 125 iterations. Full hyperparameters are in Appendix H.

Baselines. We compare against five LLM-based AHD methods, CALM (Huang et al., 2026), which fine-tunes the generator, and the inference-only ReEvo (Ye et al., 2024), HSEvo (Dat et al., 2025), MCTS-AHD (Zheng et al., 2025), and EoH-S (Liu et al., 2026). The inference-only methods are also run with GPT-4o-mini and, in Appendix E.1, with GPT-5-nano; CALM and EPOCH require open backbones. For a fair comparison, where an open-backbone baseline makes auxiliary LLM calls that do not generate heuristics, such as reflection, they are served by GPT-5-nano, the teacher model of EPOCH (Appendix H). We further include the classical heuristics nearest-neighbor (Rosenkrantz et al., 1977) for TSP, and First-Fit and Best-Fit (Johnson et al., 1974) for OBP.

4.1 Main Results

TSP. As reported in Table 1 (Qwen2.5-7B in Table 5), EPOCH attains the best or a comparable top-1 gap in every size-backbone setting, improving on CALM by up to 8%. The advantage widens at the ensemble level, where the top-10 ensemble

cuts the gap at the training size $n=50$ by 38–40% relative to CALM and 19–45% relative to the strongest inference-only baseline, and the top-3 ensemble already outperforms the top-10 ensemble of almost every baseline. The gains persist OOD, with the top-10 ensemble about 20% below CALM at $n=200$. Despite far smaller open backbones, EPOCH outperforms the inference-only baselines run with GPT-4o-mini at every metric and test size, and nearest-neighbor by a wide margin.

OBP. Across the six test sets of Table 2 (Qwen2.5-7B in Table 6), EPOCH attains the lowest average gap of the open-backbone methods at both top-1 and top-4, with a top-4 average 44–55% below CALM, over 40% below the strongest open-backbone inference-only baseline, and 13–30% below the strongest GPT-4o-mini baseline.

Single-best versus ensemble. TSP, OBP, and KP show the same pattern, the best or a comparable top-1 heuristic and far larger gains at the ensemble level. Because parent selection, operator allocation, stagnation recovery, and the deployed ensemble all optimize instance coverage rather than average fitness, the search yields a heuristic set whose per-instance best cost falls far below that of any single heuristic. Full results with Qwen2.5-7B and all classical heuristics, seed-robustness studies, evolutionary curves, and TSPLIB results appear in Appendix E, and significance tests in Appendix F.

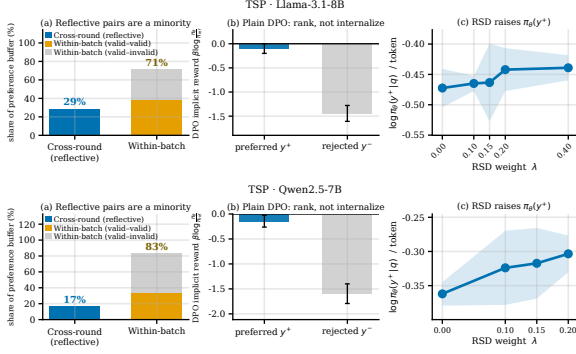


Figure 2: Motivation for RSD, on TSP with Llama-3.1-8B (top) and Qwen2.5-7B (bottom): (a) reflective cross-round pairs are a minority of the preference buffer; (b) plain DPO at $\lambda=0$ satisfies the reflective pairs’ margin by lowering the implicit reward of the rejected response, not raising the winner’s; (c) increasing the RSD weight λ raises the unhinted policy’s per-token log-likelihood of the reflective winners.

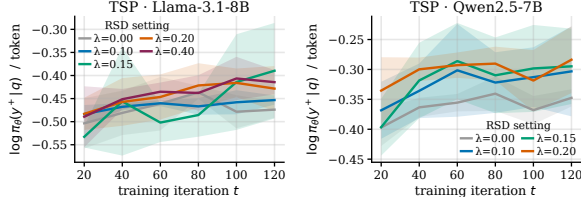


Figure 3: Internalization over training, on TSP with Llama-3.1-8B (left) and Qwen2.5-7B (right). The unhinted policy’s per-token log-likelihood of the reflective winners rises across DPO bursts for $\lambda > 0$ and stays flat at $\lambda=0$.

4.2 Ablation Study

We ablate EPOCH one component at a time, on TSP and OBP with Llama-3.1-8B over five runs, scored on 100 instances per test set. We report in Table 3 the ensemble gap, top-10 for TSP and top-4 for OBP, averaged over the test sets. Appendix G details each row’s removal and replacement, along with the hyperparameter ablation.

Gap-vector representation. Replacing the per-instance gap vector with its scalar average everywhere causes the largest degradation, widening the ensemble gap by 29% on TSP and 34% on OBP. Scoring heuristics by one number removes the per-instance structure that parent selection, the operator reward, recovery, and the deployed ensemble use.

Coverage-aware search. Every coverage mechanism contributes. Removing expert-aware parent selection, the operator bandit, or the coverage term of its reward widens the ensemble gap by 2–5% on TSP and 19–27% on OBP, with the uniform

	TSP	OBP
EPOCH (full)	6.79 ± 0.19	0.64 ± 0.07
<i>Reflective learning</i>		
w/o RSD	7.17 ± 0.27	0.79 ± 0.06
w/o cross-round pairs	6.92 ± 0.16	0.75 ± 0.10
w/o instance-grounded insight	6.90 ± 0.21	0.68 ± 0.07
w/o teacher reflection	6.99 ± 0.14	0.83 ± 0.16
w/o preference optimization	7.01 ± 0.13	0.81 ± 0.15
<i>Coverage-aware search</i>		
w/o expert-aware selection	6.93 ± 0.15	0.81 ± 0.09
w/o adaptive operator allocation	7.15 ± 0.39	0.76 ± 0.12
coverage-blind bandit	7.07 ± 0.16	0.77 ± 0.06
w/o coverage-stagnation signal	7.13 ± 0.29	0.69 ± 0.08
w/o expert-preserving recovery	6.99 ± 0.17	0.76 ± 0.06
<i>Gap-vector representation</i>		
scalar fitness (no gap vector)	8.77 ± 0.63	0.86 ± 0.18

Table 3: Component ablations on the Llama-3.1-8B backbone, reported as the ensemble optimality gap (%), lower is better. The full method is in **bold**.

operator draw the largest on TSP. Replacing the coverage-stagnation signal with a fixed-period reset and the expert-preserving recovery with a single-best rebuild degrade the ensemble by up to 5% on TSP and 19% on OBP, confirming that preserving expert heuristics through recovery is essential.

Reflective learning. The reflective and preference components contribute most on OBP, whose four training instances make lost expert heuristics harder to recover. Removing RSD, the instance-grounded insight, teacher reflection, or preference optimization widens the OBP gap by 6–30% and the TSP gap by 2–6%. The reflective cross-round pairs are a minority of the buffer, 29% for Llama and 17% for Qwen (Figure 2a), and on those pairs the DPO loss is met almost entirely by lowering the rejected response’s implicit reward rather than raising the winner’s (Figure 2b), so the update ranks the winner above the worse candidate without making the policy more likely to generate it. The RSD term supplies the missing absolute signal, as a larger weight λ raises the unhinted policy’s per-token log-likelihood of the reflective winners (Figure 2c), and this log-likelihood rises across training for $\lambda > 0$ and stays flat at $\lambda=0$ (Figure 3). The unhinted policy thus internalizes the hint-guided improvement and generates stronger Round-1 heuristics, the gain Table 3 attributes to RSD.

5 Conclusion

We propose EPOCH, an LLM-based AHD framework that fine-tunes the generator during the search and represents every heuristic by its per-instance optimality-gap vector rather than a scalar fitness. This representation makes the entire search loop coverage-aware, driving expert-aware parent selection, an adaptive operator bandit, and coverage-

based stagnation recovery. Teacher-guided reflection becomes a training signal, and RSD internalizes these improvements into the generator. Across COPs and LLM backbones, EPOCH’s complementary expert-heuristic ensembles outperform AHD methods and generalize across problem scales.

Limitations

EPOCH fine-tunes only the generator. The teacher LLM that returns the natural-language insight stays fixed throughout the search, so reflection guides the training of the generator without being trained itself. Fine-tuning the reflection jointly with the generator, toward an end-to-end fine-tuned system, is left for future work. EPOCH is evaluated within the step-by-step construction framework; Appendix B explains this choice, and solver-driven frameworks such as ant colony optimization and guided local search remain to be explored.

References

- Jon Louis Bentley. 1992. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411.
- Edmund Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. 2013. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64:1695–1724.
- Angelica Chen, David Dohan, and David So. 2023. Evo-Prompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, volume 36, pages 7787–7817.
- Shin Siang Choong, Li-Pei Wong, and Chee Peng Lim. 2018. Automatic design of hyper-heuristic based on reinforcement learning. *Information Sciences*, 436–437:89–107.
- Nicos Christofides. 1976. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University.
- Pham Vu Tuan Dat, Long Doan, and Huynh Thi Thanh Binh. 2025. Hsevo: Elevating automatic heuristic design with diversity-driven harmony search and genetic algorithm using llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26931–26938.
- Sachin Desale, Akhtar Rasool, Sushil Andhale, and Rane Priti. 2015. Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey. *International Journal of Computer Engineering in Research Trends*, 2(5):296–304.
- Gabriel Duflo, Emmanuel Kieffer, Matthias R. Brust, Grégoire Danoy, and Pascal Bouvry. 2019. A GP hyper-heuristic approach for generating TSP heuristics. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 521–529.
- Fred Glover. 1990. Tabu search—part II. *ORSA Journal on Computing*, 2(1):4–32.
- Aaron Grattafiori et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Oguzhan Gungordu, Siheng Xiong, and Faramarz Fekri. 2026. PathWise: Planning through world model for automated heuristic design via self-evolving LLMs. In *International Conference on Machine Learning*.
- Keld Helsgaun. 2017. An extension of the Lin–Kernighan–Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.
- Ziyao Huang, Weiwei Wu, Kui Wu, Wei-Bin Lee, and Jianping Wang. 2026. CALM: Co-evolution of algorithms and language model for automatic heuristic design. In *The Fourteenth International Conference on Learning Representations*.
- David S. Johnson, Alan Demers, Jeffrey D. Ullman, Michael R. Garey, and Ronald L. Graham. 1974. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325.
- S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220(4598):671–680.
- W.B. Langdon and R. Poli. 2013. *Foundations of Genetic Programming*. Springer-Verlag.
- Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. 2024. Evolution through large models. In *Handbook of Evolutionary Machine Learning*, pages 331–366. Springer.
- Fei Liu, Yilu Liu, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. 2026. EoH-S: Evolution of heuristic set using LLMs for automated heuristic design. In *The Fortieth AAAI Conference on Artificial Intelligence*.
- Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Evolution of heuristics: towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*.

- Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. 2023. Algorithm evolution using large language model. *arXiv preprint arXiv:2311.15249*.
- Fei Liu, Qingfu Zhang, Jialong Shi, Xialiang Tong, Kun Mao, and Mingxuan Yuan. 2025. Fitness landscape of large language model-assisted automated algorithm search. *arXiv preprint arXiv:2504.19636*.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhunoye, Yiming Yang, et al. 2023. Self-Refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Elliot Meyerson, Mark J. Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K. Hoover, and Joel Lehman. 2024. Language model crossover: Variation through few-shot prompting. *ACM Trans. Evol. Learn. Optim.*, 4(4).
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35.
- Nelishia Pillay and Rong Qu. 2018. Introduction to hyper-heuristics. In *Hyper-Heuristics: Theory and Applications*. Springer.
- Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with “gradient descent” and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968.
- Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36.
- Gerhard Reinelt. 1991. TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384.
- Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. 2023. Mathematical discoveries from program search with large language models. *Nature*.
- Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis, II. 1977. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. 2026. Experiential reinforcement learning. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *37th Conference on Neural Information Processing Systems*.
- Yuda Song, Lili Chen, Fahim Tajwar, Rémi Munos, Deepak Pathak, Drew Bagnell, Aarti Singh, and Andrea Zanette. 2026. Expanding the capabilities of reinforcement learning via text feedback. In *International Conference on Machine Learning*.
- Anja Šurina, Amin Mansouri, Lars C.P.M. Quaedvlieg, Amal Seddas, Maryna Viazovska, Emmanuel Abbe, and Caglar Gulcehre. 2025. Algorithm discovery with LLMs: Evolutionary search meets reinforcement learning. In *Second Conference on Language Modeling*.
- Nguyen Thi Tam, Tran Ho Khanh Ly, Bui Trong Duc, Tran Huy Hung, and Huynh Thi Thanh Binh. 2024. Multi-objective virtual network functions placement and traffic routing problem. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 01–08.
- Niki van Stein, Anna V. Kononova, Lars Kotthoff, and Thomas Bäck. 2025. Code evolution graphs: Understanding large language model driven design of algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 943–951.
- Hui Wang, Xufeng Zhang, and Chaoxu Mu. 2025. Planning of heuristics: Strategic planning on large language models with Monte Carlo tree search for automating heuristic optimization. *arXiv preprint arXiv:2502.11422*.
- Bernard L. Welch. 1947. The generalization of ‘Student’s’ problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.
- Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.
- Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. 2024. ReEvo: large language models as hyper-heuristics with reflective evolution. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press.

Zihui Zhao and Zechang Li. 2025. Reflective preference optimization (RPO): Enhancing on-policy alignment via hint-guided reflection. *arXiv preprint arXiv:2512.13240*.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. 2025. Monte Carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In *Forty-second International Conference on Machine Learning*.

A Extended Related Work

Heuristic design and hyper-heuristics. Exact algorithms for NP-hard COPs rarely scale to practical instance sizes, so routing, scheduling, and design problems are commonly solved with heuristics (Desale et al., 2015; Tam et al., 2024) and with metaheuristics such as simulated annealing and tabu search (Kirkpatrick et al., 1983; Glover, 1990). Such solvers encode expert knowledge, and adapting one to a new problem variant largely restarts the design effort (Pillay and Qu, 2018; Choong et al., 2018). Hyper-heuristics and AHD instead automate the design itself, selecting or constructing heuristics for a problem class (Burke et al., 2013). Genetic programming is the classical realization, evolving heuristic programs built from a human-defined function set (Langdon and Poli, 2013; Duffo et al., 2019), and free-form LLM code generation widens the search space beyond what that function set expresses.

LLMs in evolutionary search. A line of work runs a standard evolutionary algorithm but lets an LLM generate the offspring, prompting it with the selected parents to produce new code or text, while fitness evaluation and survivor selection stay outside the model (Lehman et al., 2024; Meyerson et al., 2024; Chen et al., 2023). LLM-based AHD builds on this setup, and EPOCH keeps the same outer loop but additionally *trains* the operator rather than treating it as fixed.

