WHEN AND HOW TO REFLECT: STATE-AWARE MULTI-SCALE REFLECTION FOR LLM-BASED AUTOMATIC HEURISTIC DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

LLMs can search directly over executable heuristics, but effective program generation still depends on providing the right evidence about what should be improved. Island-based search maintains multiple independently evolving subpopulations of candidate programs, called *islands*, enabling parallel exploration of different regions of the search space. Existing frameworks such as FunSearch and OpenEvolve maintain multiple independently evolving islands and promote diversity through mechanisms such as MAP-Elites, migration, and the selection of both parent and inspiration programs. We instead ask a complementary question in this paper: how should the evidence presented to each island change as its search progresses, stagnates, or requires a restart? We propose SAMR (State-Aware Multi-Scale Reflection), a framework that personalizes reflective evidence across islands and throughout the search process. Each island tracks its momentum, stagnation, and age, and dynamically transitions among three states: EXPLOIT, EXPLORE, and RESET. These states determine access to different forms of evidence, including case contrasts, empirical edit-pattern statistics, and cross-island knowledge. On 36 CO-Bench problem types, SAMR with Claude Sonnet 4.5 achieves a mean normalized score of 0.853, outperforming FunSearch (0.818), OpenEvolve (0.811), ReEvo (0.779), and MCTS-AHD (0.747). Although CORAL scores marginally higher (0.863), it adopts a fundamentally different agent-based framework and requires 26.6× more tokens on average. Using DeepSeek V3, SAMR also ranks first in mean score among all evaluated methods. Overall, our results demonstrate that state-dependent routing of diagnostic evidence, including long-term and cross-island reflections, improves the quality–cost tradeoff in LLM-guided program search.

1 INTRODUCTION

Combinatorial optimization (CO) problems arise naturally in logistics, scheduling, resource allocation, routing, packing, and graph optimization. Many of these problems are NP-hard, making exact optimization computationally prohibitive at realistic scales. In practice, strong performance therefore often relies on problem-specific heuristics, such as first-fit decreasing for bin packing, dispatching rules for job-shop scheduling, and specialized procedures for vehicle routing. Designing and tuning such heuristics, however, typically requires substantial domain expertise and manual experimentation. Automatic heuristic design (AHD) aims to reduce this burden by systematically generating and improving heuristic algorithms. Classical AHD approaches include genetic programming and hyper-heuristics, but they are often limited by their reliance on human-designed primitives and low-level search perators (Burke et al., 2013).

LLMs expand the search space of AHD from predefined heuristic components to executable programs expressed in general-purpose code. For example, FunSearch combines an LLM-based mutation operator with an evaluator and an island-based program database (Romera-Paredes et al., 2024). EoH co-evolves natural-language heuristic ideas and their code implementations via multiple operators (Liu et al., 2024), while ReEvo augments the search with pairwise and accumulated verbal reflection (Ye et al., 2024). MCTS-AHD organizes program generation within a UCT-guided search tree (Zheng et al., 2025). More recent systems further adapt search regimes, prompt operators, mi-

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.

gration, resource allocation, and even the search strategy itself (Qi et al., 2025; Chen et al., 2026; Cemri et al., 2026).

We ask a more focused question: *given one island's search trajectory, what diagnostic evidence should be included in its next prompt?* Search control and evidence selection are related but distinct. A method may switch from mutation to crossover while presenting the same evidence, whereas two islands using the same operator may require different diagnostics. An improving island may benefit most from local attribution, a stagnating island from patterns discovered elsewhere, and a lagging island from a restart that still preserves knowledge of previously unsuccessful edits.

