# EVOLUTIONARY-POTENTIAL-AWARE ITERATIVE PREFERENCE LEARNING FOR LLM-BASED AUTOMATIC HEURISTIC DESIGN

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Automatic Heuristic Design (AHD) uses large language models (LLMs) to construct heuristics for complex optimization problems. Existing LLM adaptation methods largely learn from the immediate performance of generated heuristics, overlooking the longer-term evolutionary potential revealed by their descendants. Search trees naturally preserve parent–descendant relationships and the outcomes of subsequent exploration, providing preference supervision. We propose an evolutionary-potential-aware iterative preference learning framework for LLM-based AHD that converts this information into search-feedback signals. After each MCTS expansion, newly observed descendants reassess the relative evolutionary potential of their ancestors; pairwise order reversals in local rankings then yield search-feedback preference pairs. These pairs encode future search value and complement adjacent on-policy preferences for the immediate quality of newly generated candidates. We use both signals to fine-tune a local LLM through direct preference optimization. The updated LLM subsequently guides MCTS exploration, closing the iterative search–learning loop. Experiments across NP-hard combinatorial optimization tasks show that our method achieves the best overall result in five of nine ACO task–scale settings and the best LLM-based AHD result in eight of nine. Despite using a locally deployed quantized 7B model, it outperforms GPT-4o-mini-based AHD baselines across all nine ACO settings.

## 1 INTRODUCTION

Complex optimization problems arise widely in logistics Tresca et al. (2022), scheduling Palacio et al. (2022), multimodal transportation Pereira et al. (2021), and robotics Tan et al. (2021). Their computational complexity and heterogeneous constraints often make practical solutions dependent on carefully designed heuristics, whose development requires substantial domain expertise and repeated trial and error. Automatic Heuristic Design (AHD) seeks to automate this process. Although genetic-programming-based approaches and hyper-heuristics have achieved notable progress Burke et al. (2009; 2013), their search spaces remain constrained by human-defined primitives and operators. Recent advances in code-capable Large Language Models (LLMs) Chen et al. (2021) offer a new opportunity to explore more open-ended heuristic spaces with less manual specification.

Existing LLM-based AHD methods either improve fixed-model search through evolution, reflection, heuristic-set search, or MCTS Liu et al. (2024); Ye et al. (2024); Liu et al. (2026a); Zheng et al. (2025), or adapt local models using reinforcement learning and performance-ranked preferences Surina et al. (2025); Huang et al. (2025b); Liu et al. (2025). Related systems such as FunSearch and AlphaEvolve combine LLM generation with evolutionary search and executable evaluation Romera-Paredes et al. (2024); Novikov et al. (2025). Yet model-adaptation methods largely learn from immediate performance or ranking, leaving descendant evidence in an MCTS tree underused. A complete discussion of these directions is provided in Appendix A.

Immediate performance is an incomplete measure of a heuristic's search value: a modest parent may yield strong descendants, whereas a competitive parent may not. An AHD tree naturally records these parent–descendant relationships, so later expansions can retrospectively revise how promising earlier heuristics appear. MCTS-DPO Xie et al. (2024) and SVPO Chen et al. (2024) show that tree-



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

search outcomes can provide model-learning signals. EPS summarizes this descendant evidence as future search value, from which we derive continually updated search-feedback preferences.

Building on this observation, we propose **M**onte **C**arlo **T**ree **S**earch-Guided **E**volutionary-**P**otential-Aware Iterative **P**reference **L**earning (**MCTS-EPPL**), an iterative search–learning framework built upon MCTS-AHD. Adjacent on-policy preferences capture the immediate quality of newly generated candidates, whereas search-feedback preferences capture an ancestor's future search value through descendant-induced reversals in pairwise EPS order. These complementary signals fine-tune a local LLM through DPO, after which the updated model continues MCTS exploration. Experiments on multiple NP-hard combinatorial optimization tasks show that MCTS-EPPL outperforms comparable local-model baselines and representative AHD methods powered by stronger commercial LLM APIs.

Our main contributions are summarized as follows: (1) we propose MCTS-EPPL, which integrates Monte Carlo tree search with continual preference optimization of a locally trainable LLM; (2) we design a preference-oriented expansion strategy that, under a fixed evaluation budget, generates multiple comparable candidates under shared operator contexts; (3) we introduce EPS and an evolutionary-potential-aware preference mechanism that converts subtree-induced pairwise ranking reversals into search-feedback preferences while jointly exploiting adjacent on-policy performance preferences.

## 2 PRELIMINARIES

### 2.1 MONTE CARLO TREE SEARCH FOR AUTOMATIC HEURISTIC DESIGN

Monte Carlo Tree Search (MCTS) incrementally constructs a search tree by balancing the exploitation of promising nodes and the exploration of insufficiently visited ones. In MCTS-AHD Zheng et al. (2025), the root $n_r$ is a virtual node, while each non-root node $n$ stores an executable heuristic $h_n \in \mathcal{H}$, its linguistic description, a quality value $Q(n)$, and a visit count $N(n)$. The tree is initialized using actions $i1$ and $e1$, and then repeatedly performs selection, expansion, simulation, and backpropagation until the evaluation budget $T$ is exhausted.

*LLM-based Actions:* MCTS-AHD employs six prompt-based actions, each containing the task description, predefined algorithmic framework, and key-function specifications; actions other than $i1$ additionally use existing heuristics and their descriptions. Action $i1$ initializes a heuristic; $e1$ combines heuristics from different root subtrees; and $e2$ improves a parent using an elite reference. Action $m1$ modifies mechanisms or structure, whereas $m2$ refines parameters. Action $s1$ integrates useful designs along the selected root path. Complete prompts are provided in Appendix G.