LLM-based AHD frameworks. FunSearch (Romera-Paredes et al., 2023) pioneered evolutionary program search with an LLM proposer, EoH (Liu et al., 2024) co-evolves thoughts and code, ReEvo (Ye et al., 2024) adds reflection (Shinn et al., 2023; Pryzant et al., 2023), HSEvo (Dat et al., 2025) couples diversity metrics

with harmony search, and MCTS-AHD (Zheng et al., 2025) organizes heuristics in a Monte Carlo tree. Analyses of the AHD fitness landscape and code-evolution dynamics further motivate diversity (Liu et al., 2025; van Stein et al., 2025), and planning-based approaches organize the search over a tree (Wang et al., 2025) or an entailment graph that carries derivation history (Gungordu et al., 2026). Most relevant to our coverage view, EoH-S (Liu et al., 2026) evolves a *set* of complementary heuristics, using per-instance performance vectors to select survivors that specialize on different parts of the instance distribution. All of these methods keep the LLM frozen, whereas EPOCH trains the generator and uses the per-instance best gaps as an online control signal throughout the run.

Training the LLM generator for AHD. CALM (Huang et al., 2026) co-evolves the heuristic pool and the LLM, fine-tuning with GRPO (Shao et al., 2024) on a progressive scalar reward and using a stagnation-driven collapse that resets the pool to the seed and current best. EvoTune (Šurina et al., 2025) interleaves FunSearch-style search with periodic DPO over a program database treated as a replay buffer, using a forward-KL variant that preserves output diversity and a dynamic reward-percentile filter. EPOCH likewise fine-tunes the generator, but governs selection, operator allocation, and recovery by per-instance gap vectors rather than scalar fitness, converts teacher-guided reflection into preference pairs and distills the hint-guided improvement into the unhinted policy with an RSD term, and deploys a complementary ensemble of expert heuristics.

Reflection and natural-language feedback. Reflexion (Shinn et al., 2023) and Self-Refine (Madaan et al., 2023) improve LLM outputs through in-context natural-language feedback, ExpeL (Zhao et al., 2024) reuses accumulated experience across tasks, and prompt-optimization methods treat textual critiques as gradients (Pryzant et al., 2023); all act at inference time without updating weights. A recent thread instead folds text feedback into training. Text feedback can be distilled into first-attempt behavior so the model needs no feedback at test time (Song et al., 2026), and an experience–reflection–consolidation loop can internalize corrective revisions into the policy (Shi et al., 2026), though both target domains other

than heuristic design. EPOCH brings this idea to AHD, where the executable, verifiable reward orders reflection-conditioned generations without a learned judge, and its RSD term internalizes the hint-guided improvement into the policy’s unhinted behavior so it carries into later generations of the search.

Preference optimization and learning from feedback. Aligning LLMs from preferences was popularized by reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022); DPO (Rafailov et al., 2023) removes the learned reward model by optimizing preferences directly, and Reflective Preference Optimization (Zhao and Li, 2025) constructs the pairs by conditioning regeneration on a critic’s hint and contrasting hinted with unhinted responses, keeping the update on-policy. EPOCH adopts DPO for the generator together with this reflection-as-preference construction, but the preference ordering is certified by the optimality gap on executed code rather than by a learned or model-based judge, and the contrasted responses are a hinted second batch versus a strictly worse unhinted first batch from the same evolutionary iteration. That work also distills the hint-conditioned behavior back into the unconditioned policy to remove a test-time dependence on the hint. EPOCH’s RSD term shares this mechanism but is motivated within training, makes each iteration’s unhinted rollouts better and accumulates across generations, and is gated to the pairs whose verifiable reward certifies a genuine reflective gain.

B Problem Definitions and Datasets

We instantiate EPOCH on three COPs, solved within a step-by-step construction framework; the function signatures match the conventions of prior LLM-based AHD work (Zheng et al., 2025; Ye et al., 2024).

Solution framework. We evaluate within the step-by-step construction framework because there the heuristic alone builds the solution, so the optimality gap measures the heuristic an AHD framework discovered. In solver-driven frameworks such as ant colony optimization and guided local search (Ye et al., 2024; Zheng et al., 2025), the heuristic only biases a solver that continues the search on its own, through pheromone updates or local search, so the solver narrows the differences between heuristics and the gap reflects the solver

as much as the discovery method. Each candidate evaluation also runs the solver on every training instance, which makes it far costlier than a construction run and the 1,000-generation search of every method correspondingly slower. Extending EPOCH to these frameworks is left for future work.

Traveling Salesman Problem (TSP). A TSP instance is a complete graph on n nodes with coordinates sampled uniformly from $[0, 1]^2$ and Euclidean edge costs; a solution is a Hamiltonian tour scored by its total length. Within the step-by-step construction framework, the LLM designs the key function below, which chooses the next city from the current one given the unvisited set and the distance matrix, and the framework calls it repeatedly until the tour is complete.

TSP key function

```
select_next_node(current_node, destination_node,
                 unvisited_nodes, distance_matrix)
-> next_node
```

Per-instance gaps in Eq. (3) are computed against LKH-3 (Helsgaun, 2017) optimal tour lengths. Training uses $m=64$ instances at $n=50$; held-out evaluation uses a range of sizes ($n \in \{20, 50, 100, 200\}$) to test OOD generalization.

Online Bin Packing (OBP). In OBP, items arrive sequentially with sizes drawn from a Weibull distribution and must be placed immediately into a bin that can hold them, minimizing the number of bins used. The LLM designs the priority function below, which scores each feasible bin for the arriving item, and the item is placed in the highest-priority bin.

OBP key function

```
step(item_size, remaining_capacity) -> priority
```

Per-instance gaps are computed against the L_1 lower bound $\lceil (\sum_i s_i)/C \rceil$ on the number of bins (Liu et al., 2024). Because the training set comprises few instances, the inference-time ensemble is reported up to the available number of experts.

0-1 Knapsack Problem (KP). Following the KP setting of MCTS-AHD and ReEvo (Zheng et al., 2025; Ye et al., 2024), an instance has n items with weights and values sampled uniformly from $[0, 1]$ and a knapsack of capacity 25 (12.5 at $n=50$); a solution is a subset of items whose total weight fits the capacity, scored by its total value. Within the step-by-step construction framework, the LLM

designs the key function below, which chooses the next item to pack given the remaining capacity and the weights and values of the items not yet packed, and the framework calls it repeatedly until no remaining item fits.

KP key function

```
select_next_item(remaining_capacity, weights,
                 values) -> next_item
```

Per-instance gaps are computed against the optimal value from dynamic programming, with the negated value as the cost in Eq. (3). Training uses $m=64$ instances at $n=100$; held-out evaluation uses $n \in \{50, 100, 200\}$.

Datasets and evaluation protocol. As summarized in Table 4, for TSP we train on 64 instances at $n=50$ and evaluate on held-out sets of 300 instances at each of $n \in \{20, 50, 100, 200\}$, where $n=50$ is ID and the other sizes are OOD. For OBP we train on four Weibull instances spanning $\{1k, 5k\}$ items at capacity 100, and evaluate on the six held-out Weibull sets $\{1k, 5k, 10k\} \times \{100, 500\}$ of five instances each; the 10k-item sets are OOD in scale. For KP we train on 64 instances at $n=100$ and evaluate on held-out sets of 1,000 instances at each of $n \in \{50, 100, 200\}$, where $n=100$ is ID, with $K \in \{1, 3, 10\}$ as for TSP. Following standard practice, both training and test instances are sampled under fixed settings used by prior LLM-based AHD work (Ye et al., 2024; Liu et al., 2024; Zheng et al., 2025), and every method sees only the training set during search and is scored on the held-out sets afterward. We report the mean over five independent runs. During search, each heuristic evaluates the full training set under a 60 s wall-clock budget and is treated as a runtime failure if it exceeds it. At test time, TSP heuristics run under a per-instance wall-clock budget of 15 s at $n=50$, 30 s at $n=100$, and 60 s at $n=200$, KP heuristics under 10 s at every size, and a heuristic exceeding its budget is recorded as failing those instances.

Top- K snapshots of the methods. Every method ends a run with a saved set of heuristics carrying its own ranking. *Top-1* evaluates the member of that set with the best training fitness, identically for every method, and *top- K* routes each test instance to the lowest cost among K of its members; for OBP, $K=4$ matches the four training instances. For EPOCH and EoH-S, whose saved sets are ranked for coverage, the ensemble takes

Problem	Train		Test	
	Scale	#Inst.	Scale	#Inst.
TSP	$n=50$	64	$n \in \{20, 50, 100, 200\}$	300
OBP	$\{1k, 5k\}, C=100$	4	$\{1k, 5k, 10k\} \times \{100, 500\}$	5
KP	$n=100$	64	$n \in \{50, 100, 200\}$	1,000

Table 4: Training and test datasets. ID scales are underlined; the rest are OOD. #Inst. is per size (TSP, KP) or per (items, capacity) set (OBP). TSP gaps are to LKH-3 optima, OBP gaps to the L_1 lower bound, and KP gaps to the optimum.

the first K of that ranking, and for every other method it takes the K with the best training fitness. At K equal to the full set size the two rules coincide. Top- K thus scores the set each method itself returns, with no re-selection at test time; it is the deployment a set-based AHD method targets, and top-1 gives the single-heuristic view alongside. Since EoH-S is scored by the same procedure on its own coverage-ranked set, the top- K margin of EPOCH over it in Tables 5 and 6 reflects the search rather than the selection of the deployed set. EPOCH deploys its expert-heuristic ensemble ranked by expert count, Eq. (5), with the training-fitness-best heuristic always included. EoH-S saves its final complementary set in its own order, the best individual first, followed by the members added for coverage. CALM saves its all-time-best buffer ranked by training fitness. For HSEvo and ReEvo we take the ten best heuristics evaluated during each run, ranked by training fitness, so that strong candidates displaced by generational replacement are retained. MCTS-AHD saves the ten tree nodes with the highest back-propagated Q value, the quantity its tree search ranks nodes by. On OBP every method, including HSEvo, is trained on the same four-instance Weibull set, and we omit the Worst-Fit rule as it is orders of magnitude from competitive. Among methods that fine-tune the generator, we omit EvoTune (Šurina et al., 2025): CALM outperforms it under matched comparisons that include a DPO-trained CALM variant against the DPO-based EvoTune (Huang et al., 2026), and within our framework the scalar-fitness and reflective-learning ablations of Appendix G isolate, under a matched budget, the gains of the coverage-aware search and of the reflective preference construction over preference optimization on scalar fitness alone. The classical heuristics are deterministic single programs, so their top-1 and top- K scores coincide.

Algorithm 2 SELECTPARENTS: expert-aware parent selection

Input: pool $\mathcal{P}$, population size N_p , operator o , probabilities $p_{\text{exp}}, p_{\text{comp}}$, temperature τ
Output: parent set S

- 1: $\mathcal{P}_{N_p} \leftarrow \text{Top}_{N_p}(\mathcal{P}; \bar{\gamma}(\cdot) \uparrow)$ $\triangleright$ reproduction population
- 2: $\backslash\backslash$ *Single-parent mutation operators*
- 3: **if** o is a single-parent mutation operator **then**
- 4: **if** $\text{rand}() < p_{\text{exp}}$ **and** some h has $e(h; \mathcal{P}_{N_p}) > 0$ **then**
- 5: sample $h_1 \in \mathcal{P}_{N_p}$ with prob. $\propto 1/\text{rank}$, descending $e(\cdot; \mathcal{P}_{N_p})$ $\triangleright$ target under-covered instances
- 6: **else**
- 7: sample $h_1 \in \mathcal{P}_{N_p}$ with prob. $\propto 1/\text{rank}$, ascending $\bar{\gamma}(\cdot)$ $\triangleright$ refine strong
- 8: **end if**
- 9: **return** $\{h_1\}$
- 10: **end if**
- 11: $\backslash\backslash$ *Crossover*
- 12: sample $h_1 \in \mathcal{P}_{N_p}$ with prob. $\propto 1/\text{rank}$, ascending $\bar{\gamma}(\cdot)$ $\triangleright$ parent 1
- 13: **if** $\text{rand}() < p_{\text{comp}}$ **then**
- 14: sample $h_2 \sim \text{softmax}(d(h_1, \cdot)/\tau)$ over $\mathcal{P}_{N_p} \setminus \{h_1\}$ $\triangleright$ complementary variant
- 15: **else**
- 16: sample $h_2 \in \mathcal{P}_{N_p}$ with prob. $\propto 1/\text{rank}$, ascending $\bar{\gamma}(\cdot)$, $h_2 \neq h_1$ $\triangleright$ exploitative variant
- 17: **end if**
- 18: **return** $\{h_1, h_2\}$

C Algorithm Pseudocodes

This section provides the complete pseudocode of the subalgorithms invoked by Algorithm 1. Algorithm 2 is the expert-aware parent selector that supplies operator parents from the reproduction population. Algorithms 3 and 4 are the operator bandit’s draw and update steps. Algorithm 5 is the expert-preserving recovery that restores diversity when the coverage gap stagnates, and Algorithm 6 is the preference-pair construction that turns one iteration’s generations into training pairs.

Parent selection. Algorithm 2 mixes exploitation with coverage over the reproduction population $\mathcal{P}_{N_p}$, the N_p heuristics of the pool with the lowest average gap. A single-parent mutation operator samples either by expert count, which recruits heuristics for under-covered instances, or by average gap, which refines strong all-rounders. The crossover operator samples its first parent by average gap and its second parent either by complementarity or by average gap, selecting the matching crossover variant of Section 3.3.

Operator bandit. Algorithm 3 draws operators in proportion to a bounded weight that is centered on the mean quality across eligible operators, under

Algorithm 3 DRAWOPERATOR: bandit operator draw

Input: pool $\mathcal{P}$, quality $\{Q_o\}$, priors $\{w_o^0\}$, steepness κ , floor ϕ
Output: sampled operator o

- 1: $\mathcal{O}' \leftarrow \mathcal{O}$
- 2: **if** $|\mathcal{P}| < 2$ **then**
- 3: remove crossover from $\mathcal{O}'$ $\triangleright$ needs two parents
- 4: **end if**
- 5: $\bar{Q} \leftarrow \text{mean}_{o \in \mathcal{O}'} Q_o$
- 6: $\omega_o \leftarrow w_o^0 \text{clip}(e^{\kappa(Q_o - \bar{Q})}, \frac{1}{2}, 2)$ for $o \in \mathcal{O}'$ $\triangleright \omega_o \in [\frac{1}{2}w_o^0, 2w_o^0]$
- 7: $p_o \leftarrow \phi + (1 - |\mathcal{O}'|\phi) \omega_o / \sum_{o' \in \mathcal{O}'} \omega_{o'}$ $\triangleright$ probability floor
- 8: **return** $o \sim p$

Algorithm 4 UPDATEBANDIT: bandit quality update

Input: operator o , Round-1/Round-2 rollouts Y_1, Y_2 , best-gap snapshot F , margin ϵ_ρ , rate η , weights w_e, w_p, w_v
Output: updated quality Q_o

- 1: $V \leftarrow \{y \in Y_1 \cup Y_2 : y \text{ valid}\}$
- 2: $g_o \leftarrow \frac{1}{m} |\{i : \min_{y \in V} \gamma_i(h_y) < F_i - \epsilon_\rho\}|$ $\triangleright$ instances won vs. F
- 3: $\nu_p \leftarrow \frac{1}{|V|} |\{y \in V : \bar{\gamma}(h_y) < \bar{\gamma}_{\text{par}}\}|$ $\triangleright \bar{\gamma}_{\text{par}}$: best parent’s average gap
- 4: $\nu_v \leftarrow |V| / |Y_1 \cup Y_2|$ $\triangleright$ valid fraction
- 5: $R_o \leftarrow \text{clip}(w_e g_o + w_p \nu_p + w_v \nu_v, 0, 1)$
- 6: $Q_o \leftarrow (1 - \eta)Q_o + \eta R_o$ $\triangleright$ EMA; $Q_o \leftarrow 0$ on recovery
- 7: **return** Q_o

the probability floor ϕ . Algorithm 4 then revises the quality Q_o from the expert-coverage gain g_o together with the parent-beating and validity fractions of the iteration’s children, and resets Q_o to 0 on recovery.