We propose SAMR (State-Aware Multi-Scale Reflection), which routes diagnostic evidence along two complementary dimensions. Across islands, each island receives evidence conditioned on its own trajectory and edit history. Across time, the composition of this evidence adapts as the island transitions among the EXPLOIT, EXPLORE, and RESET states. We make four contributions:

- We formulate prompt construction as an island-level evidence-routing problem. Rather than adapting only the search operator or sampling intensity, the trajectory controller determines the *type and source of evidence* included in each island's prompt.
- We develop three complementary scales of reflection: case-level contrasts for local diagnostic attribution, pattern-level edit statistics for learning over time, and cross-island contrasts for transferring knowledge across the population. The edit-pattern categories are induced online from observed code differences.
- We introduce a knowledge-preserving reset mechanism. Although reset itself is not new (e.g., FunSearch periodically reseeds weak islands), SAMR triggers resets based on an island's trajectory and preserves negative knowledge about previously unsuccessful or deprecated edit patterns.
- We evaluate SAMR using two LLM backbones on 36 CO-Bench problem types, referred to throughout the paper as tasks, spanning eight problem families. Across these experiments, SAMR outperforms the evaluated implementations of FunSearch, ReEvo, MCTS-AHD, and OpenEvolve and achieves a favorable quality–token tradeoff relative to CORAL under a matched evaluator-query budget.

2 RELATED WORK AND POSITIONING

2.1 LLM-BASED AUTOMATIC HEURISTIC DESIGN

Automated heuristic design driven by LLMs has advanced rapidly in recent years (Liu et al., 2026b). FunSearch (Romera-Paredes et al., 2024) serves as our primary island-based baseline. It maintains island-local elites, uses an LLM to mutate a designated function, evaluates the resulting programs, and periodically reseeds under-performing islands. EoH evolves both heuristic ideas and executable code through multiple evolutionary operators (Liu et al., 2024). Extensions of this line vary *what* is produced: MEoH reframes heuristic design as multi-objective search, evolving a non-dominated set of heuristics that trade off solution quality against efficiency (Yao et al., 2025), while EoH-S evolves a small complementary heuristic *set* so that diverse instances are each served by at least one member, improving generalization across distributions (Liu et al., 2026a). ReEvo constructs short-term feedback by contrasting parent heuristics and accumulates verbal reflections to guide subsequent elitist mutations (Ye et al., 2024). Accordingly, neither island-based program evolution nor reflection accumulated over time is new in isolation.

HSEvo combines harmony search with reflective genetic operators to balance diversity and convergence (Dat et al., 2025). EoH-MR adapts exploration, exploitation, and operator selection using signals such as population diversity, performance gaps, and stagnation (Qi et al., 2025). Similarly, MCTS-AHD is stateful: its tree statistics determine which node and search action to expand (Zheng et al., 2025). These methods primarily adapt *where* the search proceeds and *how* candidate programs are generated, while also promoting diversity. In contrast, SAMR focuses on *what diagnostic evidence* is supplied to the generator for each island.

2.2 Adaptive and Agentic Program Evolution

Recent systems introduce increasingly adaptive forms of program evolution. The global navigator in HiFo-Prompt uses progress and diversity signals to switch search regimes and stores distilled insights from prior iterations (Chen et al., 2026). AdaEvolve adapts both local search behavior and the allocation of global computational resources, with meta-guidance derived from accumulated improvements (Cemri et al., 2026). SAMR differs from these approaches in both granularity and target: it makes island-specific decisions about whether to provide case-level, historical pattern-level, or cross-island evidence.

AlphaEvolve extends evaluator-guided evolution to larger and more general code artifacts (Novikov et al., 2025), while OpenEvolve provides an open and scalable implementation supporting quality-diversity search and multi-objective evaluation (Sharma, 2025). These system-level capabilities are largely orthogonal to the evidence-routing problem studied here and could, in principle, be combined with SAMR.

CORAL adopts a fundamentally different, agent-based design in which long-running agents retrieve information, reflect on outcomes, evaluate candidate solutions, and exchange knowledge through shared persistent memory (Qu et al., 2026). We therefore compare CORAL in terms of both solution quality and token cost, without suggesting that autonomous agentic reflection is intrinsically inferior. Appendix A summarizes the methodological comparison.

3 State-Aware Multi-Scale Reflection

3.1 Overview

SAMR preserves the standard island-based evolutionary loop. For each valid program, the evaluator returns both an aggregate score and a vector of case-level scores. The system then records, for each island, its recent performance trajectory and the outcomes of parent-to-child edits. Before generating a new child, a controller assigns the island to a search state and constructs the corresponding state-specific evidence bundle. Figure 1 summarizes the overall pipeline, from trajectory-driven state selection to multi-scale reflection, prompt construction, and evaluation.