*Selection:* Starting from the root, MCTS-AHD recursively selects the child with the largest Upper Confidence Bound for Trees (UCT) value:

$$\mathrm{UCT}(c) = \frac{Q(c) - q_{\min}}{q_{\max} - q_{\min}} + \lambda_t \sqrt{\frac{\ln(N(n)+1)}{N(c)}}, \tag{1}$$

where $c \in \mathrm{Children}(n)$, and $q_{\max}$ and $q_{\min}$ are the maximum and minimum quality values observed during search. The first term favors high-quality nodes, while the second encourages the exploration of less frequently visited nodes. The exploration factor decays linearly as

$$\lambda_t = \lambda_0 \frac{T - t}{T}, \tag{2}$$

where $t$ is the number of completed heuristic evaluations and $\lambda_0$ is the initial exploration factor.

*Progressive Widening:* MCTS-AHD allows previously expanded nodes to acquire additional children through progressive widening. A node $n$ can be expanded again when

$$\lfloor N(n)^{\alpha} \rfloor \geq |\mathrm{Children}(n)|\,, \tag{3}$$

where $\alpha$ controls how the branching factor grows with the visit count. When this condition is satisfied, $e1$ is applied to the virtual root and $e2$ is applied to a non-root node.

*Expansion:* For regular expansion, MCTS-AHD applies $e2$, $m1$, $m2$, and $s1$ to the selected leaf node. If each mutation action generates $k$ candidates and $e2$ and $s1$ each generate one candidate, the expansion produces $2k + 2$ new heuristics.

*Simulation:* Instead of performing stochastic rollouts, MCTS-AHD directly executes each generated heuristic $h$ on a task-specific evaluation dataset $\mathcal{D}$. Its evaluated performance $g(h)$ is assigned to the corresponding leaf node $n_\ell$:

$$Q(n_\ell) \leftarrow g(h), \qquad N(n_\ell) \leftarrow 1. \tag{4}$$

The evaluation results are also used to update the global quality bounds and the elite heuristic set.

*Backpropagation:* After leaf evaluation, quality and visit statistics are propagated toward the root. For each node $n$ on the selected ancestor path,

$$Q(n) \leftarrow \max_{c \in \mathrm{Children}(n)} Q(c), \qquad N(n) \leftarrow \sum_{c \in \mathrm{Children}(n)} N(c). \tag{5}$$

Thus, an internal node records the best quality among its children and the sum of their visit counts.

### 2.2 Direct Preference Optimization

Direct Preference Optimization (DPO) optimizes a language model directly from preference pairs without training a separate reward model Rafailov et al. (2023). In algorithm discovery, evaluation results collected during search can be converted into preference supervision, as adopted by EvoTune Surina et al. (2025) and Fine-tuning Large Language Model for Automated Algorithm Design Liu et al. (2025). Let $\pi_\theta(y \mid x)$ denote the trainable policy and $\pi_{\mathrm{ref}}(y \mid x)$ a fixed reference policy initialized from the base model.

A preference dataset is represented as

$$\mathcal{D}_{\mathrm{pref}} = \left\{ \left( x_i, y_i^+, y_i^- \right) \right\}_{i=1}^{M}, \tag{6}$$

where $y_i^+$ and $y_i^-$ are respectively the preferred and rejected algorithms under the same prompt $x_i$. In automated algorithm design, their ordering is typically determined by evaluated program performance.

Given preference pairs, the model is optimized using the standard DPO loss:

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\mathbb{E}_{(x,y^+,y^-)\sim\mathcal{D}_{\mathrm{pref}}} \left[ \log \sigma \left( \beta \left[ \log \frac{\pi_\theta(y^+ \mid x)}{\pi_{\mathrm{ref}}(y^+ \mid x)} - \log \frac{\pi_\theta(y^- \mid x)}{\pi_{\mathrm{ref}}(y^- \mid x)} \right] \right) \right]. \tag{7}$$

Here, $\sigma(\cdot)$ is the sigmoid function. Minimizing this loss increases the relative likelihood of the preferred algorithm while regularizing the policy with respect to the reference model.

## 3 Methodology

### 3.1 Overview

We propose MCTS-EPPL, an iterative framework that couples MCTS-based heuristic search with continual preference optimization of a local LLM. Each expansion yields two complementary learning signals: comparisons among co-generated candidates identify immediate quality, while descendant evidence retrospectively assesses future search value. MCTS statistics separately direct subsequent node selection. Following MCTS-AHD Zheng et al. (2025), the root is a virtual node, while each non-root node $v$ stores a complete heuristic $h_v$, together with its linguistic description or thought, executable implementation, direction-normalized immediate objective value $f(h_v)$, MCTS quality value $Q(v)$, and visit count $N(v)$. We further associate each node with an Evolutionary Potential Score (EPS) that characterizes its potential according to the outcomes of its generated descendants. The objective $f$ is unchanged for minimization tasks and negated for maximization tasks, so smaller values consistently indicate better performance, whereas $Q(v)$ is a separate node-level MCTS utility used for selection and backpropagation, with larger values indicating more promising nodes.