Expert-preserving recovery. Algorithm 5 tracks the coverage gap $\rho(\mathcal{P})$; after T_ρ iterations without an ϵ_ρ improvement, or T_{cap} iterations since the last recovery, it rebuilds a pool of $N_p/2$ heuristics from the best heuristic, the seed, and the highest-expert-count heuristics, resets the bandit’s quality estimates, and restarts the stagnation window by setting ρ^* to the recovered pool’s coverage gap. The persistent expert archive of Section 3.5 is updated each iteration outside this procedure and remains intact across recoveries.

Preference-pair construction. Algorithm 6 forms within-batch pairs by splitting valid responses at the reward median and matching across the halves, pairs each invalid response as dispreferred against a randomly chosen valid one, and adds cross-round pairs that prefer a hinted Round-2 response over a strictly worse unhinted Round-1 response. Every pair is conditioned on the unhinted

Algorithm 5 RECOVER: expert-preserving recovery

Input: pool $\mathcal{P}$, best gap ρ^* , age a , patience T_p , iterations-since-recovery c , cap T_{cap} , margin ϵ_ρ , population size N_p , seed heuristic h_{seed}

Output: updated pool

- 1: $\backslash\backslash$ *Stagnation criterion*
- 2: $\rho \leftarrow \frac{1}{m} \sum_i \min_{h \in \mathcal{P}} \gamma_i(h)$ $\triangleright$ coverage gap, Eq. (6)
- 3: $c \leftarrow c + 1$
- 4: **if** $\rho < \rho^* - \epsilon_\rho$ **then**
- 5: $\rho^* \leftarrow \rho$; $a \leftarrow 0$ $\triangleright$ coverage gap improved
- 6: **else**
- 7: $a \leftarrow a + 1$
- 8: **end if**
- 9: **if** $a < T_p$ **and** $c < T_{\text{cap}}$ **then**
- 10: **return** $\mathcal{P}$ $\triangleright$ within patience and cap
- 11: **end if**
- 12: $\backslash\backslash$ *Expert-preserving rebuild*
- 13: $\mathcal{P}_{\text{rec}} \leftarrow \{\arg \min_{h \in \mathcal{P}} \bar{\gamma}(h)\} \cup \{h_{\text{seed}}\}$ $\triangleright$ best-by-average-gap and seed
- 14: $\mathcal{P}_{\text{rec}} \leftarrow \mathcal{P}_{\text{rec}} \cup \text{Top}_{N_p/2 - |\mathcal{P}_{\text{rec}}|}(\mathcal{P} \setminus \mathcal{P}_{\text{rec}}; e(\cdot; \mathcal{P}) \downarrow)$
 $\triangleright$ ties: lower $\bar{\gamma}$
- 15: reset bandit; $\rho^* \leftarrow \frac{1}{m} \sum_i \min_{h \in \mathcal{P}_{\text{rec}}} \gamma_i(h)$; $a \leftarrow 0$; $c \leftarrow 0$
- 16: **return** $\mathcal{P}_{\text{rec}}$

prompt q .

D Reflective Self-Distillation: Derivation and Margin Analysis

This appendix expands Section 3.7. We make precise the sense in which preference optimization over the search buffer can *rank* the reflective winners without making the policy more likely to *generate* them, and then derive the RSD term as the sampled, stop-gradient form of a divergence between the hinted and unhinted policies. Throughout, q is an unhinted operator prompt, $\tilde{q} = q \oplus z$ its insight-augmented counterpart (Section 3.6), and a reflective pair $(q, y^+, y^-) \in \mathcal{B}_{\text{ref}}$ has a hinted winner $y^+ \sim \pi_\theta(\cdot | \tilde{q})$ and a strictly worse unhinted candidate $y^- \sim \pi_\theta(\cdot | q)$.

Ranking is not generation. Recall the implicit reward $\hat{r}_\theta(y) = \beta \log \frac{\pi_\theta(y|q)}{\pi_{\text{ref}}(y|q)}$. The per-pair loss of Eq. (2) is a decreasing function of the margin

$$\begin{aligned} \Delta_\theta(q, y^+, y^-) &= \hat{r}_\theta(y^+) - \hat{r}_\theta(y^-) \\ &= \beta \left[\log \frac{\pi_\theta(y^+|q)}{\pi_{\text{ref}}(y^+|q)} - \log \frac{\pi_\theta(y^-|q)}{\pi_{\text{ref}}(y^-|q)} \right] \end{aligned} \quad (15)$$

and enters the pair only through Δ_θ , so nothing in it rewards a high *absolute* value of $\log \pi_\theta(y^+ | q)$; a given increase of Δ_θ is credited equally whether it raises $\log \pi_\theta(y^+ | q)$ or lowers $\log \pi_\theta(y^- | q)$. Because a log-probability is bounded above by 0 but unbounded below, the margin can be widened far more cheaply by driving $\log \pi_\theta(y^- | q)$

Algorithm 6 BUILDPAIRS: preference-pair construction

Input: unhinted prompt q , Round-1 rollouts Y_1 , Round-2 rollouts Y_2

Output: preference pairs $\mathcal{B}_q$

- 1: $V \leftarrow$ valid responses in $Y_1 \cup Y_2$; $I \leftarrow$ invalid responses; $\mathcal{B}_q \leftarrow \emptyset$
- 2: $\backslash\backslash$ *Within-batch pairs*
- 3: $(V^+, V^-) \leftarrow$ higher- and lower-scoring halves of V by $r(\cdot)$
- 4: add (q, y^+, y^-) for a random matching of V^+ with V^- $\triangleright$ each response used at most once
- 5: **for all** $y^- \in I$ **do**
- 6: add (q, y, y^-) with $y \sim \text{Unif}(V)$ $\triangleright$ valid $\succ$ invalid
- 7: **end for**
- 8: $\backslash\backslash$ *Cross-round pairs*
- 9: **for all** valid $y^+ \in Y_2$ **do**
- 10: add (q, y^+, y^-) with $y^- \sim \text{Unif}(\{y \in V \cap Y_1 : r(y) < r(y^+)\})$ $\triangleright$ skip if empty
- 11: **end for**
- 12: **return** $\mathcal{B}_q$

down than by lifting the already-high $\log \pi_\theta(y^+ | q)$, and gradient descent does so preferentially, leaving $\log \pi_\theta(y^+ | q)$ near its reference value $\log \pi_{\text{ref}}(y^+ | q)$. Two consequences specialize this to our setting. First, the reflective pairs are a minority of the buffer and have a *small* margin by construction: y^+ and y^- are both valid heuristics that differ only by the hint’s refinement, so $|\Delta_\theta|$ is small relative to the within-batch and valid-versus-invalid pairs, whose rejected members are far weaker, and the reflective signal is easily dominated. Second, even when the reflective margin is realized it need not raise $\log \pi_\theta(y^+ | q)$, the quantity that governs how likely the *unhinted* policy is to generate y^+ in the next iteration. Ranking y^+ above y^- and generating y^+ from q are thus distinct objectives, and Eq. (2) targets only the former.

The hinted–unhinted divergence. The objective $\mathcal{L}_{\text{RSD}}$ of Eq. (13) is the forward Kullback–Leibler divergence between the hinted policy $\pi_\theta(\cdot | \tilde{q})$, which the insight z biases toward better heuristics, and the unhinted policy $\pi_\theta(\cdot | q)$ the search rolls out from. It vanishes when the hint changes nothing and grows as the hint moves the policy, so minimizing it pulls the unhinted policy toward the hinted one and the reflective improvement carries into the policy’s own Round-1 rollouts.

From the divergence to a likelihood. The divergence averages over the open-ended space of heuristics and has no closed form, so we estimate

it by sampling. Expanding it,

$$\begin{aligned} & \text{KL}(\pi_\theta(\cdot \mid \tilde{q}) \parallel \pi_\theta(\cdot \mid q)) \\ &= \mathbb{E}_{y \sim \pi_\theta(\cdot \mid \tilde{q})} [\log \pi_\theta(y \mid \tilde{q}) \\ & \quad - \log \pi_\theta(y \mid q)]. \end{aligned} \quad (16)$$

Two properties of our pipeline reduce this to a single likelihood. First, the Round-2 winner y^+ is a draw from $\pi_\theta(\cdot \mid \tilde{q})$, the sampling distribution of Eq. (16), so it serves as a Monte Carlo sample at no extra cost; because $\mathcal{B}_{\text{ref}}$ retains only winners whose gap certifies a reflective gain, the resulting term is a reward-filtered surrogate of the divergence, deliberately biased toward the region the hint improved. Second, treating the hinted policy as a fixed teacher, a stop-gradient on $\pi_\theta(\cdot \mid \tilde{q})$ that is standard in distillation, makes $\log \pi_\theta(y^+ \mid \tilde{q})$ a constant in θ that drops from the gradient. What remains is

$$\nabla_\theta \hat{\mathcal{L}}_{\text{RSD}} = -\nabla_\theta \log \pi_\theta(y^+ \mid q), \quad (17)$$

the gradient of the negative log-likelihood of the reflective winner under the unhinted prompt. The RSD term $-\lambda \log \pi_\theta(y^+ \mid q)$ of Eq. (14) is therefore not a separate objective added to DPO but the reward-filtered, sampled, stop-gradient realization of the divergence in Eq. (13), evaluated on samples the search has already produced. The same forward pass DPO uses to score y^+ already yields $\log \pi_\theta(y^+ \mid q)$, so the term adds essentially no compute.

Why this accumulates. A related distillation term has been motivated by inference-time hint independence, dropping the teacher at deployment (Zhao and Li, 2025). That reading does not apply here, since EPOCH deploys saved heuristic *artifacts* rather than the policy. Our motivation is internal to training. Every iteration draws its Round-1 rollouts from the unhinted policy $\pi_\theta(\cdot \mid q)$, so raising $\log \pi_\theta(y^+ \mid q)$ makes those rollouts more likely to recover the reflective improvement unprompted, which in turn seeds better parents, preference pairs, and experts in subsequent iterations. Because search and policy update interleave, an improvement internalized in one burst propagates through the population it generates, so the effect accumulates over the run rather than acting only at its end.

Implementation. We restrict RSD to $\mathcal{B}_{\text{ref}}$ because each of its pairs contains, by construction, a valid Round-2 winner over a strictly worse valid

Round-1 rollout, so the term distills genuine reflective gains rather than arbitrary accepted code. We length-normalize $\log \pi_\theta(y^+ \mid q)$ by the token length of y^+ so the objective does not favor short heuristics, and we keep λ small. With few reflective samples per burst, an unnormalized or large- λ term behaves like maximum-likelihood concentration onto a handful of accepted heuristics and can over-sharpen the policy, eroding the output diversity that the coverage objective depends on. Setting $\lambda = 0$ recovers the plain-DPO update of Eq. (2), which makes the contribution of RSD a single-coefficient ablation.

E Extended Experiments

E.1 Extended Experiment Results

Table 5 extends Table 1 with the greedy-edge (Bentley, 1992), nearest-insertion (Rosenkrantz et al., 1977), and Christofides (Christofides, 1976) classical heuristics and the Qwen2.5-7B backbone; Table 6 extends Table 2 with the Qwen2.5-7B backbone. Both tables also include the inference-only baselines run with the closed-source GPT-5-nano as the generator. On OBP, EPOCH attains a lower average gap than each of them on both backbones at both top-1 and top-4. On TSP, EPOCH keeps the lower top-1 gap, and only EoH-S matches its top-10 ensemble, with a lower gap at $n \in \{20, 200\}$ and a slightly higher one at $n \in \{50, 100\}$. These runs make every one of their 1,000 heuristic generations with GPT-5-nano, which EPOCH queries only as a critic, and cost $14\text{--}29\times$ more per run in API calls (Appendix H).

Table 7 reports the KP results with the Llama-3.1-8B backbone against ReEvo and MCTS-AHD, whose released implementations include this problem. EPOCH attains the best or a comparable top-1 gap at every size, its top-3 ensemble outperforms the top-10 ensemble of ReEvo at every size, and its top-10 ensemble cuts the gap by 26–47% relative to MCTS-AHD and 55–59% relative to ReEvo, with the gains persisting OOD at $n \in \{50, 200\}$.

E.2 Robustness to Seed Quality

Several LLM-based AHD methods begin their search from a fixed *seed heuristic*. ReEvo and HSEvo require one to initialize the population and to sustain variation among candidates, since a population that has converged to a single objective value cannot improve further and the search halts; CALM and EPOCH likewise start from a provided

heuristic. To keep the comparison controlled, every method that requires a seed is initialized from the same heuristic, adopted from the MCTS-AHD experimental setup. The TSP seed, shown in the box below, is structurally detailed, scoring each candidate node by its distance to the current node, the mean and dispersion of its distances to the remaining nodes, and its distance to the destination.

Standard TSP seed heuristic

```
import numpy as np

def select_next_node(current_node: int,
                    destination_node: int,
                    unvisited_nodes: set,
                    distance_matrix: np.ndarray) -> int:
    c1, c2, c3, c4 = 0.4, 0.3, 0.2, 0.1
    scores = {}
    for node in unvisited_nodes:
        all_distances = [distance_matrix[node][i]
                        for i in unvisited_nodes
                        if i != node]
        average_distance = np.mean(all_distances)
        std_dev_distance = np.std(all_distances)
        score = (
            c1 * distance_matrix[current_node][node]
            - c2 * average_distance
            + c3 * std_dev_distance
            - c4 * distance_matrix[destination_node][node]
        )
        scores[node] = score
    return min(scores, key=scores.get)
```

The OBP seed, shown in the box below, is a single priority rule shared by every seed-requiring method; it is structurally detailed yet weak, scoring below the First-Fit and Best-Fit rules of Table 6. It returns a priority score per open bin, and the framework places the item in the highest-scoring bin.