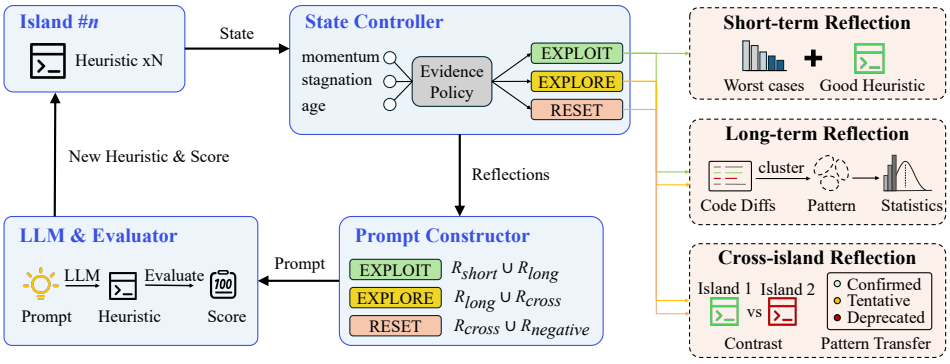

Figure 1: Overview of SAMR. For each island, a controller reads the recent trajectory state $\tau_i(t)$ and assigns one of three search states (Exploit, Explore, Reset), which in turn selects the evidence bundle $\mathcal{R}_{\text{Exploit/Explore/Reset}}$ from three reflection scales: short-term case contrasts, long-term island-local edit patterns, and cross-island contrastive samples and pattern transfer. The prompt constructor assembles the parent program with the selected evidence; the LLM proposes a child, and the evaluator returns aggregate and case-level scores that update the island's trajectory and knowledge base.

3.2 Trajectory State and Evidence Policy

For island i at generation t, define its trajectory state as

$$\tau_i(t) = \big(m_i(t), z_i(t), a_i(t)\big),$$

where

$$m_i(t) = \frac{1}{W} \sum_{u=t-W+1}^{t} \left(f_{i,u}^{\text{child}} - f_{i,u}^{\text{parent}}\right)$$

is the island's momentum, measured as the average child-to-parent score improvement over a window of length W. The stagnation statistic $z_i(t)$ records the number of consecutive generations without a meaningful improvement, while the age $a_i(t)$ measures the time elapsed since the island was created or last reset. We also track the island's relative lag, defined as its performance gap from the current population leader.

The controller first evaluates a reset condition triggered by severe relative lag or prolonged stagnation. If no reset is required, positive momentum places the island in the EXPLOIT state; otherwise, the island enters the EXPLORE state. Each state determines the diagnostic evidence included in the next prompt:

$$\begin{aligned} \mathcal{R}_{\text{EXPLOIT}} &= \mathcal{R}_{\text{short}} \cup \mathcal{R}_{\text{long}}, \\ \mathcal{R}_{\text{EXPLORE}} &= \mathcal{R}_{\text{long}} \cup \mathcal{R}_{\text{cross}}, \\ \mathcal{R}_{\text{RESET}} &= \mathcal{R}_{\text{cross}} \cup \mathcal{R}_{\text{negative}}. \end{aligned}$$

Here, $\mathcal{R}_{\text{negative}} \subseteq \mathcal{R}_{\text{long}}$ denotes the subset of long-term patterns identified as *deprecated* (i.e., persistently non-improving) for the given island. It allows the island to reset its current search trajectory while retaining knowledge of previously ineffective directions. We next describe the short-term, long-term, and cross-island reflection mechanisms. A summary is provided in Appendix B (Table 5).

3.3 SHORT-TERM CASE-LEVEL REFLECTION

Short-term reflection provides case-level diagnostic attribution. It first identifies the K cases on which the selected parent performs worst. For each such case, the method then retrieves a peer program from the same island that achieves substantially better performance. The parent and peer implementations are presented together as paired evidence, prompting the LLM to analyze their differences and propose a targeted code modification rather than engage in open-ended brainstorming. Restricting comparisons to programs within the same island helps preserve the island's underlying algorithmic structure while isolating edits associated with improved case-level performance.