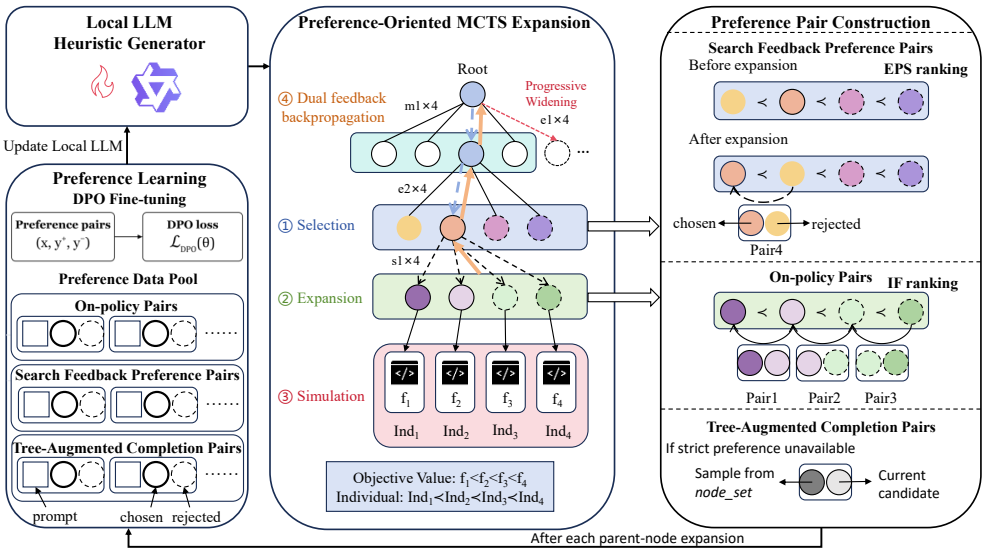



Figure 1: Overview of MCTS-EPPL. The local LLM generates multiple heuristic candidates during MCTS expansion. Adjacent candidates in the immediate-fitness ranking form on-policy preference pairs, while descendant-induced reversals in pairwise EPS order are converted into search-feedback preference pairs. Both types of preferences are used for DPO fine-tuning, after which the updated model continues MCTS search on the existing tree. The illustrated rankings follow the minimization convention.

As illustrated in Figure 1, MCTS selects a node for expansion, after which the local LLM generates multiple candidate heuristics under a shared operator context. Candidate evaluation updates MCTS statistics for subsequent node selection and updates the EPS values of affected ancestors. Comparisons of adjacent candidates in the immediate-objective ranking produce on-policy preferences for immediate quality, whereas reversals in pairwise EPS order produce search-feedback preferences for future search value and DPO fine-tuning.

### 3.2 Preference-Oriented MCTS Expansion

MCTS-EPPL retains the node-selection and progressive-widening mechanisms of MCTS-AHD Zheng et al. (2025), but modifies the expansion stage to produce multiple comparable candidates under the same input context. This controlled context makes preference comparisons attributable to generated heuristics rather than to prompt variation. For a parent node selected through regular MCTS selection, only one operator is applied per expansion. Let

$$\mathcal{O} = \{e2, m1, m2, s1\} \tag{8}$$

be the operator set, and let $w_o$ denote the weight of operator $o \in \mathcal{O}$. Its sampling probability, which controls the frequency of each evolutionary direction, is

$$P(o) = \frac{w_o}{\sum_{o' \in \mathcal{O}} w_{o'}}. \tag{9}$$

After selecting an operator, the current policy independently samples $n$ candidate heuristics using the same parent, operator, and input prompt; for $e2$, all candidates in the same expansion additionally share the same reference heuristic. All candidates are evaluated, and at most the $k_{\text{save}}$ valid candidates with the smallest $f(h)$ values are inserted into the MCTS tree, where

$$1 \leq k_{\text{save}} \leq n. \tag{10}$$

Other valid candidates do not participate in subsequent MCTS selection, but their generation and evaluation records remain available for EPS computation and preference construction.

When progressive widening is triggered, the operator logic follows MCTS-AHD: $e1$ is applied to the virtual root, whereas $e2$ is applied to a non-root node. The current policy likewise samples $n$ candidate heuristics under the same input context, but only the valid candidate with the smallest $f(h)$ is retained, preserving the one-child widening behavior of the original method. The remaining valid candidates are still used for EPS computation and preference construction. After evaluation, dual-feedback backpropagation updates the MCTS statistics $Q$ and $N$ along the ancestor path and recomputes the EPS values of affected nodes.

### 3.3 Evolutionary Potential Score

The immediate objective value $f(h)$ measures only the current quality of a heuristic and may not reflect its ability to generate strong descendants. We therefore define EPS by aggregating descendant performance across different generation depths. Let $\mathcal{G}_t$ denote all valid candidate records $u$ generated up to search step $t$, including those not retained by the top-$k_{\text{save}}$ rule, where $h_u$ denotes the heuristic stored in record $u$. We exclude a candidate from EPS computation if $f(h_u)$ is identical to that of its parent heuristic or, when generated by $e2$, identical to that of the reference heuristic.

For a node $v$, its remaining descendant records at depth $\ell$ are

$$\mathcal{D}_\ell^t(v) = \{u \in \mathcal{G}_t \mid \text{dist}_{\text{gen}}(v, u) = \ell\}, \tag{11}$$

where $\text{dist}_{\text{gen}}(v, u)$ is the distance defined by the recorded generation relationships. EPS therefore considers all eligible generated heuristics rather than only formal tree nodes.

The maximum observed descendant depth is

$$D_t(v) = \max\left\{\ell : \mathcal{D}_\ell^t(v) \neq \varnothing\right\}. \tag{12}$$

At each depth, the best and average descendant objective values are combined:

$$S_\ell^t(v) = \alpha_{\text{EPS}} \min_{u \in \mathcal{D}_\ell^t(v)} f(h_u) \; + (1 - \alpha_{\text{EPS}}) \frac{1}{|\mathcal{D}_\ell^t(v)|} \sum_{u \in \mathcal{D}_\ell^t(v)} f(h_u), \tag{13}$$

where $\alpha_{\text{EPS}} \in [0, 1]$ balances the best and average descendant objective values at each depth.