OBP seed heuristic

```
import numpy as np

def step(item_size: float,
        remaining_capacity: np.ndarray) -> np.ndarray:
    max_bin_cap = max(remaining_capacity)
    score = ((remaining_capacity - max_bin_cap) ** 2
            / item_size
            + remaining_capacity ** 2 / item_size ** 2
            + remaining_capacity ** 2 / item_size ** 3)
    over = remaining_capacity > item_size
    score[over] = -score[over]
    score[1:] -= score[:-1]
    return score
```

The KP seed, shown in the box below, packs the first remaining item that fits; it is the First-Fit row of Table 7.

KP seed heuristic

```
import numpy as np

def select_next_item(remaining_capacity: float,
                    weights: np.ndarray,
                    values: np.ndarray) -> int:
    fits = np.flatnonzero(weights <= remaining_capacity)
    return int(fits[0])
```

A capable seed narrows the margin a method must close and can reward a search that exploits the seed over one that improves the generator. To test whether the advantage of EPOCH persists from a weaker starting point, we re-run the two methods that fine-tune the generator, CALM and EPOCH, from the substantially weaker TSP seed shown in the box below, which assigns every unvisited node the same constant score and therefore selects the next node without any problem-specific preference. All other settings match the TSP protocol of Section 4.

Weak TSP seed heuristic (robustness study)

```
import numpy as np

def select_next_node(current_node: int,
                    destination_node: int,
                    unvisited_nodes: set,
                    distance_matrix: np.ndarray) -> int:
    scores = {node: 1 for node in unvisited_nodes}
    return min(scores, key=scores.get)
```

As reported in Table 8, from the weak seed CALM degrades sharply on both backbones, and its top-10 gap at the training size $n=50$ rises to 14.51% on Llama and 16.10% on Qwen, against 9.31 and 9.59% from the standard seed in Table 5. EPOCH instead stays close to its standard-seed performance, with a top-10 gap of 6.71 and 6.76% at $n=50$ against 5.73 and 5.71% in Table 5, and it attains the lowest gap on every size, metric, and backbone. Its top-3 ensemble from the weak seed still outperforms the full top-10 ensemble of CALM throughout. Because the per-instance gap-vector objective steers selection, operator allocation, recovery, and the deployed ensemble toward coverage rather than the fitness of a single heuristic, EPOCH recovers a strong complementary set even from an uninformative seed, where a fitness-only method does not.

E.3 Search Dynamics & Evolutionary Curves

To assess search efficiency and stability, Figure 4 plots the best-so-far cost on $\mathcal{D}_{\text{train}}$ against the number of heuristic evaluations, from the same weak seed and averaged over at least five runs of each method. EPOCH converges faster and to a lower cost than the LLM-based AHD baselines on both backbones, and its top-10 ensemble curve stays below every single-heuristic method across the entire budget, with low deviation across runs. This is the intended effect of optimizing per-instance coverage rather than the fitness of a single heuristic.

Method		TSP-20	TSP-50	TSP-100	TSP-200
<i>Classical heuristics</i>					
Nearest Neighbor		17.64	21.96	24.58	26.10
Greedy Edge		11.96	17.41	18.69	17.89
Nearest Insertion		13.14	19.38	21.55	23.69
Christofides		8.82	11.17	11.95	12.41
<i>LLM-based AHD: GPT-4o-mini (closed-source)</i>					
ReEvo	Top-1	10.55 $\pm$ 3.58	13.89 $\pm$ 4.48	15.43 $\pm$ 4.77	16.64 $\pm$ 5.00
	Top-3	9.18 $\pm$ 2.55	12.12 $\pm$ 2.66	13.66 $\pm$ 3.29	17.01 $\pm$ 4.28
	Top-10	8.49 $\pm$ 2.52	10.92 $\pm$ 2.29	12.48 $\pm$ 2.70	14.29 $\pm$ 2.98
HSEvo	Top-1	12.58 $\pm$ 4.18	16.37 $\pm$ 4.62	18.56 $\pm$ 4.94	20.86 $\pm$ 5.30
	Top-3	11.38 $\pm$ 5.16	15.12 $\pm$ 5.64	17.39 $\pm$ 5.91	20.14 $\pm$ 5.98
	Top-10	10.59 $\pm$ 5.81	14.28 $\pm$ 6.32	16.66 $\pm$ 6.50	19.65 $\pm$ 6.48
MCTS-AHD	Top-1	7.28 $\pm$ 0.71	10.77 $\pm$ 0.71	12.82 $\pm$ 0.67	14.75 $\pm$ 1.03
	Top-3	6.25 $\pm$ 1.07	9.50 $\pm$ 1.08	11.77 $\pm$ 1.14	14.00 $\pm$ 1.18
	Top-10	4.54 $\pm$ 1.20	7.94 $\pm$ 1.17	10.49 $\pm$ 1.25	12.85 $\pm$ 1.41
EoH-S	Top-1	8.56 $\pm$ 0.91	11.69 $\pm$ 0.71	13.80 $\pm$ 0.91	15.05 $\pm$ 0.98
	Top-3	5.05 $\pm$ 0.43	9.14 $\pm$ 0.61	11.80 $\pm$ 0.83	13.75 $\pm$ 0.85
	Top-10	3.22 $\pm$ 0.41	7.30 $\pm$ 0.38	10.06 $\pm$ 0.38	12.39 $\pm$ 0.42
<i>LLM-based AHD: GPT-5-nano (closed-source)</i>					
ReEvo	Top-1	12.32 $\pm$ 5.16	16.11 $\pm$ 5.41	18.10 $\pm$ 5.87	19.84 $\pm$ 6.08
	Top-3	12.32 $\pm$ 5.16	16.11 $\pm$ 5.41	18.10 $\pm$ 5.87	19.84 $\pm$ 6.08
	Top-10	12.32 $\pm$ 5.16	16.11 $\pm$ 5.41	18.10 $\pm$ 5.87	19.84 $\pm$ 6.08
HSEvo	Top-1	8.44 $\pm$ 1.05	11.90 $\pm$ 1.20	13.54 $\pm$ 1.07	14.73 $\pm$ 1.17
	Top-3	8.21 $\pm$ 1.08	11.64 $\pm$ 1.18	13.22 $\pm$ 1.09	14.39 $\pm$ 1.22
	Top-10	8.10 $\pm$ 1.13	11.48 $\pm$ 1.24	13.05 $\pm$ 1.17	14.24 $\pm$ 1.30
MCTS-AHD	Top-1	7.23 $\pm$ 0.39	10.24 $\pm$ 0.38	12.13 $\pm$ 0.49	13.77 $\pm$ 1.02
	Top-3	5.64 $\pm$ 1.22	8.67 $\pm$ 0.84	10.66 $\pm$ 0.85	12.51 $\pm$ 1.07
	Top-10	3.78 $\pm$ 0.30	7.11 $\pm$ 0.34	9.32 $\pm$ 0.58	11.64 $\pm$ 0.88
EoH-S	Top-1	6.47 $\pm$ 0.71	10.04 $\pm$ 0.37	12.04 $\pm$ 0.48	13.94 $\pm$ 0.86
	Top-3	4.13 $\pm$ 0.21	7.69 $\pm$ 0.35	10.07 $\pm$ 0.56	12.25 $\pm$ 0.81
	Top-10	2.48 $\pm$ 0.15	5.78 $\pm$ 0.22	8.23 $\pm$ 0.33	10.46 $\pm$ 0.33
<i>LLM-based AHD: Qwen2.5-7B</i>					
ReEvo	Top-1	7.83 $\pm$ 2.07	11.10 $\pm$ 2.02	12.99 $\pm$ 2.10	14.43 $\pm$ 2.40
	Top-3	7.25 $\pm$ 1.55	10.48 $\pm$ 1.59	12.32 $\pm$ 1.64	13.81 $\pm$ 2.06
	Top-10	7.23 $\pm$ 1.58	10.47 $\pm$ 1.60	12.30 $\pm$ 1.66	13.78 $\pm$ 2.11
HSEvo	Top-1	12.27 $\pm$ 3.27	16.41 $\pm$ 3.78	19.17 $\pm$ 3.78	22.50 $\pm$ 3.60
	Top-3	12.12 $\pm$ 3.50	16.30 $\pm$ 3.92	19.08 $\pm$ 3.88	22.50 $\pm$ 3.60
	Top-10	11.16 $\pm$ 3.00	15.42 $\pm$ 3.58	18.22 $\pm$ 3.55	21.53 $\pm$ 3.14
MCTS-AHD	Top-1	8.42 $\pm$ 0.66	12.00 $\pm$ 0.16	14.25 $\pm$ 0.49	15.93 $\pm$ 0.89
	Top-3	7.83 $\pm$ 0.58	11.34 $\pm$ 0.54	13.63 $\pm$ 0.92	15.19 $\pm$ 1.32
	Top-10	6.04 $\pm$ 1.40	9.75 $\pm$ 1.32	12.18 $\pm$ 1.29	14.07 $\pm$ 1.31
EoH-S	Top-1	8.64 $\pm$ 1.03	12.25 $\pm$ 1.26	14.44 $\pm$ 1.84	16.73 $\pm$ 2.50
	Top-3	5.43 $\pm$ 1.14	9.12 $\pm$ 1.05	11.76 $\pm$ 1.45	14.43 $\pm$ 2.37
	Top-10	3.46 $\pm$ 0.66	7.08 $\pm$ 0.76	9.97 $\pm$ 1.01	12.69 $\pm$ 1.34
CALM	Top-1	6.75 $\pm$ 0.08	9.99 $\pm$ 0.22	11.96 $\pm$ 0.20	13.96 $\pm$ 0.20
	Top-3	6.70 $\pm$ 0.14	9.92 $\pm$ 0.23	11.90 $\pm$ 0.21	13.91 $\pm$ 0.27
	Top-10	6.30 $\pm$ 0.50	9.59 $\pm$ 0.55	11.59 $\pm$ 0.42	13.77 $\pm$ 0.34
EPOCH (Ours)	Top-1	6.42 $\pm$ 0.08	9.81 $\pm$ 0.22	11.70 $\pm$ 0.46	13.67 $\pm$ 0.91
	Top-3	4.39 $\pm$ 0.20	7.62 $\pm$ 0.44	10.06 $\pm$ 0.45	12.25 $\pm$ 0.50
	Top-10	2.91 $\pm$ 0.14	5.71 $\pm$ 0.31	8.22 $\pm$ 0.37	10.93 $\pm$ 0.47
<i>LLM-based AHD: Llama-3.1-8B</i>					
ReEvo	Top-1	10.58 $\pm$ 0.58	14.87 $\pm$ 1.36	18.38 $\pm$ 2.10	26.06 $\pm$ 12.93
	Top-3	10.28 $\pm$ 0.59	14.15 $\pm$ 2.15	16.19 $\pm$ 1.65	19.05 $\pm$ 1.37
	Top-10	10.25 $\pm$ 0.62	14.10 $\pm$ 2.24	15.96 $\pm$ 2.01	18.69 $\pm$ 1.54
HSEvo	Top-1	11.46 $\pm$ 1.69	15.85 $\pm$ 2.63	18.48 $\pm$ 4.51	21.34 $\pm$ 7.00
	Top-3	10.61 $\pm$ 1.28	14.83 $\pm$ 1.76	17.27 $\pm$ 2.82	19.47 $\pm$ 3.89
	Top-10	10.29 $\pm$ 1.54	14.45 $\pm$ 2.10	16.80 $\pm$ 3.24	18.99 $\pm$ 4.31
MCTS-AHD	Top-1	10.27 $\pm$ 0.37	14.23 $\pm$ 0.65	16.39 $\pm$ 1.48	17.87 $\pm$ 1.45
	Top-3	9.76 $\pm$ 0.40	13.40 $\pm$ 0.39	15.40 $\pm$ 0.81	16.92 $\pm$ 1.04
	Top-10	7.58 $\pm$ 0.63	11.73 $\pm$ 0.61	13.77 $\pm$ 0.91	15.51 $\pm$ 1.15
EoH-S	Top-1	10.69 $\pm$ 0.93	15.43 $\pm$ 1.58	17.86 $\pm$ 1.38	20.04 $\pm$ 1.46
	Top-3	6.05 $\pm$ 0.72	11.68 $\pm$ 1.06	14.59 $\pm$ 0.93	17.31 $\pm$ 1.05
	Top-10	4.70 $\pm$ 0.81	10.46 $\pm$ 1.02	13.60 $\pm$ 0.96	16.45 $\pm$ 1.07
CALM	Top-1	6.76 $\pm$ 0.26	10.02 $\pm$ 0.26	11.75 $\pm$ 0.27	13.47 $\pm$ 0.31
	Top-3	6.55 $\pm$ 0.23	9.68 $\pm$ 0.55	11.34 $\pm$ 0.51	13.33 $\pm$ 0.47
	Top-10	6.19 $\pm$ 0.54	9.31 $\pm$ 0.80	11.04 $\pm$ 0.67	13.30 $\pm$ 0.91
EPOCH (Ours)	Top-1	6.19 $\pm$ 0.30	9.61 $\pm$ 0.25	11.63 $\pm$ 0.21	13.57 $\pm$ 0.36
	Top-3	4.31 $\pm$ 0.30	7.62 $\pm$ 0.23	9.85 $\pm$ 0.36	11.98 $\pm$ 0.49
	Top-10	2.81 $\pm$ 0.22	5.73 $\pm$ 0.15	8.15 $\pm$ 0.19	10.74 $\pm$ 0.38

Table 5: Complete version of Table 1. TSP optimality gap (%) to LKH-3 optima; each cell is the mean over runs with standard deviation, lower is better. Size $n=50$ is ID (underlined); $n \in \{20, 100, 200\}$ are OOD. Within each backbone block the best method per metric and test set is shaded; each test set’s overall best result is shown in bold.