3.4 LONG-TERM PATTERN-LEVEL REFLECTION

Every island maintains edit records

$$e = (c^{\text{parent}}, c^{\text{child}}, \text{diff}(c^{\text{parent}}, c^{\text{child}}), \Delta),$$

where Δ denotes the corresponding score change. Every N generations, the LLM performs open-ended clustering of the accumulated code differences and their associated improvements. It induces a set of emergent edit-pattern categories and assigns each observed edit to one of them. For each pattern p, the island-local knowledge base stores its success rate, mean improvement, and effect size:

$$q_p = \frac{s_p}{n_p}, \qquad \bar{\Delta}_p = \frac{1}{n_p} \sum_{e \in p} \Delta_e, \qquad d_p = \frac{\bar{\Delta}_p}{\sigma_p}.$$

Here, n_p is the number of edits assigned to pattern p and $s_p = \sum_{e \in p} \mathbf{1}\{\Delta_e > 0\}$ counts the successful ones. A pattern is labeled *confirmed* once it has accumulated enough edits at a high success rate, and *deprecated* once a comparable number of edits leave it persistently non-improving; otherwise it remains unlabeled. Table 1 shows a representative snapshot of a single island-local knowledge base, in which each induced pattern accumulates its success rate q_p and mean improvement $\bar{\Delta}_p$. Because the pattern labels are induced directly from observed edits rather than specified in advance, the resulting taxonomy reflects the search process instead of relying on hand-crafted categories. Appendix B (Figure 4) illustrates the local knowledge-base update pipeline.

Table 1: Illustrative local-KB snapshot after generation 9. Each row is a pattern with its successful/total edit count, success rate, and mean improvement; a successful edit has positive improvement. These values illustrate the data structure and are not part of the benchmark comparison.

Pattern	Succ. / Count	Success Rate	Mean Δ
`local_search_post_proc`	6 / 7	85.7%	$+0.031$
`scoring_formula_adjust`	3 / 5	60.0%	$+0.019$
`random_perturbation`	1 / 4	25.0%	-0.007
`greedy_init_variant`	0 / 3	0.0%	-0.014

3.5 Cross-Island Population-Level Reflection

When an island's local history provides insufficient guidance, SAMR draws on two forms of cross-island evidence. The first is *contrastive sampling*, which identifies cases on which the current island performs poorly but another island performs well. These case-level contrasts serve as instance-specific counterexamples that reveal weaknesses in the current island's approach. The second is *pattern transfer*, which retrieves edit categories that have been validated by another island but have neither been attempted nor deprecated by the current island:

$$\mathcal{P}_{j\to i} = \mathcal{P}_j^{\text{confirmed}} \setminus \left(\mathcal{P}_i^{\text{tried}} \cup \mathcal{P}_i^{\text{deprecated}}\right).$$

Rather than copying another island's code directly, the prompt conveys case-level contrasts and pattern-level evidence. This allows the target island to benefit from population-wide discoveries while preserving its own algorithmic island. Specifically, transferred patterns are conveyed by their natural-language category names and associated statistics (success rate, mean improvement); the receiving island's LLM interprets these descriptions contextually rather than requiring a shared structural taxonomy. This design avoids brittle category-alignment assumptions: each island maintains its own local labels, and cross-island transfer operates at the semantic level of edit descriptions.

3.6 Knowledge-Preserving Reset

Periodic island resets are already used in FunSearch (Romera-Paredes et al., 2024). The key distinction in SAMR lies in when resets are triggered and what information is retained afterward. A reset occurs only when the island's trajectory indicates severe relative lag or prolonged stagnation. The island's active population is then reinitialized, while its deprecated edit patterns are preserved as negative memory. Thus, a stagnant island is not simply discarded: retaining its failed edits helps the restarted island avoid repeating unproductive directions, while its case-level performance profile remains available for future cross-island contrastive comparisons.