The EPS of node $v$ is defined as

$$\text{EPS}_t(v) = \begin{cases} \dfrac{\sum_{\ell=1}^{D_t(v)} \beta_{\text{EPS}}^{\ell-1} S_\ell^t(v)}{\sum_{\ell=1}^{D_t(v)} \beta_{\text{EPS}}^{\ell-1}}, & D_t(v) \geq 1, \\ f(h_v), & D_t(v) = 0, \end{cases} \tag{14}$$

where $\beta_{\text{EPS}} \in (0, 1]$ discounts distant descendants, following the depth-attenuated credit assignment used in Clade-AHD Lai et al. (2026). Under the direction-normalized objective convention, a smaller EPS indicates stronger evolutionary potential; if no eligible descendant exists, EPS reduces to the immediate objective value $f(h_v)$ of the heuristic stored at the node. Consider an ancestor whose immediate objective is worse than that of a comparable peer. A newly generated strong descendant can lower the ancestor's EPS below that peer's and reverse their local ranking; this reversal, rather than an isolated EPS value, produces a search-feedback preference.

### 3.4 On-Policy Preference Pair Construction

On-policy preference pairs are constructed from candidates generated by the current model during the same node expansion. Because they share the parent, operator, and input prompt, their objective values $f(h)$ are directly comparable. All valid candidates are ranked, including those not retained as MCTS nodes. We construct pairs only between adjacent candidates in this ranking, rather than enumerating all pairwise combinations. For each adjacent pair with different $f(h)$ values, the candidate with the smaller value is chosen and the other rejected, forming a DPO example with their shared prompt. Adjacent ties establish no strict preference, and invalid candidates are excluded.

**Search-Tree-Augmented Completion for On-Policy Candidates.** For a current on-policy candidate without a strict local preference (invalid or tied), we use it as rejected and retrieve a distinguishable tree response from the elite set `node_set` as chosen. After excluding the parent, current candidate, $e2$ reference, objective ties, and duplicate code, eligible responses are rank-sampled to favor

better objective values. Stochastic sampling avoids repeatedly selecting the global best response and promotes diverse competitive patterns while retaining a bias toward high-quality heuristics, thereby reducing premature concentration on one heuristic family. A strictly better response is preferred; otherwise, a different-objective elite response provides a diversity-oriented alternative rather than a strict performance preference. The pair uses the current expansion prompt. Appendix F.4 gives the complete eligibility and sampling rules.

### 3.5 Search-Feedback Preference Pair Construction

Search-feedback preferences complement on-policy preferences: the latter compare the immediate quality of newly generated candidates, whereas the former capture the descendant-based future search value of existing heuristics. Before expansion, the EPS rankings of affected ancestor nodes are recorded within neighborhoods containing nodes with the same input prompt. After candidate evaluation, EPS is recomputed along the ancestor path and the corresponding post-expansion rankings are obtained.

For any pair of comparable nodes, a search-feedback preference pair is constructed only if their relative order strictly reverses between the pre- and post-expansion EPS rankings. The node with the smaller post-expansion EPS is treated as chosen and the other as rejected. A change in an individual rank is therefore insufficient when the pairwise ordering is preserved; ties in either ranking also produce no pair. For example, if the pre-expansion order is $1 \prec 2 \prec 3 \prec 4$, the post-expansion order $2 \prec 1 \prec 3 \prec 4$ yields only $2 \prec 1$, whereas $2 \prec 3 \prec 1 \prec 4$ yields $2 \prec 1$ and $3 \prec 1$. This mechanism converts descendant-induced reversals in the search direction into parameter-level preference supervision.

### 3.6 Iterative Preference Learning

After each parent-node expansion, the system collects the on-policy and search-feedback preference pairs generated since the previous model update. When the resulting training set is non-empty, the local model is fine-tuned using DPO. To reduce the computational and memory costs of continual training, we employ LoRA-based parameter-efficient fine-tuning, keeping the base-model parameters frozen and updating only the LoRA adapter weights, consistent with CALM Huang et al. (2025b).

After fine-tuning, the updated LoRA weights are loaded into the local heuristic generator. The existing MCTS tree is preserved together with its node relationships, candidate records, and descendant information. The updated model then continues heuristic generation and tree expansion, forming the iterative search–learning process shown in Figure 1.

Table 1: ACO-based heuristic performance on OP, CVRP, and MKP (mean of three runs). Arrows indicate the better direction; gray marks the best LLM-based AHD result and bold the best overall result in each column.