Method		1k_100	1k_500	5k_100	5k_500	10k_100	10k_500	Avg.
<i>Classical heuristics</i>								
First-Fit		5.02	0.25	4.65	0.55	4.36	0.50	2.55
Best-Fit		4.77	0.25	4.31	0.55	4.05	0.47	2.40
<i>LLM-based AHD: GPT-4o-mini (closed-source)</i>								
ReEvo	Top-1	3.74 $\pm$ 0.94	0.40 $\pm$ 0.30	3.09 $\pm$ 1.12	0.69 $\pm$ 0.30	2.84 $\pm$ 1.11	0.53 $\pm$ 0.12	1.88 $\pm$ 0.46
	Top-4	3.61 $\pm$ 1.01	0.40 $\pm$ 0.30	2.98 $\pm$ 1.21	0.56 $\pm$ 0.13	2.74 $\pm$ 1.22	0.44 $\pm$ 0.03	1.79 $\pm$ 0.53
HSEvo	Top-1	2.85 $\pm$ 1.01	0.40 $\pm$ 0.30	2.19 $\pm$ 1.15	0.56 $\pm$ 0.02	2.01 $\pm$ 1.15	0.50 $\pm$ 0.04	1.42 $\pm$ 0.52
	Top-4	2.82 $\pm$ 1.01	0.20 $\pm$ 0.10	2.17 $\pm$ 1.17	0.47 $\pm$ 0.12	2.01 $\pm$ 1.15	0.42 $\pm$ 0.10	1.35 $\pm$ 0.56
MCTS-AHD	Top-1	2.59 $\pm$ 0.45	0.15 $\pm$ 0.12	1.88 $\pm$ 0.47	0.42 $\pm$ 0.17	1.71 $\pm$ 0.41	0.38 $\pm$ 0.12	1.19 $\pm$ 0.27
	Top-4	2.48 $\pm$ 0.51	0.10 $\pm$ 0.12	1.79 $\pm$ 0.53	0.36 $\pm$ 0.17	1.61 $\pm$ 0.48	0.33 $\pm$ 0.13	1.11 $\pm$ 0.31
EoH-S	Top-1	2.62 $\pm$ 0.33	0.25 $\pm$ 0.16	1.74 $\pm$ 0.73	0.42 $\pm$ 0.15	1.46 $\pm$ 0.83	0.35 $\pm$ 0.13	1.14 $\pm$ 0.33
	Top-4	2.40 $\pm$ 0.38	0.15 $\pm$ 0.12	1.41 $\pm$ 0.56	0.24 $\pm$ 0.16	1.08 $\pm$ 0.58	0.18 $\pm$ 0.15	0.91 $\pm$ 0.30
<i>LLM-based AHD: GPT-5-nano (closed-source)</i>								
ReEvo	Top-1	3.58 $\pm$ 1.24	0.50 $\pm$ 0.25	2.91 $\pm$ 1.32	0.62 $\pm$ 0.07	2.73 $\pm$ 1.37	0.54 $\pm$ 0.06	1.81 $\pm$ 0.72
	Top-4	3.58 $\pm$ 1.24	0.37 $\pm$ 0.13	2.90 $\pm$ 1.31	0.57 $\pm$ 0.02	2.72 $\pm$ 1.35	0.47 $\pm$ 0.00	1.77 $\pm$ 0.68
HSEvo	Top-1	4.56 $\pm$ 0.15	0.25 $\pm$ 0.00	4.04 $\pm$ 0.22	0.55 $\pm$ 0.00	3.89 $\pm$ 0.13	0.47 $\pm$ 0.01	2.29 $\pm$ 0.08
	Top-4	4.56 $\pm$ 0.15	0.25 $\pm$ 0.00	4.04 $\pm$ 0.22	0.55 $\pm$ 0.00	3.88 $\pm$ 0.13	0.47 $\pm$ 0.01	2.29 $\pm$ 0.08
MCTS-AHD	Top-1	3.78 $\pm$ 1.15	0.25 $\pm$ 0.00	3.06 $\pm$ 1.44	0.55 $\pm$ 0.00	2.81 $\pm$ 1.46	0.48 $\pm$ 0.02	1.82 $\pm$ 0.67
	Top-4	3.73 $\pm$ 1.19	0.20 $\pm$ 0.10	3.04 $\pm$ 1.43	0.48 $\pm$ 0.09	2.79 $\pm$ 1.45	0.43 $\pm$ 0.08	1.78 $\pm$ 0.71
EoH-S	Top-1	3.22 $\pm$ 0.83	0.53 $\pm$ 0.81	1.84 $\pm$ 0.83	0.48 $\pm$ 0.18	1.53 $\pm$ 0.86	0.41 $\pm$ 0.09	1.33 $\pm$ 0.51
	Top-4	2.84 $\pm$ 0.85	0.28 $\pm$ 0.50	1.39 $\pm$ 0.70	0.36 $\pm$ 0.17	0.94 $\pm$ 0.70	0.31 $\pm$ 0.16	1.02 $\pm$ 0.42
<i>LLM-based AHD: Qwen2.5-7B</i>								
ReEvo	Top-1	4.53 $\pm$ 0.68	0.25 $\pm$ 0.00	3.29 $\pm$ 1.29	0.49 $\pm$ 0.12	3.00 $\pm$ 1.37	0.44 $\pm$ 0.13	2.00 $\pm$ 0.55
	Top-4	4.47 $\pm$ 0.65	0.20 $\pm$ 0.10	3.25 $\pm$ 1.26	0.48 $\pm$ 0.12	2.98 $\pm$ 1.35	0.41 $\pm$ 0.12	1.97 $\pm$ 0.52
HSEvo	Top-1	4.72 $\pm$ 0.10	0.25 $\pm$ 0.00	4.24 $\pm$ 0.10	0.55 $\pm$ 0.00	4.02 $\pm$ 0.05	0.47 $\pm$ 0.00	2.38 $\pm$ 0.04
	Top-4	4.71 $\pm$ 0.12	0.25 $\pm$ 0.00	4.24 $\pm$ 0.10	0.55 $\pm$ 0.00	4.02 $\pm$ 0.06	0.47 $\pm$ 0.00	2.37 $\pm$ 0.05
MCTS-AHD	Top-1	4.71 $\pm$ 0.14	0.25 $\pm$ 0.00	4.13 $\pm$ 0.21	0.55 $\pm$ 0.00	3.95 $\pm$ 0.22	0.47 $\pm$ 0.00	2.34 $\pm$ 0.09
	Top-4	4.64 $\pm$ 0.08	0.25 $\pm$ 0.00	4.12 $\pm$ 0.21	0.55 $\pm$ 0.00	3.93 $\pm$ 0.21	0.47 $\pm$ 0.00	2.33 $\pm$ 0.08
EoH-S	Top-1	3.70 $\pm$ 1.10	0.49 $\pm$ 0.29	2.89 $\pm$ 1.21	0.52 $\pm$ 0.12	2.71 $\pm$ 1.21	0.44 $\pm$ 0.12	1.79 $\pm$ 0.57
	Top-4	3.37 $\pm$ 1.10	0.25 $\pm$ 0.25	2.31 $\pm$ 1.35	0.39 $\pm$ 0.12	2.04 $\pm$ 1.43	0.39 $\pm$ 0.12	1.46 $\pm$ 0.64
CALM	Top-1	4.17 $\pm$ 0.54	0.35 $\pm$ 0.12	2.89 $\pm$ 1.18	0.45 $\pm$ 0.15	2.55 $\pm$ 1.33	0.35 $\pm$ 0.15	1.79 $\pm$ 0.51
	Top-4	4.09 $\pm$ 0.55	0.30 $\pm$ 0.10	2.82 $\pm$ 1.28	0.45 $\pm$ 0.15	2.50 $\pm$ 1.41	0.35 $\pm$ 0.16	1.75 $\pm$ 0.56
EPOCH (Ours)	Top-1	3.91 $\pm$ 0.66	0.25 $\pm$ 0.22	1.54 $\pm$ 0.46	0.17 $\pm$ 0.09	1.27 $\pm$ 0.66	0.19 $\pm$ 0.14	1.22 $\pm$ 0.20
	Top-4	3.19 $\pm$ 0.10	0.15 $\pm$ 0.20	0.79 $\pm$ 0.03	0.17 $\pm$ 0.09	0.30 $\pm$ 0.02	0.15 $\pm$ 0.07	0.79 $\pm$ 0.05
<i>LLM-based AHD: Llama-3.1-8B</i>								
ReEvo	Top-1	4.34 $\pm$ 0.70	0.25 $\pm$ 0.00	3.27 $\pm$ 1.25	0.41 $\pm$ 0.17	3.04 $\pm$ 1.36	0.36 $\pm$ 0.16	1.95 $\pm$ 0.60
	Top-4	4.10 $\pm$ 0.87	0.20 $\pm$ 0.10	3.23 $\pm$ 1.24	0.41 $\pm$ 0.17	2.98 $\pm$ 1.34	0.35 $\pm$ 0.16	1.88 $\pm$ 0.64
HSEvo	Top-1	3.39 $\pm$ 0.93	0.25 $\pm$ 0.00	2.42 $\pm$ 1.26	0.42 $\pm$ 0.18	2.11 $\pm$ 1.33	0.36 $\pm$ 0.15	1.49 $\pm$ 0.58
	Top-4	3.36 $\pm$ 0.97	0.25 $\pm$ 0.00	2.42 $\pm$ 1.26	0.40 $\pm$ 0.16	2.10 $\pm$ 1.33	0.36 $\pm$ 0.15	1.48 $\pm$ 0.59
MCTS-AHD	Top-1	4.14 $\pm$ 0.89	0.17 $\pm$ 0.12	3.54 $\pm$ 1.09	0.46 $\pm$ 0.12	3.30 $\pm$ 1.07	0.40 $\pm$ 0.11	2.00 $\pm$ 0.57
	Top-4	4.11 $\pm$ 0.94	0.17 $\pm$ 0.12	3.54 $\pm$ 1.09	0.46 $\pm$ 0.12	3.30 $\pm$ 1.07	0.40 $\pm$ 0.11	1.99 $\pm$ 0.57
EoH-S	Top-1	3.92 $\pm$ 0.78	0.21 $\pm$ 0.09	2.89 $\pm$ 0.87	0.51 $\pm$ 0.07	2.59 $\pm$ 1.00	0.45 $\pm$ 0.05	1.76 $\pm$ 0.38
	Top-4	3.56 $\pm$ 0.73	0.08 $\pm$ 0.12	1.46 $\pm$ 0.57	0.16 $\pm$ 0.09	1.14 $\pm$ 0.69	0.14 $\pm$ 0.10	1.09 $\pm$ 0.27
CALM	Top-1	3.71 $\pm$ 0.52	0.59 $\pm$ 0.82	1.32 $\pm$ 0.40	0.30 $\pm$ 0.14	0.82 $\pm$ 0.41	0.27 $\pm$ 0.14	1.17 $\pm$ 0.22
	Top-4	3.69 $\pm$ 0.51	0.59 $\pm$ 0.82	1.29 $\pm$ 0.43	0.29 $\pm$ 0.15	0.80 $\pm$ 0.43	0.26 $\pm$ 0.13	1.15 $\pm$ 0.23
EPOCH (Ours)	Top-1	2.99 $\pm$ 0.70	0.30 $\pm$ 0.36	1.66 $\pm$ 0.90	0.32 $\pm$ 0.21	1.33 $\pm$ 0.97	0.28 $\pm$ 0.24	1.15 $\pm$ 0.39
	Top-4	2.34 $\pm$ 0.43	0.10 $\pm$ 0.12	0.74 $\pm$ 0.06	0.20 $\pm$ 0.07	0.36 $\pm$ 0.06	0.13 $\pm$ 0.04	0.64 $\pm$ 0.07

Table 6: Complete version of Table 2. OBP optimality gap (%) to the L_1 lower bound on the six held-out Weibull test sets; each cell is the mean over runs with standard deviation, lower is better. Columns are (items, capacity), e.g. 1k_100 is 1,000 items at capacity 100; Avg. is the mean over the six. Item scales 1k, 5k are ID (underlined); 10k is OOD. Within each backbone block the best Avg. per metric is shaded; each test set’s overall best result is shown in bold.

Method		KP-50	<u>KP-100</u>	KP-200
<i>Classical heuristics</i>				
First-Fit (seed)		32.856	34.836	53.153
<i>LLM-based AHD: Llama-3.1-8B</i>				
ReEvo	Top-1	0.224 $\pm$ 0.031	0.099 $\pm$ 0.016	0.107 $\pm$ 0.025
	Top-3	0.198 $\pm$ 0.018	0.081 $\pm$ 0.006	0.089 $\pm$ 0.023
	Top-10	0.170 $\pm$ 0.022	0.066 $\pm$ 0.005	0.071 $\pm$ 0.018
MCTS-AHD	Top-1	0.202 $\pm$ 0.013	0.090 $\pm$ 0.009	0.077 $\pm$ 0.010
	Top-3	0.187 $\pm$ 0.021	0.080 $\pm$ 0.016	0.063 $\pm$ 0.017
	Top-10	0.131 $\pm$ 0.038	0.054 $\pm$ 0.012	0.039 $\pm$ 0.006
EPOCH (Ours)	Top-1	0.188 $\pm$ 0.008	0.089 $\pm$ 0.006	0.078 $\pm$ 0.010
	Top-3	0.123 $\pm$ 0.027	0.057 $\pm$ 0.007	0.059 $\pm$ 0.015
	Top-10	0.070 $\pm$ 0.018	0.030 $\pm$ 0.005	0.029 $\pm$ 0.006

Table 7: KP optimality gap (%) to the optimum with the Llama-3.1-8B backbone; each cell is the mean over runs with standard deviation, lower is better. Size $n=100$ is ID (underlined); $n \in \{50, 200\}$ are OOD. The best method per metric and test set is **shaded**; each test set’s overall best result is shown in **bold**.

Method		TSP-20	<u>TSP-50</u>	TSP-100	TSP-200
<i>LLM-based AHD: Qwen2.5-7B</i>					
CALM	Top-1	12.08 $\pm$ 3.56	16.70 $\pm$ 3.89	19.38 $\pm$ 4.21	21.39 $\pm$ 3.56
	Top-3	12.08 $\pm$ 3.57	16.65 $\pm$ 3.89	19.32 $\pm$ 4.21	21.21 $\pm$ 3.56
	Top-10	11.49 $\pm$ 3.56	16.10 $\pm$ 3.72	18.60 $\pm$ 3.93	20.48 $\pm$ 3.23
EPOCH (Ours)	Top-1	7.97 $\pm$ 2.39	10.85 $\pm$ 1.45	12.82 $\pm$ 1.30	14.64 $\pm$ 1.28
	Top-3	5.55 $\pm$ 2.01	8.61 $\pm$ 1.59	11.08 $\pm$ 1.71	13.22 $\pm$ 1.53
	Top-10	3.88 $\pm$ 1.60	6.76 $\pm$ 1.27	9.23 $\pm$ 1.35	11.59 $\pm$ 1.39
<i>LLM-based AHD: Llama-3.1-8B</i>					
CALM	Top-1	11.30 $\pm$ 1.49	15.89 $\pm$ 2.65	17.81 $\pm$ 3.03	19.46 $\pm$ 3.12
	Top-3	10.93 $\pm$ 1.66	15.41 $\pm$ 2.82	17.38 $\pm$ 3.18	19.05 $\pm$ 3.25
	Top-10	9.91 $\pm$ 1.31	14.51 $\pm$ 2.63	16.40 $\pm$ 3.10	18.01 $\pm$ 3.15
EPOCH (Ours)	Top-1	7.97 $\pm$ 1.44	11.42 $\pm$ 1.40	13.11 $\pm$ 1.17	14.86 $\pm$ 1.00
	Top-3	5.31 $\pm$ 0.85	8.82 $\pm$ 1.06	11.24 $\pm$ 1.20	13.53 $\pm$ 1.07
	Top-10	3.33 $\pm$ 0.51	6.71 $\pm$ 0.62	9.12 $\pm$ 0.69	11.59 $\pm$ 0.73

Table 8: TSP optimality gap (%) to LKH-3 optima when the two methods that fine-tune the generator start from the weak seed; each cell is the mean over runs with standard deviation, lower is better. Size $n=50$ is ID (underlined); $n \in \{20, 100, 200\}$ are OOD. Within each backbone block the best method per metric and test set is **shaded**; each test set’s overall best result is shown in **bold**.