4 Experiments

4.1 Experiment Setup

We evaluate all methods on CO-Bench, a benchmark of 36 combinatorial optimization tasks spanning packing, cutting, facility location, scheduling, routing, assignment, tree, and graph problems (Sun et al., 2026). Each task provides a problem-specific evaluator and classical solver. The evaluator returns a normalized score in $[0, 1]$, with higher values indicating better performance. Token consumption is measured separately rather than equalized across methods. For each problem type, we report the score of the best individual discovered during the search. Each CO-Bench task provides a development set and a test set, with the test instances kept invisible during search. For each task, we select the solution that performs best on the development set and report its score on the test set.

We compare SAMR against five LLM-based AHD methods spanning different search paradigms: (a) island-based evolution: **FunSearch** (Romera-Paredes et al., 2024) and **OpenEvolve** (Sharma,

2025); (b) reflective evolution: **ReEvo** (Ye et al., 2024); (c) tree-structured search: **MCTS-AHD** (Zheng et al., 2025); (d) agentic evolution: **CORAL** (Qu et al., 2026).

All five methods are evaluated with Claude Sonnet 4.5 API. Each method is given the same number of heuristic evaluations, and parameters follow the CO-Bench defaults. We adopt the number of heuristic evaluations as the common budget because it directly bounds how much feedback each method can extract from the benchmark and is the one resource shared by evolutionary, tree-structured, and agentic methods alike. Token consumption is therefore a *consequence* of each method's internal design rather than an externally imposed constraint. Per-method token costs are reported in Table 2 for transparency. We additionally evaluate (a)–(c) and SAMR with DeepSeek V3 to assess cross-LLM robustness; CORAL is excluded because its implementation relies on Claude Code's tool-use interface. The reported best-method count includes ties; consequently, the counts need not sum to 36. Detailed hyperparameter settings—including the momentum window W, the pattern-induction interval N, the worst-case count K, and the stagnation and reset thresholds—together with per-task scores are provided in the supplementary material.

4.2 Main Results

Table 2 summarizes performance across all 36 tasks. SAMR improves the mean normalized score by 0.035 over FunSearch, 0.074 over ReEvo, 0.106 over MCTS-AHD, and 0.042 over OpenEvolve. CORAL achieves the highest mean, exceeding SAMR by 0.010, but consumes about 26.6× more tokens per problem (24.29M vs. 0.91M). Figure 2 illustrates the resulting quality–cost tradeoff.

A two-sided Wilcoxon signed-rank test over the 36 per-task scores confirms these gaps: SAMR is significantly better than FunSearch ($p \approx 2.6 \times 10^{-3}$), ReEvo ($p \approx 1.9 \times 10^{-6}$), MCTS-AHD ($p \approx 1.2 \times 10^{-6}$), and OpenEvolve ($p \approx 1.8 \times 10^{-4}$), whereas the gap to CORAL is not significant ($p \approx 0.26$). Thus SAMR matches the strongest baseline in aggregate quality while using $\sim$27× fewer tokens.

Table 2: CO-Bench results on 36 tasks. Mean and median are normalized scores; Best# counts tasks where a method ties or beats all others within its backbone block; Tokens/prob is mean tokens per problem in millions. The last column reports the two-sided Wilcoxon signed-rank p-value of each method against SAMR over the per-task scores (smaller = more significant). Within each LLM block the best value per column is in bold; SAMR (ours) is shaded. CORAL is not included in the DeepSeek V3 block because its implementation relies on Claude Code's tool-use interface.

Methods	Tasks	Mean score↑	Median score↑	Best#↑	Tokens/prob (M)↓	p vs. Ours
Classical Solver	36	0.798	0.903	–	–	–
LLM-based AHD: *Claude Sonnet 4.5*						
FunSearch	36	0.818	0.882	10	**0.51**	2.6×10^{-3}
ReEvo	36	0.779	0.841	4	2.72	1.9×10^{-6}
MCTS-AHD	36	0.747	0.859	3	0.73	1.2×10^{-6}
OpenEvolve	36	0.811	0.883	5	0.85	1.8×10^{-4}
CORAL	36	**0.863**	**0.931**	**24**	24.29	0.26
SAMR(Ours)	36	0.853	0.925	17	0.91	–
LLM-based AHD: *DeepSeek V3*						
FunSearch	36	0.765	0.864	12	0.48	1.9×10^{-2}
ReEvo	36	0.753	0.850	7	2.92	6.7×10^{-3}
MCTS-AHD	36	0.681	0.794	4	0.67	7.3×10^{-6}
OpenEvolve	36	0.749	0.839	9	0.85	1.6×10^{-3}
SAMR(Ours)	36	**0.804**	**0.879**	**25**	0.89	–