| | OP | | | CVRP | | | MKP | | |
|---|---|---|---|---|---|---|---|---|---|
| Methods | $N = 50$ Obj.↑ | $N = 100$ Obj.↑ | $N = 200$ Obj.↑ | $N = 50$ Obj.↓ | $N = 100$ Obj.↓ | $N = 200$ Obj.↓ | $N = 100$ Obj.↑ | $N = 200$ Obj.↑ | $N = 500$ Obj.↑ |
| ACO | 14.808 | 29.100 | 49.402 | 11.422 | 18.871 | 34.855 | 22.714 | 40.681 | 95.486 |
| DeepACO | 14.438 | 29.490 | 51.829 | **8.899** | **14.884** | **26.973** | 23.086 | 41.973 | 101.067 |
| *LLM-based AHD: GPT-4o-mini* | | | | | | | | | |
| EoH | 14.694 | 29.921 | 52.768 | 9.478 | 15.721 | 28.026 | 22.951 | 42.097 | 102.771 |
| ReEvo | 14.901 | 30.481 | 52.944 | 9.682 | 16.084 | 28.593 | 22.587 | 41.287 | 99.962 |
| MCTS-AHD | 14.436 | 29.503 | 53.064 | 9.811 | 15.703 | 27.645 | 22.907 | 41.832 | 102.684 |
| CALM | 14.944 | 30.558 | 52.821 | 9.397 | 15.648 | 28.244 | 23.012 | 42.118 | 102.743 |
| *LLM-based AHD: Qwen2.5-7B-Instruct-INT4* | | | | | | | | | |
| EoH | 14.896 | 30.478 | 55.179 | 9.224 | 15.812 | 28.351 | **23.180** | 42.126 | 102.887 |
| ReEvo | 14.812 | 30.529 | 55.217 | 10.219 | 16.982 | 29.503 | 22.926 | 42.093 | 102.996 |
| MCTS-AHD | 14.716 | 30.503 | 55.146 | 9.231 | 15.618 | 27.672 | 23.067 | 42.214 | 103.074 |
| *Local-model adaptation (with fine-tuning)* | | | | | | | | | |
| EvoTune | 14.236 | 29.831 | 53.417 | 9.985 | 16.898 | 29.551 | 22.325 | 40.571 | 99.720 |
| CALM | 14.982 | 30.641 | 55.613 | 9.561 | 16.164 | 28.736 | 23.017 | 42.024 | 102.331 |
| **MCTS-EPPL (Ours)** | **15.063** | **30.917** | **55.919** | 9.202 | 15.427 | 27.091 | 23.108 | **42.244** | **103.438** |

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**Optimization Problems.** We evaluate MCTS-EPPL on four representative NP-hard combinatorial optimization problems: the Orienteering Problem (OP), Capacitated Vehicle Routing Problem (CVRP), Multiple Knapsack Problem (MKP), and Traveling Salesman Problem (TSP). OP, CVRP, and MKP are evaluated under the Ant Colony Optimization (ACO) framework, where the LLM designs task-specific heuristic measures for guiding solution construction Dorigo et al. (2006); Ye et al. (2023). TSP is evaluated under the Guided Local Search (GLS) framework, where the generated heuristic constructs the knowledge-guided penalty matrix used by KGLS Voudouris & Tsang (1999); Arnold & Sörensen (2019). Detailed task definitions and instance distributions are provided in Appendix C.

**Baselines.** We compare MCTS-EPPL with ACO Dorigo et al. (2006), DeepACO Ye et al. (2023), EoH Liu et al. (2024), ReEvo Ye et al. (2024), MCTS-AHD Zheng et al. (2025), EvoTune Surina et al. (2025), and CALM Huang et al. (2025b). EvoTune reports TSP-GLS but not ACO-guided heuristics, so we rerun it under our common protocol Surina et al. (2025). Its original 22,400-output budget exceeds our 1,000-evaluation budget; we thus generate four candidates per iteration rather than eight, matching CALM and MCTS-EPPL. Since its original 400-iteration update interval would not trigger here, we update every iteration for one DPO epoch rather than two. For TSP-GLS, we additionally compare VRP-DACT Ma et al. (2021), NeuOpt Ma et al. (2023), NeuralGLS Sui et al. (2024), and GNNGLS Hudson et al. (2022). All LLM-based AHD methods use the same seed heuristics, heuristic-evaluation datasets, and task-specific solver configurations; the fixed heuristic interfaces and solver settings are detailed in Appendix D.

**Heuristic Selection and Testing.** Following the single-output evaluation convention of MCTS-AHD and CALM Zheng et al. (2025); Huang et al. (2025b), each independent search run returns exactly one heuristic: the best-so-far candidate according to its objective on the fixed heuristic-evaluation dataset $\mathcal{D}$. This selection is completed before any test instance is accessed. The returned heuristic is then evaluated on each corresponding test set, and results are averaged over the three independently returned heuristics. Test performance is never used to select candidates or to choose among runs.

Table 2: Task-specific MCTS expansion settings. Each expansion generates $n = 4$ candidates. Operator weights are listed as $w_{e2} : w_{m1} : w_{m2} : w_{s1}$.

| Problem | Weights | $k_{\text{save}}$ |
|---|---|---|
| OP-ACO | $4:2:1:1$ | 2 |
| CVRP-ACO | $1:1:1:1$ | 2 |
| MKP-ACO | $4:2:1:1$ | 4 |
| TSP-GLS | $1:2:2:1$ | 2 |

**Implementation Settings.** We use an INT4-quantized Qwen2.5-7B-Instruct model as the local heuristic generator Qwen Team (2024). DPO fine-tuning is performed using the Unsloth framework Han et al. (2023) with LoRA adapters Hu et al. (2022), updating approximately 1.05% of the model's parameters; detailed training and inference settings are provided in Appendix F.

Following the notation introduced in Section 3.2, each MCTS expansion generates $n = 4$ candidates. Task-specific operator weights and retained-candidate counts are listed in Table 2.

The EPS parameters are set to $\alpha_{\text{EPS}} = 1.0$ and $\beta_{\text{EPS}} = 0.8$.