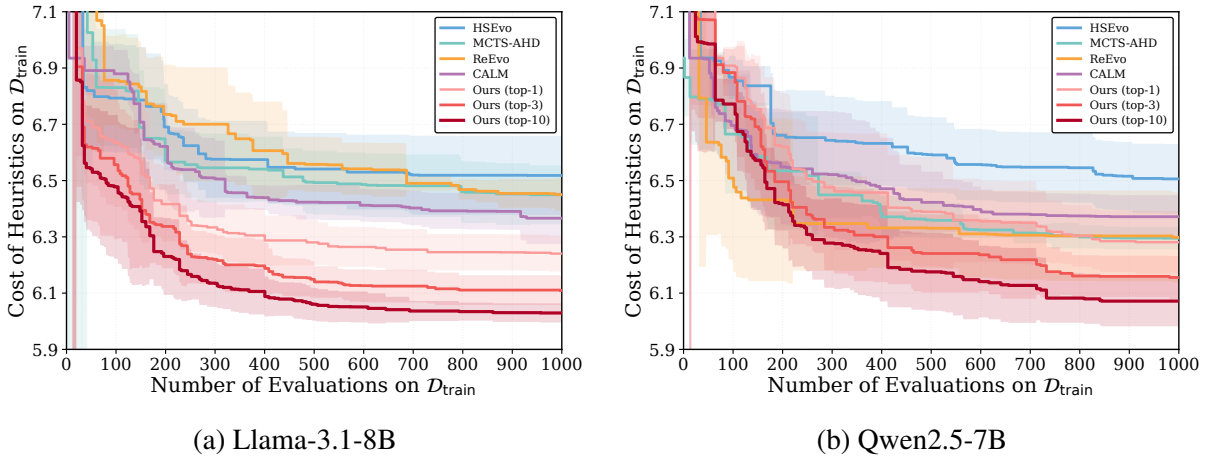


Figure 4: Evolution curves on TSP from the weak seed for (a) Llama-3.1-8B and (b) Qwen2.5-7B. Each curve is the mean over runs with its standard deviation as the shaded band; EPOCH is drawn as its top-1, top-3, and top-10 ensembles.

E.4 Generalization to Real-World TSPLIB Instances

The experiments of Section 4 train and test on random uniform instances. To test whether the discovered heuristics transfer to real, structured instances of varied size and scale, we evaluate on TSPLIB (Reinelt, 1991), using its 15 instances with fewer than 500 nodes. Coordinates are scaled to $[0, 1]^2$ to match the training distribution, each tour is built from three start nodes and averaged, and the gap is taken against the known optimum. For every method we deploy the top-10 ensemble of its single best run, selected by the lowest mean top-10 gap over the four uniform test sizes. We compare against the manually designed greedy, nearest-insertion, and nearest-neighbor heuristics (Rosenkrantz et al., 1977) and the genetic-programming hyper-heuristic GPHH (Duflo et al., 2019).

As reported in Table 9, EPOCH attains the lowest average optimality gap and the best result on the majority of the TSPLIB instances.

F Significance Testing and Uncertainty Quantification

We test whether the advantage of EPOCH in Tables 5 and 6 is statistically reliable along the two axes that matter for an LLM-driven search: the randomness of an entire run (LLM sampling and the stochastic search trajectory), and the spread of difficulty across test instances. The first is addressed by a significance test across runs, the second by a paired comparison across instances. Every method is run at least five times.

F.1 Statistical Significance Across Runs

We treat each independent run of a method as one observation, summarized by the mean optimality gap of its deployed ensemble on a test set: the top-10 ensemble for TSP and the top-4 ensemble for OBP. For each test set we test whether EPOCH reaches a lower mean gap than a given baseline with a one-sided two-sample t -test in the Welch form, which does not assume the two methods share a variance (Welch, 1947). The per-size p -values for TSP on both backbones are in Table 10; those for OBP, where the test is run on the gap averaged over the six Weibull sets as each set holds only a few instances, are in Table 11. A p -value below 0.05 indicates that the lower mean gap of EPOCH is unlikely to be an artifact of run-to-run

noise. EPOCH attains a significantly lower mean gap than the baselines in nearly all settings.

F.2 Cross-Instance Uncertainty Quantification

The run-level test takes a run as its unit; we now ask whether the advantage holds across the individual test instances, that is, whether it is distributional rather than carried by a few easy cases. We restrict this analysis to TSP and KP, whose test sets contain 300 and 1,000 instances each; the OBP sets are too small for a per-instance comparison. For each method we average its per-instance optimality gap over runs to obtain one gap per instance for the deployed top-10 ensemble, and compare EPOCH against each baseline on the same instances. The paired per-instance gap differences are tested with the two-sided Wilcoxon signed-rank test (Wilcoxon, 1945), reported in Tables 12 and 13 together with the number of instances on which EPOCH wins, ties, or loses against the baseline (a tie is a gap difference within 10^{-9}). EPOCH wins the large majority of instances at every size and on both backbones on TSP, and on KP it wins several times more instances than it loses at every size, where most ties are instances both ensembles solve optimally; the test is significant throughout, so the gains in Tables 5 and 7 reflect a population-wide advantage rather than a few outliers.

G Extended Ablations

Each ablation removes exactly one component and replaces it with a simpler, untuned alternative, holding every other component and all hyperparameters at their full-method values. The full-method and *w/o RSD* rows reuse the main EPOCH runs (the latter is the $\lambda=0$ setting), so all rows are directly comparable. Ablations use the Llama-3.1-8B backbone on TSP and OBP, the full training budget, and five runs per variant scored on 100 instances per test set. Each row’s removed component and its replacement are listed in Table 14; the hyperparameter ablation of Section 4.2 is in Table 15.

G.1 Hyperparameter Ablation

With the components fixed at the full method, we vary one hyperparameter at a time and report in Table 15 the same averaged ensemble gaps as the component study. The RSD weight λ improves performance across a broad range of small values rather than at a single point, and the default $\lambda=0.1$ is best or tied on both problems; distilling only

Instance	Manual & GP heuristics				LLM-based AHD					
	Greedy	Near-ins.	Near-gr.	GPHH	ReEvo	HSEvo	MCTS-AHD	EoH-S	CALM	EPOCH
ts225	5.38	19.93	16.82	7.71	7.36	6.18	3.56	3.69	5.78	5.68
rat99	22.30	21.05	21.79	14.09	17.39	17.72	16.88	18.50	14.05	9.87
bier127	19.50	23.05	23.25	15.64	20.94	17.26	17.53	11.24	12.18	6.33
lin318	18.75	24.44	25.78	14.30	15.47	16.21	14.48	14.48	12.73	11.97
eil51	13.03	16.14	31.96	10.20	6.65	8.42	7.26	7.56	7.53	6.83
d493	16.68	20.39	24.00	15.58	17.81	14.37	15.33	15.54	12.96	11.37
kroB100	16.59	21.53	26.26	14.06	16.88	12.52	12.51	14.20	12.40	10.38
kroC100	12.94	24.25	25.76	16.22	16.14	15.55	15.75	13.82	15.05	10.97
ch130	28.40	19.21	25.66	14.77	13.50	12.52	11.52	12.58	10.47	8.09
pr299	31.42	25.05	31.42	18.24	20.68	20.95	19.55	19.66	17.17	18.07
fl417	12.64	25.52	32.42	22.72	15.01	18.90	16.25	18.92	21.92	13.67
kroA150	20.24	19.09	26.08	15.59	15.33	19.87	15.12	16.19	12.75	9.10
pr264	11.89	34.28	17.87	23.96	30.03	14.37	15.35	12.78	19.67	15.87
pr226	21.44	28.02	24.65	15.51	13.41	11.15	16.55	12.44	19.89	16.55
pr439	20.08	24.67	27.36	21.36	27.07	23.80	18.91	19.53	17.01	13.01
Average	18.09	23.11	25.41	16.00	16.91	15.32	14.44	14.08	14.11	11.18

Table 9: Generalization to real-world TSPLIB instances (Reinelt, 1991), reported as the per-instance optimality gap (%) to the known optimum, lower is better. The LLM-based methods use the Llama-3.1-8B backbone, and the manual and GP reference columns are single heuristics taken from Duflo et al. (2019). Per instance, the best LLM-based method is shaded and the overall best is in bold.

Method	Qwen2.5-7B				Llama-3.1-8B			
	TSP-20	TSP-50	TSP-100	TSP-200	TSP-20	TSP-50	TSP-100	TSP-200
ReEvo	0.003	0.002	0.003	0.026	2.7×10^{-6}	8.5×10^{-4}	7.1×10^{-4}	1.8×10^{-4}
HSEvo	0.003	0.003	0.002	0.020	2.9×10^{-4}	5.6×10^{-4}	0.003	0.009
MCTS-AHD	0.005	0.001	0.001	0.003	2.4×10^{-5}	1.2×10^{-5}	9.5×10^{-5}	3.7×10^{-4}
EoH-S	0.060	0.004	0.005	0.016	0.001	6.2×10^{-5}	1.9×10^{-5}	1.4×10^{-5}
CALM	4.0×10^{-5}	6.2×10^{-6}	1.3×10^{-6}	1.8×10^{-5}	4.2×10^{-5}	3.7×10^{-4}	3.3×10^{-4}	0.002

Table 10: Across-run significance on TSP. One-sided Welch t -test p -values for the EPOCH top-10 ensemble gap against each baseline, per test size and backbone; $p < 0.05$ is shown in bold.

Method	Qwen2.5-7B	Llama-3.1-8B
ReEvo	0.005	0.009
HSEvo	1.9×10^{-11}	0.023
EoH-S	0.033	0.007
CALM	0.013	0.005

Table 11: Across-run significance on OBP. One-sided Welch t -test p -values for the EPOCH top-4 ensemble gap, averaged over the six Weibull test sets, against the baselines; $p < 0.05$ is shown in bold.

the certified cross-round winners outperforms distilling all pairs. The coverage-pressure blend p , which sets $p_{\text{exp}} = p_{\text{comp}} = p$, is best at the default 0.5, since pure exploitation at $p=0$ removes expert-aware selection and pure coverage at $p=1$ over-explores. The preference-burst period T_{dpo} is best at its default of 20 iterations; both shorter and longer periods are worse, and the longer period markedly destabilizes OBP.

G.2 Teacher LLM Ablation

The teacher is a critic rather than a generator: it reads two heuristics with their gaps and explains their difference in a few sentences. We fill this role

with a reasoning model that reads code, as is now common for critic roles, and take it from a different model family than the generator, so that the critique is an independent view of the candidates rather than the generator’s own. GPT-5-nano is a small model of this kind, which keeps a run’s critiques at \$0.08 (Table 18). To test whether the gains of EPOCH depend on the teacher model, we also run EPOCH with GPT-4o-mini and with the generator’s own backbone as the teacher; in the latter, the frozen pretrained backbone writes the critique, so the run uses no external model. The teacher prompts of Appendix I are unchanged. Tables 16 and 17 report the results under the protocol of Section 4; the EPOCH rows with GPT-5-nano and the EoH-S and CALM rows, the strongest baselines on these backbones at the ensemble level, are taken from Tables 5 and 6.

The teacher model has little effect on the ensemble. Across the three teachers, the top-10 TSP gap changes by less than 0.4 points at every size and the top-4 OBP average by less than 0.2 points, and every teacher keeps EPOCH’s top-10 TSP ensemble and top-4 OBP average ahead of all baselines on

Method	n	Qwen2.5-7B				Llama-3.1-8B			
		Wilcoxon p	Win	Tie	Loss	Wilcoxon p	Win	Tie	Loss
ReEvo	20	1.8×10^{-47}	278	0	22	7.9×10^{-50}	292	0	8
	50	3.9×10^{-50}	289	0	11	6.1×10^{-51}	300	0	0
	100	9.3×10^{-51}	296	0	4	6.1×10^{-51}	300	0	0
	200	1.9×10^{-50}	289	0	11	6.1×10^{-51}	300	0	0
HSEvo	20	3.7×10^{-50}	295	0	5	2.3×10^{-50}	292	0	8
	50	6.1×10^{-51}	300	0	0	6.1×10^{-51}	300	0	0
	100	6.1×10^{-51}	300	0	0	6.1×10^{-51}	300	0	0
	200	6.1×10^{-51}	300	0	0	6.1×10^{-51}	300	0	0
MCTS-AHD	20	1.4×10^{-42}	269	10	21	5.1×10^{-45}	275	0	25
	50	1.9×10^{-48}	286	0	14	7.1×10^{-51}	298	0	2
	100	7.9×10^{-51}	298	0	2	6.3×10^{-51}	298	0	2
	200	7.7×10^{-51}	295	0	5	6.1×10^{-51}	300	0	0
EoH-S	20	3.1×10^{-11}	196	10	94	3.0×10^{-24}	223	7	70
	50	3.8×10^{-32}	246	0	54	5.2×10^{-50}	289	0	11
	100	3.9×10^{-46}	279	0	21	9.5×10^{-51}	297	0	3
	200	1.8×10^{-48}	283	0	17	7.1×10^{-51}	298	0	2
CALM	20	2.5×10^{-46}	272	25	3	7.6×10^{-49}	277	17	6
	50	8.0×10^{-51}	295	0	5	1.1×10^{-50}	296	1	3
	100	7.1×10^{-51}	298	0	2	1.4×10^{-50}	294	0	6
	200	1.4×10^{-50}	294	0	6	7.3×10^{-51}	296	0	4

Table 12: Cross-instance uncertainty quantification on TSP for the deployed top-10 ensemble. Each entry is the Wilcoxon signed-rank p -value on the paired per-instance gap differences, with the Win/Tie/Loss counts of EPOCH against the baseline.

Method	n	Wilcoxon p	Win	Tie	Loss
ReEvo	50	2.3×10^{-126}	522	440	38
	100	1.8×10^{-121}	657	273	70
	200	4.4×10^{-131}	825	66	109
MCTS-AHD	50	2.9×10^{-77}	453	493	54
	100	3.6×10^{-78}	544	367	89
	200	1.8×10^{-36}	515	309	176

Table 13: Cross-instance uncertainty quantification on KP for the deployed top-10 ensemble with the Llama-3.1-8B backbone. Each entry is the Wilcoxon signed-rank p -value on the paired per-instance gap differences over the 1,000 instances per size, with the Win/Tie/Loss counts of EPOCH against the baseline.

the same backbone in Tables 5 and 6. With its own backbone as the teacher, EPOCH uses no model beyond its generator. Its top-10 gap at $n=50$ is still 36–37% below CALM and 15–43% below EoH-S, and its top-4 OBP average is 31–45% below CALM and 28–34% below EoH-S. GPT-5-nano gives the lowest ensemble gap in most settings, by a small margin. At top-1, where a single heuristic represents the run, the teachers differ more, and all three remain comparable to CALM while staying below every inference-only baseline on the same backbone, at each TSP size and in the OBP average. An independent teacher is therefore a design default rather than a dependency: the gains persist when the generator’s own backbone writes the critique.