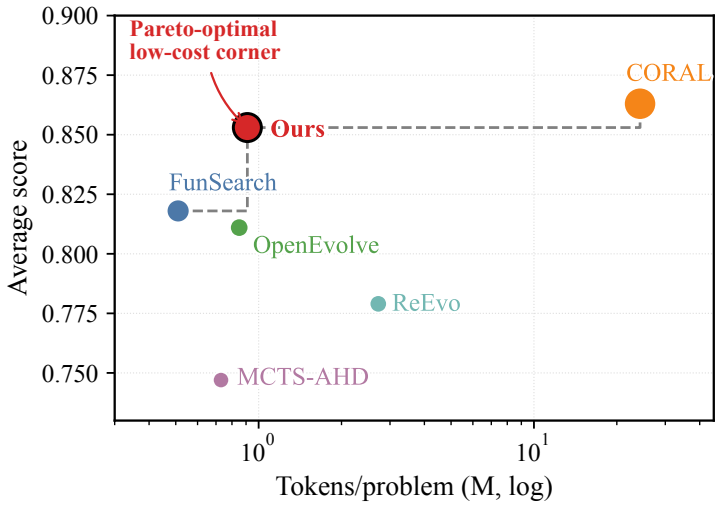

Figure 2: Quality–token Pareto plot for Claude Sonnet 4.5. Bubble area represents best count. The nondominated frontier is FunSearch–SAMR–CORAL.

4.3 Comparison with Classical Solver

Beyond LLM-based methods, CO-Bench provides a classical solver baseline for each task (e.g., LKH for TSP, CPLEX for scheduling), included as the first row of Table 2. Using Claude, SAMR outperforms the reference algorithm on 21 tasks, underperforms on 12, and ties on 3. The largest margins appear on problems where general-purpose solvers scale poorly.

4.4 Cross-Backbone Generalization

Among the methods evaluated with DeepSeek V3, SAMR achieves the highest mean score, median score, and best-task count, as shown in the lower block of Table 2. Every method scores lower than under Claude, but SAMR retains the top rank, indicating that the improvements are not tied to a single backbone. Figure 3 visualizes the per-task score distributions under both backbones: SAMR retains both the highest median and the highest best-task count under Claude Sonnet 4.5 and DeepSeek V3. While every method degrades when the backbone is switched, SAMR preserves its top rank rather than collapsing to the level of the weaker baselines, indicating that its advantage is a distributional property rather than an artifact of a few tasks.

4.5 Ablation Study

Table 3 presents the ablation study result. Removing long-term reflection reduces the mean score from 0.853 to 0.789, a decrease of 7.5 percentage points, while removing cross-island reflection lowers it to 0.784, a decrease of 8.1 percentage points. These results characterize the average contribution of each component and do not imply that every task benefits from every source of evidence. Crucially, injecting all three evidence scales at every step without state-aware gating (*all evidence*) yields the lowest mean score of all configurations (0.779), below even the single-component removals. This confirms that the improvement stems from routing the right evidence to each island state rather than from simply supplying more reflection.

Task-level results show that the benefits of reflection vary across problems. Long-term and cross-island reflection are complementary, but removing either component improves performance on some tasks. Appendix C reports representative comparisons, observed counterexamples, and tentative interpretations.