Table 3: TSP-GLS mean optimality gaps (%) over three independent runs; lower values are better.

| | TSP-GLS | | |
|---|---|---|---|
| Methods | $N = 100$ | $N = 200$ | $N = 500$ |
| Optimal | 0.0000% | 0.0000% | 0.0000% |
| KGLS | **0.0034%** | 0.2270% | 0.9578% |
| *NCO methods with the GLS general framework* | | | |
| VRP-DACT | 1.7943% | 91.9267% | – |
| NeuOpt | 0.2950% | 0.9152% | – |
| NeuralGLS* | 0.4700%* | 3.6220%* | – |
| GNNGLS* | 0.7050%* | 3.5220%* | – |
| *LLM-based AHD: GPT-4o-mini* | | | |
| EoH | 0.0065% | **0.2025%** | 0.9534% |
| ReEvo | 0.0076% | 0.2210% | 0.9993% |
| MCTS-AHD | 0.0060% | 0.2106% | 0.9495% |
| CALM | 0.0082% | 0.2399% | 0.9132% |
| *LLM-based AHD: Qwen2.5-7B-Instruct-INT4* | | | |
| EvoTune | 0.0092% | 0.2575% | 0.9804% |
| CALM | 0.0063% | 0.2165% | 0.8956% |
| **MCTS-EPPL (Ours)** | 0.0055% | 0.2103% | **0.8469%** |

*Values follow the corresponding ReEvo comparison.

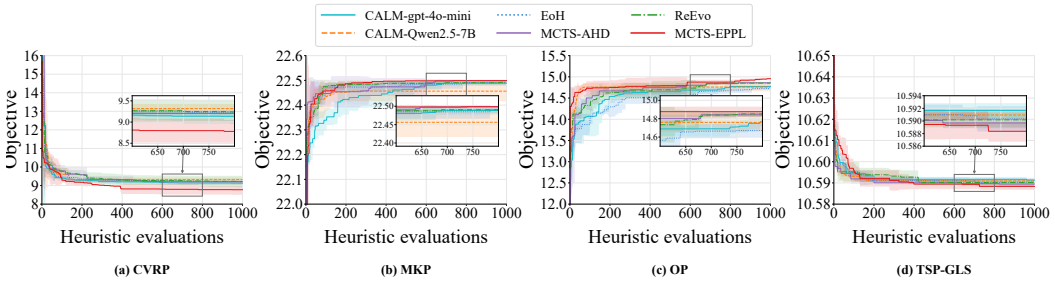



Figure 2: Convergence curves over heuristic evaluations for (a) CVRP, (b) MKP, (c) OP, and (d) TSP-GLS, averaged over three independent runs. Shaded bands show standard deviations, and insets highlight the late-evaluation region. Higher objectives are better for MKP and OP; lower objectives are better for CVRP and TSP-GLS.

The MCTS-related parameters are set to $N_I = 4$, $\lambda_0 = 0.1$, and $\alpha_{\text{PW}} = 0.25$, where $N_I$ is the number of initial tree nodes, $\lambda_0$ is the initial exploration coefficient, and $\alpha_{\text{PW}}$ is the progressive-widening coefficient.

The maximum evaluation time for each heuristic is 60 seconds. For all problems and all LLM-based AHD methods, the total heuristic-evaluation budget is fixed to $T = 1000$.

### 4.2 Overall Performance

**ACO-Based Heuristic Design.** Under the local-model setting, MCTS-EPPL achieves the best overall result in five of the nine task–scale settings and the best LLM-based AHD result in eight of nine (Table 1). Despite using a locally deployed quantized 7B model, it outperforms all GPT-4o-mini-based AHD baselines across all nine settings. Its best overall results cover all OP scales and MKP with $N = 200$ and $N = 500$. DeepACO remains strongest across the three CVRP scales, whereas the Qwen2.5-7B-based EoH variant attains the best overall result on MKP with $N = 100$.

**Search Convergence.** Figure 2 indicates gains in both early-search efficiency and late-stage solution quality. MCTS-EPPL improves rapidly early in search and continues to improve after most competing methods plateau. By the late stage, it attains the lowest mean CVRP objective and the highest mean OP objective. EoH, ReEvo, and MCTS-AHD use GPT-4o-mini via API; CALM is evaluated with both GPT-4o-mini and the locally deployed Qwen2.5-7B-Instruct-INT4 model, which is also used by MCTS-EPPL. Figure 2 reports the same convergence view across all four tasks.

**GLS-Based Heuristic Design.** Table 3 shows that MCTS-EPPL outperforms CALM and Evo-Tune at all three TSP scales. Its gaps are $0.0055\%$, $0.2103\%$, and $0.8469\%$, compared with Evo-Tune's $0.0092\%$, $0.2575\%$, and $0.9804\%$, respectively. MCTS-EPPL achieves the best LLM-based result at $N = 100$ and $N = 500$, and ranks second at $N = 200$, behind EoH ($0.2025\%$). At $N = 500$, it also yields the lowest gap among all compared methods except the Optimal reference.

### 4.3 Ablation Study

We conduct ablation experiments on OP-ACO to examine the contributions of the preference-learning components, the number of retained candidates, and the operator-weight configuration (Table 4). All reported relative gaps are averaged over five independent runs.

**Learning and Preference Components.** Removing DPO yields the largest gap ($2.277\%$), followed by removing on-policy pairs ($1.127\%$). Removing tree-augmented completion or search-feedback pairs yields gaps of $0.503\%$ and $0.695\%$, respectively, supporting complementary roles. The no-DPO variant also trails search-only MCTS-AHD, so search modifications alone do not explain the full gain. Among completion pairs, 26.6% favor a heuristic with worse immediate performance (Figure 3), providing supervision beyond immediate quality.