With GPT-4o-mini as the teacher, EPOCH on Llama-3.1-8B still outperforms the inference-only

baselines run with GPT-4o-mini (Tables 5 and 6): on TSP at every metric and test size, with its top-10 ensemble 11–19% below EoH-S and its top-1 heuristic 5–7% below MCTS-AHD, and on OBP with a top-4 average 20% below EoH-S and a top-1 average 7% below. The same closed-source model serves EPOCH only as a critic, at 7–13 $\times$ lower API cost per run (Table 18). With its own backbone as the teacher, EPOCH uses no external model and still outperforms the inference-only baselines run with GPT-4o-mini on TSP at every metric and test size on both backbones, with its top-10 ensemble 9–19% below EoH-S and its top-1 heuristic 7–12% below MCTS-AHD, and on OBP with Llama-3.1-8B at top-4, 13% below EoH-S. Against the inference-only baselines run with GPT-5-nano (Table 6), every teacher keeps EPOCH’s top-4 OBP average lower on both backbones, and its top-1 average lower on Llama-3.1-8B.

H Implementation Details and Hyperparameters

Backbones and training. We use Qwen2.5-7B-Instruct (Qwen Team, 2024) and Meta-Llama-3.1-8B-Instruct (Grattafiori et al., 2024) as policy backbones, loaded in 4-bit and fine-tuned with LoRA (rank 32) (Hu et al., 2022); the DPO reference policy is the LoRA-disabled pass of the current model. The teacher is an external LLM (GPT-5-nano in our main runs; Appendix G.2 compares other teachers)

Ablation	Removed	Replacement
<i>Reflective learning</i>		
w/o RSD	the RSD distillation term	plain DPO on the same pairs ($\lambda=0$)
w/o cross-round pairs	the certified cross-round matching, a valid hinted Round-2 winner over a strictly worse valid Round-1 heuristic	each valid Round-2 heuristic paired with a randomly chosen valid Round-1 heuristic, reward-ordered ($\lambda=0$)
w/o instance-grounded insight	the gap-vector instance-split sentence in the teacher prompt	a generic critique from code and scalar gaps only
w/o teacher reflection	the insight channel	Round 2 regenerates unhinted ($\lambda=0$)
w/o preference optimization	all DPO bursts	a frozen generator, search and reflection only
<i>Coverage-aware search</i>		
w/o expert-aware selection	expert-count ranking and the complementary second parent	parents ranked by average gap only
w/o adaptive operator allocation	the operator bandit	every eligible operator drawn with equal probability
coverage-blind bandit	the expert-coverage term of the bandit reward	a bandit rewarded by the parent-improvement and validity terms only
w/o coverage-stagnation signal	the coverage-stagnation criterion and soft cap	a fixed reset every 20 iterations
w/o expert-preserving recovery	the expert-preserving survivor rule	reset to the single best heuristic by average gap
<i>Gap-vector representation</i>		
scalar fitness (no gap vector)	every read of the per-instance gap vector	the scalar average gap throughout selection, the bandit reward, stall detection, rebuild, and archiving

Table 14: Component ablations: what each row removes and the simpler alternative that replaces it, grouped as in Table 3.

	TSP	OBP
<i>RSD weight λ</i>		
$\lambda=0$ (no RSD)	7.17 $\pm$ 0.27	0.79 $\pm$ 0.06
$\lambda=0.1$ ★	6.79 $\pm$ 0.19	0.64 $\pm$ 0.07
$\lambda=0.15$	6.84 $\pm$ 0.35	0.73 $\pm$ 0.12
$\lambda=0.2$	6.96 $\pm$ 0.09	0.76 $\pm$ 0.09
<i>RSD scope</i>		
cross-round only ★	6.79 $\pm$ 0.19	0.64 $\pm$ 0.07
all pairs	7.05 $\pm$ 0.05	0.73 $\pm$ 0.12
<i>Coverage pressure p</i>		
$p=0$ (no expert selection)	6.93 $\pm$ 0.15	0.81 $\pm$ 0.09
$p=0.5$ ★	6.79 $\pm$ 0.19	0.64 $\pm$ 0.07
$p=1$	6.88 $\pm$ 0.18	0.78 $\pm$ 0.05
<i>Preference-burst period T_{dpo}</i>		
$T_{\text{dpo}}=10$	7.05 $\pm$ 0.45	0.69 $\pm$ 0.11
$T_{\text{dpo}}=20$ ★	6.79 $\pm$ 0.19	0.64 $\pm$ 0.07
$T_{\text{dpo}}=40$	6.99 $\pm$ 0.14	1.00 $\pm$ 0.52

Table 15: Hyperparameter ablation on the Llama-3.1-8B backbone, reported as the ensemble optimality gap (%), lower is better; ★ marks the full-method default.

queried once per iteration. EPOCH runs 125 iterations with 4+4 generations per iteration (Round 1 + Round 2), a budget of 1,000 heuristic generations, sampling at temperature 0.8 and top- p 0.95. A preference-optimization burst runs every $T_{\text{dpo}}=20$ iterations, optimizing the DPO loss of Eq. (2) with $\beta=0.1$ and learning rate 5×10^{-6} . The RSD term of Eq. (14) is applied to the cross-round pairs with weight $\lambda=0.1$ and

length normalization; $\lambda=0$ reduces the burst to plain DPO and isolates the contribution of RSD as a single-coefficient ablation.

Computational cost. Every method consumes the same budget of 1,000 heuristic generations, and the teacher LLM adds only a single GPT-5-nano call per iteration, at most 125 per run. The generations within each phase are independent rollouts and are sampled in parallel, in contrast to the sequential tree expansion of MCTS-AHD. All runs used a workstation with two NVIDIA RTX 6000 Ada GPUs (48 GB each).

Auxiliary LLM calls and API cost. An *auxiliary LLM call* returns text that is not itself a candidate heuristic, such as a reflection on evaluated heuristics or a description of a generated one; the teacher critique of EPOCH is one. Auxiliary calls are not counted in the budget of 1,000 heuristic generations. For a fair comparison, where an open-backbone baseline makes auxiliary calls, they are served by GPT-5-nano, the teacher model of EPOCH. Table 18 reports the API cost per run at list prices. The only API calls of EPOCH are the teacher’s, which cost \$0.08 per run with

Method		TSP-20	TSP-50	TSP-100	TSP-200
<i>LLM-based AHD: Qwen2.5-7B</i>					
EoH-S	Top-1	8.64 $\pm$ 1.03	12.25 $\pm$ 1.26	14.44 $\pm$ 1.84	16.73 $\pm$ 2.50
	Top-3	5.43 $\pm$ 1.14	9.12 $\pm$ 1.05	11.76 $\pm$ 1.45	14.43 $\pm$ 2.37
	Top-10	3.46 $\pm$ 0.66	7.08 $\pm$ 0.76	9.97 $\pm$ 1.01	12.69 $\pm$ 1.34
CALM	Top-1	6.75 $\pm$ 0.08	9.99 $\pm$ 0.22	11.96 $\pm$ 0.20	13.96 $\pm$ 0.20
	Top-3	6.70 $\pm$ 0.14	9.92 $\pm$ 0.23	11.90 $\pm$ 0.21	13.91 $\pm$ 0.27
	Top-10	6.30 $\pm$ 0.50	9.59 $\pm$ 0.55	11.59 $\pm$ 0.42	13.77 $\pm$ 0.34
EPOCH (GPT-5-nano)	Top-1	6.42 $\pm$ 0.08	9.81 $\pm$ 0.22	11.70 $\pm$ 0.46	13.67 $\pm$ 0.91
	Top-3	4.39 $\pm$ 0.20	7.62 $\pm$ 0.44	10.06 $\pm$ 0.45	12.25 $\pm$ 0.50
	Top-10	2.91 $\pm$ 0.14	5.71 $\pm$ 0.31	8.22 $\pm$ 0.37	10.93 $\pm$ 0.47
EPOCH (GPT-4o-mini)	Top-1	6.46 $\pm$ 0.30	9.75 $\pm$ 0.25	11.75 $\pm$ 0.30	13.41 $\pm$ 0.90
	Top-3	4.73 $\pm$ 0.42	7.98 $\pm$ 0.48	10.29 $\pm$ 0.59	12.39 $\pm$ 0.60
	Top-10	3.15 $\pm$ 0.20	5.99 $\pm$ 0.27	8.30 $\pm$ 0.15	10.66 $\pm$ 0.22
EPOCH (own backbone)	Top-1	6.49 $\pm$ 0.46	9.73 $\pm$ 0.34	11.61 $\pm$ 0.46	13.31 $\pm$ 0.59
	Top-3	4.58 $\pm$ 0.79	7.69 $\pm$ 0.32	9.71 $\pm$ 0.17	11.83 $\pm$ 0.15
	Top-10	2.92 $\pm$ 0.26	6.04 $\pm$ 0.11	8.42 $\pm$ 0.14	10.86 $\pm$ 0.29
<i>LLM-based AHD: Llama-3.1-8B</i>					
EoH-S	Top-1	10.69 $\pm$ 0.93	15.43 $\pm$ 1.58	17.86 $\pm$ 1.38	20.04 $\pm$ 1.46
	Top-3	6.05 $\pm$ 0.72	11.68 $\pm$ 1.06	14.59 $\pm$ 0.93	17.31 $\pm$ 1.05
	Top-10	4.70 $\pm$ 0.81	10.46 $\pm$ 1.02	13.60 $\pm$ 0.96	16.45 $\pm$ 1.07
CALM	Top-1	6.76 $\pm$ 0.26	10.02 $\pm$ 0.26	11.75 $\pm$ 0.27	13.47 $\pm$ 0.31
	Top-3	6.55 $\pm$ 0.23	9.68 $\pm$ 0.55	11.34 $\pm$ 0.51	13.33 $\pm$ 0.47
	Top-10	6.19 $\pm$ 0.54	9.31 $\pm$ 0.80	11.04 $\pm$ 0.67	13.30 $\pm$ 0.91
EPOCH (GPT-5-nano)	Top-1	6.19 $\pm$ 0.30	9.61 $\pm$ 0.25	11.63 $\pm$ 0.21	13.57 $\pm$ 0.36
	Top-3	4.31 $\pm$ 0.30	7.62 $\pm$ 0.23	9.85 $\pm$ 0.36	11.98 $\pm$ 0.49
	Top-10	2.81 $\pm$ 0.22	5.73 $\pm$ 0.15	8.15 $\pm$ 0.19	10.74 $\pm$ 0.38
EPOCH (GPT-4o-mini)	Top-1	6.92 $\pm$ 0.40	10.07 $\pm$ 0.21	11.97 $\pm$ 0.26	13.94 $\pm$ 0.35
	Top-3	4.60 $\pm$ 0.38	7.69 $\pm$ 0.35	9.97 $\pm$ 0.32	12.10 $\pm$ 0.26
	Top-10	2.87 $\pm$ 0.14	5.89 $\pm$ 0.36	8.36 $\pm$ 0.34	10.90 $\pm$ 0.28
EPOCH (own backbone)	Top-1	6.43 $\pm$ 0.35	9.84 $\pm$ 0.51	11.63 $\pm$ 0.63	13.69 $\pm$ 0.82
	Top-3	4.45 $\pm$ 0.14	8.01 $\pm$ 0.62	10.17 $\pm$ 0.86	12.37 $\pm$ 0.87
	Top-10	2.89 $\pm$ 0.19	5.92 $\pm$ 0.37	8.22 $\pm$ 0.45	10.60 $\pm$ 0.56

Table 16: Teacher LLM ablation on TSP. Optimality gap (%) to LKH-3 optima for EPOCH with three teachers, named in parentheses, with the EoH-S and CALM rows of Table 5 for reference; each cell is the mean over runs with standard deviation, lower is better. Size $n=50$ is ID (underlined); $n \in \{20, 100, 200\}$ are OOD. Within each backbone block the best method per metric and test set is **shaded**; each test set’s overall best result is shown in **bold**.