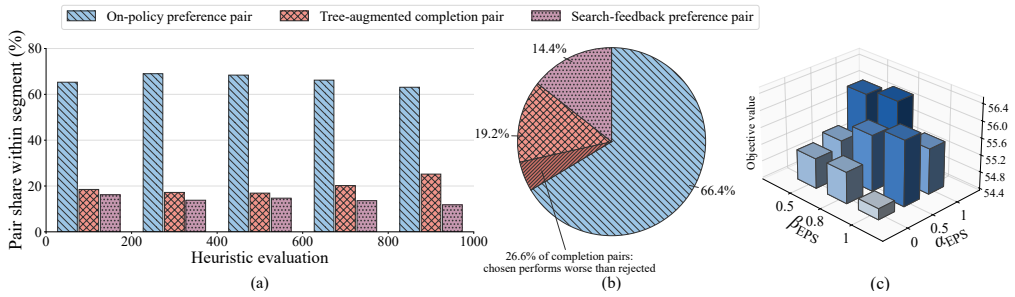



Figure 3: Preference-pair composition and EPS sensitivity on OP-ACO. Panels (a) and (b) show stage-wise and overall pair composition; panel (c) shows EPS sensitivity. In (b), the darker completion slice marks the 26.6% of completion pairs whose chosen heuristic performs worse than the rejected heuristic in the immediate evaluation. Results are averaged over three runs.

**Number of Retained Candidates.** The full setting with $k_{\text{save}} = 2$ performs best. Retaining one or four candidates yields gaps of $0.650\%$ and $2.355\%$; the latter also raises SD to $2.790$ percentage points versus $0.647$ for the full setting. Thus, two candidates balance selection and diversity.

**Operator Weights.** The proposed $4:2:1:1$ weights perform best. MCTS-AHD's $1:2:2:1$ weights Zheng et al. (2025) and uniform weights yield gaps of $0.865\%$ and $1.427\%$; removing $e2$ causes the largest gap ($2.435\%$). Table 4 orders weights as $w_{e2} : w_{m1} : w_{m2} : w_{s1}$.

**Sensitivity to EPS Parameters.** Figure 3(c) favors $\alpha_{\text{EPS}} = 1$ and $\beta_{\text{EPS}} = 0.8$. The former selects the best descendant instead of averaging descendant objectives; the latter discounts deeper evidence. Across tested $\alpha_{\text{EPS}}$ values, $\beta_{\text{EPS}} = 1$ performs worse than smaller values.

Table 4: OP-ACO ablations over five runs. Gap is the mean percentage difference from the full-setting mean objective (lower is better); SD is the standard deviation of run-level gaps (percentage points).

| Ablation | Setting | Gap (%)↓ | SD |
|---|---|---|---|
| *Learning and preference components* | | | |
| MCTS-EPPL (full) | All components | **0.000%** | **0.647%** |
| w/o on-policy pairs | Removed | 1.127% | 1.360% |
| w/o tree-augmented completion | Removed | 0.503% | 1.071% |
| w/o search-feedback pairs | Removed | 0.695% | 0.833% |
| w/o DPO | No fine-tuning | 2.277% | 1.597% |
| MCTS-AHD | Search-only | 1.634% | 1.300% |
| *Retained candidates* | | | |
| $k_{\text{save}} = 1$ | Full weights | 0.650% | 0.958% |
| $k_{\text{save}} = 2$ | Full setting | **0.000%** | **0.647%** |
| $k_{\text{save}} = 4$ | Full weights | 2.355% | 2.790% |
| *Operator weights* | | | |
| Full weights | $4:2:1:1$ | **0.000%** | **0.647%** |
| MCTS-AHD weights | $1:2:2:1$ | 0.865% | 0.896% |
| Uniform weights | $1:1:1:1$ | 1.427% | 1.271% |
| w/o `e2` | $0:2:1:1$ | 2.435% | 1.858% |
| w/o `m1` | $4:0:1:1$ | 0.864% | 1.112% |
| w/o `m2` | $4:2:0:1$ | 1.070% | 1.345% |
| w/o `s1` | $4:2:1:0$ | 1.322% | 1.620% |

**Computational Cost.** Search takes 1.44 hours for CALM, 3.18 for MCTS-EPPL, and 13.25 for EvoTune on the same hardware. Table 5 in the appendix reports mean/peak incremental fine-tuning memory of 2.1/2.5, 2.2/2.7, and 3.9/6.2 GB in the same order. EvoTune's accumulated preference pairs can lengthen later updates, although accumulation alone need not increase peak memory with fixed microbatches.

## 5 Conclusion

We presented MCTS-EPPL, an iterative search–learning framework for LLM-based automatic heuristic design. It treats the search tree as both an exploration structure and a source of training supervision, combining on-policy preferences for current quality with descendant-based preferences for future search value. Across four NP-hard problems, MCTS-EPPL achieves the best LLM-based result in 10 of 12 task–scale settings and remains close in the other two settings. Ablations support the preference-learning components and the selected search configuration. The evidence is limited to the evaluated domains and model scales.

## AI USE STATEMENT

Generative language models were used as experimental components of the proposed framework, as described in the main text. In the research workflow, we used generative AI tools to provide feedback on experimental design and to assist with language polishing. All research questions, methodological decisions, experimental analyses, interpretations of results, and conclusions were made and verified by the authors. We reviewed all AI-assisted revisions and take responsibility for the final content of this work, including all text, claims, and artifacts.

## REFERENCES

Florian Arnold and Kenneth Sörensen. Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research*, 105:32–46, 2019. doi: 10.1016/j.cor.2019.01.002.

Edmund K Burke, Mathew R Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R Woodward. Exploring hyper-heuristic methodologies with genetic programming. In *Computational intelligence: Collaboration, fusion and emergence*, pp. 177–201. Springer, 2009. doi: 10.1007/978-3-642-01799-5_6.

Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013. doi: 10.1057/jors.2013.71.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Step-level value preference optimization for mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7889–7903, 2024. doi: 10.18653/v1/2024.findings-emnlp.463. URL https://aclanthology.org/2024.findings-emnlp.463/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374v2.

Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006. doi: 10.1109/MCI.2006.329691.

Daniel Han, Michael Han, and Unsloth Team. Unsloth. https://github.com/unslothai/unsloth, 2023. Software repository; accessed 2026-09-03.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. ORLM: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 73(6):2986–3009, 2025a. doi: 10.1287/opre.2024.1233.

Ziyao Huang, Weiwei Wu, Kui Wu, Jianping Wang, and Wei-Bin Lee. CALM: Co-evolution of algorithms and language model for automatic heuristic design. *arXiv preprint arXiv:2505.12285*, 2025b. URL https://arxiv.org/abs/2505.12285v1.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ar92oEosBIg.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, 35 (2):1–72, 2026. doi: 10.1145/3747588.

Kezhao Lai, Yutao Lai, and Hai-Lin Liu. Beyond the node: Clade-level selection for efficient mcts in automatic heuristic design. *arXiv preprint arXiv:2602.00549*, 2026. URL https://arxiv.org/abs/2602.00549v2.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024. URL https://arxiv.org/abs/2401.02051v3.

Fei Liu, Rui Zhang, Xi Lin, Zhichao Lu, and Qingfu Zhang. Fine-tuning large language model for automated algorithm design. *arXiv preprint arXiv:2507.10614*, 2025. URL https://arxiv.org/abs/2507.10614v2.

Fei Liu, Yilu Liu, Qingfu Zhang, Tong Xialiang, and Mingxuan Yuan. EoH-S: Evolution of heuristic set using LLMs for automated heuristic design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 37090–37098, 2026a. doi: 10.1609/aaai.v40i43.41038. URL https://ojs.aaai.org/index.php/AAAI/article/view/41038.

Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Zhe Zhao, Xialiang Tong, Kun Mao, Zhichao Lu, Zhenkun Wang, et al. A systematic survey on large language models for algorithm design. *ACM Computing Surveys*, 58(8):1–32, 2026b. doi: 10.1145/3787585.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In *Advances in Neural Information Processing Systems*, volume 34, pp. 11096–11107, 2021. URL https://proceedings.neurips.cc/paper/2021/file/5c53292c032b6cb8510041c54274e65f-Paper.pdf.

Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49555–49578, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9bae70d354793a95fa18751888cea07d-Paper-Conference.pdf.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131v1.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract.html.

Jessica Coto Palacio, Yailen Martínez Jiménez, Leander Schietgat, Bart Van Doninck, and Ann Nowé. A q-learning algorithm for flexible job shop scheduling in a real-world manufacturing scenario. *Procedia CIRP*, 106:227–232, 2022. doi: 10.1016/j.procir.2022.02.183.

Rafael HM Pereira, Marcus Saraiva, Daniel Herszenhut, Carlos Kaue Vieira Braga, and Matthew Wigginton Conway. r5r: Rapid realistic routing on multimodal transport networks with $R^5$ in R. *Findings*, 2021. doi: 10.32866/001c.21262. URL https://doi.org/10.32866/001c.21262.

Qwen Team. Qwen2.5 Technical Report, 2024. URL https://arxiv.org/abs/2412.15115v2. arXiv version 2.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741, 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300v3.

Jingyan Sui, Shizhe Ding, Boyang Xia, Ruizhi Liu, and Dongbo Bu. Neuralgls: learning to guide local search with graph convolutional network for the traveling salesman problem. *Neural Computing and Applications*, 36(17):9687–9706, 2024. doi: 10.1007/s00521-023-09042-6.

Anja Surina, Amin Mansouri, Lars Quaedvlieg, Amal Seddas, Maryna Viazovska, Emmanuel Abbe, and Caglar Gulcehre. Algorithm discovery with llms: Evolutionary search meets reinforcement learning. *arXiv preprint arXiv:2504.05108*, 2025. URL https://arxiv.org/abs/2504.05108v4.

Chee Sheng Tan, Rosmiwati Mohd-Mokhtar, and Mohd Rizal Arshad. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*, 9: 119310–119342, 2021. doi: 10.1109/ACCESS.2021.3108177.

Giulia Tresca, Graziana Cavone, Raffaele Carli, Antonio Cerviotti, and Mariagrazia Dotoli. Automating bin packing: A layer building matheuristics for cost effective logistics. *IEEE Transactions on Automation Science and Engineering*, 19(3):1599–1613, 2022. doi: 10.1109/TASE.2022.3177422.

Christos Voudouris and Edward Tsang. Guided local search and its application to the traveling salesman problem. *European journal of operational research*, 113(2):469–499, 1999. doi: 10.1016/S0377-2217(98)00099-X.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024. URL https://arxiv.org/abs/2405.00451v2.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. DeepACO: Neural-enhanced ant systems for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43706–43728, 2023. doi: 10.52202/075280-1895. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/883105b282fe15275991b411e6b200c5-Abstract-Conference.html.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. Reevo: Large language models as hyper-heuristics with reflective evolution. *Advances in neural information processing systems*, 37:43571–43608, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4ced59d480e07d290b6f29fc8798f195-Abstract-Conference.html.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. *arXiv preprint arXiv:2501.08603*, 2025. URL https://arxiv.org/abs/2501.08603v3.