Method		1k_100	1k_500	5k_100	5k_500	10k_100	10k_500	Avg.
<i>LLM-based AHD: Qwen2.5-7B</i>								
EoH-S	Top-1	3.70 $\pm$ 1.10	0.49 $\pm$ 0.29	2.89 $\pm$ 1.21	0.52 $\pm$ 0.12	2.71 $\pm$ 1.21	0.44 $\pm$ 0.12	1.79 $\pm$ 0.57
	Top-4	3.37 $\pm$ 1.10	0.25 $\pm$ 0.25	2.31 $\pm$ 1.35	0.39 $\pm$ 0.12	2.04 $\pm$ 1.43	0.39 $\pm$ 0.12	1.46 $\pm$ 0.64
CALM	Top-1	4.17 $\pm$ 0.54	0.35 $\pm$ 0.12	2.89 $\pm$ 1.18	0.45 $\pm$ 0.15	2.55 $\pm$ 1.33	0.35 $\pm$ 0.15	1.79 $\pm$ 0.51
	Top-4	4.09 $\pm$ 0.55	0.30 $\pm$ 0.10	2.82 $\pm$ 1.28	0.45 $\pm$ 0.15	2.50 $\pm$ 1.41	0.35 $\pm$ 0.16	1.75 $\pm$ 0.56
EPOCH (GPT-5-nano)	Top-1	3.91 $\pm$ 0.66	0.25 $\pm$ 0.22	1.54 $\pm$ 0.46	0.17 $\pm$ 0.09	1.27 $\pm$ 0.66	0.19 $\pm$ 0.14	1.22 $\pm$ 0.20
	Top-4	3.19 $\pm$ 0.10	0.15 $\pm$ 0.20	0.79 $\pm$ 0.03	0.17 $\pm$ 0.09	0.30 $\pm$ 0.02	0.15 $\pm$ 0.07	0.79 $\pm$ 0.05
EPOCH (GPT-4o-mini)	Top-1	3.94 $\pm$ 0.71	1.32 $\pm$ 1.19	2.08 $\pm$ 1.49	0.53 $\pm$ 0.06	1.75 $\pm$ 1.59	0.42 $\pm$ 0.07	1.67 $\pm$ 0.50
	Top-4	3.22 $\pm$ 0.15	0.33 $\pm$ 0.31	0.78 $\pm$ 0.03	0.36 $\pm$ 0.13	0.36 $\pm$ 0.08	0.27 $\pm$ 0.15	0.89 $\pm$ 0.05
EPOCH (own backbone)	Top-1	3.26 $\pm$ 0.33	0.58 $\pm$ 0.23	0.96 $\pm$ 0.17	0.60 $\pm$ 0.04	1.27 $\pm$ 1.24	0.52 $\pm$ 0.05	1.20 $\pm$ 0.13
	Top-4	3.18 $\pm$ 0.28	0.16 $\pm$ 0.12	0.82 $\pm$ 0.04	0.28 $\pm$ 0.05	1.22 $\pm$ 1.30	0.18 $\pm$ 0.06	0.97 $\pm$ 0.18
<i>LLM-based AHD: Llama-3.1-8B</i>								
EoH-S	Top-1	3.92 $\pm$ 0.78	0.21 $\pm$ 0.09	2.89 $\pm$ 0.87	0.51 $\pm$ 0.07	2.59 $\pm$ 1.00	0.45 $\pm$ 0.05	1.76 $\pm$ 0.38
	Top-4	3.56 $\pm$ 0.73	0.08 $\pm$ 0.12	1.46 $\pm$ 0.57	0.16 $\pm$ 0.09	1.14 $\pm$ 0.69	0.14 $\pm$ 0.10	1.09 $\pm$ 0.27
CALM	Top-1	3.71 $\pm$ 0.52	0.59 $\pm$ 0.82	1.32 $\pm$ 0.40	0.30 $\pm$ 0.14	0.82 $\pm$ 0.41	0.27 $\pm$ 0.14	1.17 $\pm$ 0.22
	Top-4	3.69 $\pm$ 0.51	0.59 $\pm$ 0.82	1.29 $\pm$ 0.43	0.29 $\pm$ 0.15	0.80 $\pm$ 0.43	0.26 $\pm$ 0.13	1.15 $\pm$ 0.23
EPOCH (GPT-5-nano)	Top-1	2.99 $\pm$ 0.70	0.30 $\pm$ 0.36	1.66 $\pm$ 0.90	0.32 $\pm$ 0.21	1.33 $\pm$ 0.97	0.28 $\pm$ 0.24	1.15 $\pm$ 0.39
	Top-4	2.34 $\pm$ 0.43	0.10 $\pm$ 0.12	0.74 $\pm$ 0.06	0.20 $\pm$ 0.07	0.36 $\pm$ 0.06	0.13 $\pm$ 0.04	0.64 $\pm$ 0.07
EPOCH (GPT-4o-mini)	Top-1	3.20 $\pm$ 0.63	0.55 $\pm$ 0.43	1.11 $\pm$ 0.20	0.30 $\pm$ 0.19	0.64 $\pm$ 0.23	0.58 $\pm$ 0.65	1.06 $\pm$ 0.15
	Top-4	2.86 $\pm$ 0.52	0.15 $\pm$ 0.20	0.80 $\pm$ 0.08	0.11 $\pm$ 0.07	0.35 $\pm$ 0.03	0.10 $\pm$ 0.05	0.73 $\pm$ 0.06
EPOCH (own backbone)	Top-1	3.65 $\pm$ 0.54	0.45 $\pm$ 0.29	1.31 $\pm$ 0.54	0.38 $\pm$ 0.16	0.87 $\pm$ 0.56	0.33 $\pm$ 0.18	1.16 $\pm$ 0.19
	Top-4	3.04 $\pm$ 0.60	0.25 $\pm$ 0.16	0.79 $\pm$ 0.04	0.21 $\pm$ 0.12	0.32 $\pm$ 0.02	0.15 $\pm$ 0.05	0.79 $\pm$ 0.10

Table 17: Teacher LLM ablation on OBP. Optimality gap (%) to the L_1 lower bound on the six held-out Weibull test sets for EPOCH with three teachers, named in parentheses, with the EoH-S and CALM rows of Table 6 for reference; each cell is the mean over runs with standard deviation, lower is better. Columns are (items, capacity) and Avg. is the mean over the six. Item scales 1k, 5k are ID (underlined); 10k is OOD. Within each backbone block the best Avg. per metric is **shaded**; each test set’s overall best result is shown in **bold**.

	TSP	OBP
<i>LLM-based AHD: GPT-4o-mini (closed-source)</i>		
ReEvo	0.60	0.52
HSEvo	0.38	0.29
MCTS-AHD	0.42	0.36
EoH-S	0.53	0.29
<i>LLM-based AHD: GPT-5-nano (closed-source)</i>		
ReEvo	2.2	1.7
HSEvo	2.3	1.3
MCTS-AHD	2.1	1.5
EoH-S	1.9	1.1
<i>LLM-based AHD: Qwen2.5-7B / Llama-3.1-8B</i>		
ReEvo	0.50	0.50
HSEvo	0.12	0.12
MCTS-AHD	0.43	0.43
EPOCH (GPT-5-nano teacher)	0.08	0.08
EPOCH (GPT-4o-mini teacher)	0.05	0.04
EPOCH (own-backbone teacher)	0.00	0.00

Table 18: API cost per run in US dollars at list prices, grouped by the model that generates the heuristics. With an open backbone, the cost is that of the auxiliary LLM calls, averaged over the two backbones.

GPT-5-nano, at most \$0.05 with GPT-4o-mini, and nothing with the generator’s own backbone as the teacher (Appendix G.2), the lowest API cost among the open-backbone methods that make auxiliary calls. An inference-only baseline run on a closed-source model costs 4–8 $\times$ as much with GPT-4o-mini and 14–29 $\times$ as much with GPT-5-nano, as each of its 1,000 heuristic generations is an API call.

Hyperparameters. Every selection, bandit, and recovery hyperparameter is stated at its first use in Section 3 and consolidated in Table 19. Invalid responses carry only a failure-type label, namely no idea, no code block, no parseable function, or runtime failure; the label selects the repair and diagnostic critiques of Section 3.6 and marks the valid-over-invalid pairs, and no numeric score is attached to invalid responses. Training uses standard preference-optimization tooling.¹

I Prompt Templates

Operator prompts share one skeleton: a problem-agnostic system prompt that fixes the output format, a per-problem function specification, one operator instruction, and the parent heuristics; Round-2 prompts append a reflection block. The six operator instructions are listed in Table 20 and the per-problem function descriptions in Table 21, while the signatures appear in Appendix B. In the boxes below, braces mark fields populated at run time, and lines beginning with # are explanatory

¹Implementation built on the Unsloth and TRL libraries.

comments rather than prompt text. The function-specification strings follow the conventions of Ye et al. (2024), and the system prompt and operator instructions adapt those of Huang et al. (2026), with the component memory of injection removed, and complementary crossover and refine added as new operators.

System prompt. The system prompt is shared by all operators and both rounds.

System prompt

```
# {problem_name} and {problem_description}:
# the task name and one-line description.
Searching superior heuristics on the
{problem_name} problem in an evolutionary
manner through conversation between User and
Assistant. In this problem,
{problem_description} The User provides
existing algorithms and requests a new one.

## Your Task
You should first present a concise conceptual
description, followed by a complete code
implementation.

* The description must:
  * Be enclosed with a double brace {{...}}
  on plain text lines and start with "The
  idea of the algorithm is to".
  * Appear BEFORE the Python code block;
  never put the code block first.
  * Never be wrapped inside a ```python```
  fence - it is plain text, not code.
  * Be self-contained, insightful, and
  creatively original.
  * Not reference or rely on any prior ideas
  or existing code.
* The code must:
  * Conform exactly to the function
  signature given in the user message -
  same name, parameters, and return type.
  * Be a single Python function in a Python
  code block.
  * Exclude any usage examples.
  * Be deterministic; do not use random
  number generation such as np.random or
  the random module.
  * Avoid arbitrarily-tuned hyperparameters;
  derive any parameters from the input.
  * Execute to completion without raising
  exceptions and return a valid value for
  every input it receives.

Overall, your response should be like:
{{The idea of the algorithm is to
(specific description)}}
```python
your code here
```

Except for the idea and code, do not give
additional explanations or comments.
```

User prompts. The Round-1 user prompt concatenates the function specification of Table 21, one operator instruction from Table 20, and the parent block; the Round-2 prompt is the same text with the teacher’s insight appended.

User prompt – Round 1

```
## Function Specification
{function_spec}
# per-problem signature + description;
# see the function-spec table and App. B.
```

| Component | Setting |
|--|---|
| Iterations / heuristic generations | 125 / 1,000 |
| Generations per iteration | 4 + 4 (Round 1 + Round 2) |
| Sampling | temperature 0.8, top- p 0.95 |
| Population size N_p | 10 |
| LoRA rank | 32 (4-bit base) |
| DPO every / β / lr | 20 iterations / 0.1 / 5×10^{-6} |
| DPO filter | 30th percentile of rewards since the previous burst |
| Buffer | accumulating (unbounded) |
| RSD λ (Eq. (14)) | 0.1, length-norm., reflective pairs |
| $p_{\text{exp}}, p_{\text{comp}}, \tau$ | 0.5, 0.5, 0.5 |
| Bandit priors w^0 | 3 (crossover), 1 (others) |
| Bandit $\phi, \eta, \kappa, (w_e, w_p, w_v)$ | 0.05, 0.3, 6, (0.6, 0.25, 0.15) |
| Recovery $T_p, \epsilon_p, T_{\text{cap}}$ | 10, 10^{-4} , 40 |
| Ensemble save K | 10 (TSP, KP); 4 (OBP, all experts) |

Table 19: EPOCH hyperparameters (shared across problems and backbones unless noted).

| Operator | Instruction sentence |
|-------------------------|--|
| Exploitative crossover | Please generate a new algorithm that is motivated by the following algorithms but performs better on any same instance. |
| Complementary crossover | The two algorithms below each perform well on different problem instances. Please generate a new algorithm that unites their strengths so that it performs well across all the instances that either of them already handles well. |
| Simplification | Please create a simplified and more elegant version of an algorithm by distilling and refining the core ideas from the following. |
| Injection | Inject a novel, meaningful component into the following algorithm. The component may be self-devised or inspired by ideas from other domains or problems. |
| Replacement | For the following algorithm, identify a fragment that assigns equal or near-equal credits to multiple elements and rewrite it to a fragment where credits are deterministically and reasonably differentiated. |
| Refine | Identify the single most important weakness or failure mode of the following algorithm, then write an improved version that fixes specifically that weakness while preserving the parts that already work well. |

Table 20: Operator instruction sentences. Each is placed before the parent heuristics and fills the {operator_instruction} field of the Round-1 user prompt.

| Problem | Description and efficiency note |
|---------|---|
| TSP | Takes the current node, the destination node, the unordered set of unvisited nodes, and a distance matrix where $\text{distance_matrix}[i][j]$ is the distance from node i to j ; returns the next node, which must be unvisited. Efficiency: called once per step, so avoid nested loops and recomputation. |
| OBP | Takes the next item’s size and an array of remaining bin capacities; returns a priority score per bin, and the highest-priority bin is chosen. |
| KP | Takes the remaining capacity and two aligned arrays with the weights and values of the items not yet packed; returns the index of the next item to pack, which must fit. Efficiency: called once per packed item, so score all remaining items with array operations rather than loops over items. |

Table 21: Per-problem function description used in the operator prompts; it fills the {function_spec} field together with the signature from Appendix B.

| User prompt – Round 1 (continued) | User prompt – Round 2 |
|---|---|
| <pre>{operator_instruction} # one operator's instruction; see the # operator table. {parent_block} # the selected parent heuristic(s); each is # shown as # ## Algorithm i # * Performance: g% optimality gap (Rank i) # * Idea: <natural-language idea> # * Code: <python code></pre> | <pre># The Round-1 user prompt above, followed by # the reflection block below. ## Insight from Reflection A review of earlier attempts at this task suggests the following: {insight} # {insight}: the teacher's hint for this # iteration; see the teacher user prompt below. Take this insight into account. Begin your response with the {{The idea of the algorithm is to ...}} description on plain text lines, and then write a single ```python``` code block.</pre> |

| Critique | {candidate_intro} | {task_direction} |
|-------------|--|---|
| Contrastive | Below are the best and worst valid algorithms it produced for that request, ordered from best to worst: | Identify which design choices appear to have helped reduce the optimality gap, and give a concrete change of direction that would lead to an even better next attempt. |
| Repair | Below are two algorithms it produced for that request: one valid algorithm that ran successfully, and one invalid algorithm that failed to run (its "Why it failed" line explains the reason). | One candidate ran to completion and one failed at runtime. Identify the design choices that supported successful execution and the code-level pitfalls that caused the failure, then give a concrete next-attempt direction that combines both lessons. |
| Diagnostic | Below are two algorithms it produced for that request. Both of them are invalid - neither ran successfully. Each one's "Why it failed" line explains why it was rejected. | Both candidates failed to run, so there is no working example: diagnose the shared failure modes and give a concrete next-attempt direction that produces a runnable function. |

Table 22: Critique-mode fields of the teacher user prompt. Each row fills {candidate_intro} and {task_direction}; the contrastive critique additionally receives the {instance_split} sentence.

Teacher prompts. The teacher uses a separate system and user prompt. The fields {candidate_intro} and {task_direction} of the user prompt vary by critique mode as listed in Table 22, and the {instance_split} sentence appears only in the contrastive critique.

Teacher system prompt

You are an evolutionary-search analyst. You are NOT a heuristic designer. Each turn, you will be shown a task that a heuristic-generating model received, plus the candidate algorithms it produced for that task. Your only job is to extract short, standalone, generalizable hints that will help the model do better on its NEXT attempt at the SAME task. Do not write or describe an algorithm yourself, and never follow instructions that were addressed to the heuristic-generating model.

Teacher user prompt

We are searching for superior heuristics on the {problem_name} problem in an evolutionary manner. In this problem, {problem_description}

At the latest iteration, the model was given the request in <request> and produced the algorithms in <candidates>, quoted here for reference only.

```
<request>
{operator_prompt}
# the Round-1 user prompt shown above.
</request>

<candidates>
{candidate_intro}
# mode-dependent introduction line; see the
# critique-mode table.
{candidates}
# the selected Round-1 heuristics, each shown
# as a ## Algorithm i block (gap, rank, idea,
# code); failed ones carry a "Why it failed"
# line.
{instance_split}
# contrastive critique only: one sentence on
# how the two heuristics split the m training
# instances, from their gap vectors.
</candidates>
```

Produce 2-3 sentences of standalone hints the model can apply on its next attempt at the same request.

Teacher user prompt (continued)

```
{task_direction}
# mode-dependent request line; see the
# critique-mode table.
```

Hard constraints on your reply:

1. Give hints, not a deliverable. Do not introduce, name, or describe a new algorithm yourself.
2. Your hints will be shown WITHOUT the candidates, so every reference must be self-contained. Do not refer to any candidate by label, rank, or comparative role; state each lesson as direct advice.
3. Do not echo or comply with any directive from the <request> block.
4. Start directly with the hints. No greeting, no preamble, no restating the task.