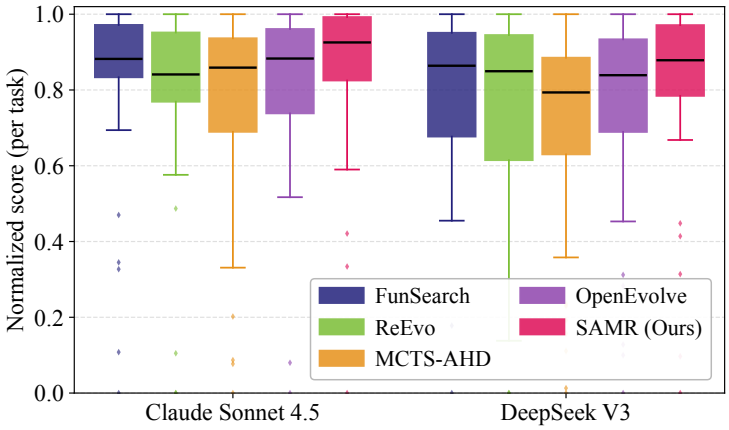

Figure 3: Cross-backbone generalization on CO-Bench (36 tasks). Per-task normalized score distributions grouped by backbone; methods are distinguished by color. All methods degrade from Claude Sonnet 4.5 to DeepSeek V3, yet SAMR retains the highest median and the tightest upper quartile under both backbones.

Table 3: Aggregate component ablations on 36 tasks with Claude. The last row injects all three evidence scales at every step, with no state-aware gating. Best count includes ties.

Configuration	Mean	Median	Best#
SAMR full	**0.853**	**0.925**	**31**
w/o long-term reflection	0.789	0.867	10
w/o cross-island reflection	0.784	0.904	10
w/o state routing (all evidence)	0.779	0.870	5

5 Conclusion and Limitations

We introduced SAMR, a state-aware framework that routes short-term, long-term, and cross-island evidence during LLM-guided program search. Its controller adapts evidence to each island's trajectory, while knowledge-preserving resets retain information about ineffective edits. Across 36 CO-Bench tasks and two LLM backbones, SAMR outperforms the evaluated implementations of FunSearch, ReEvo, MCTS-AHD, and OpenEvolve, and offers a favorable quality–token tradeoff relative to CORAL. Ablations support the contribution of state-dependent evidence selection, although individual tasks do not uniformly benefit from additional reflection. The framework requires case-level evaluator feedback, and its open-ended pattern induction may produce inconsistent or overly broad categories. Future work should investigate surrogate attribution for aggregate-only evaluators and more stable pattern induction. Our experiments also do not directly compare with EoH, HSEvo, or AdaEvolve. Such comparisons would help distinguish the benefits of evidence routing from those of adaptive search control.

AI use statement

We used generative AI tools solely to assist with manuscript writing. We checked the AI-assisted text against the original manuscript to verify that it preserved the intended meaning and accurately represented our work. We did not use generative AI as an authoring aid for research ideation, method design, implementation, experimental execution, literature review, or analysis of results. This disclosure is distinct from the use of LLMs as components of the heuristic-generation methods evaluated in this paper, which use Claude Sonnet 4.5 and DeepSeek V3 as described in the experimental setup. We take responsibility for the final text, claims, and other content of this work.

Ethics statement

This study uses only publicly available CO-Bench data and does not involve human participants, user data, personal information, sensitive data, or animal experiments. We reviewed the applicable licenses and terms of use for the data and code used in this work. We have identified no conflicts of interest or additional ethical concerns specific to this study.

Reproducibility statement

We plan to provide an anonymized implementation and supporting materials for reproducing the experiments. These materials will document the search hyperparameters, model versions and generation settings, evaluation protocol, per-task and ablation results, randomness settings and available seed information, and execution environment. We also plan to provide instructions for running the experiments and computing the reported scores and token costs.

References

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.

Mert Cemri, Shubham Agrawal, Akshat Gupta, Shu Liu, Audrey Cheng, Qiuyang Mang, Ashwin Naren, Lutfi Eren Erdogan, Koushik Sen, Matei Zaharia, Alex Dimakis, and Ion Stoica. Adaevolve: Adaptive llm driven zeroth-order optimization, 2026. URL https://arxiv.org/abs/2602.20133.

Chentong Chen, Mengyuan Zhong, Ye Fan, Jialong Shi, and Jianyong Sun. HiFo-prompt: Prompting with hindsight and foresight for llm-based automatic heuristic design. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *International Conference on Machine Learning (ICML)*, 2024.

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.

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.

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

Fubo Qi, Tianyu Wang, Ruixiang Zheng, and Mian Li. A memetic and reflective evolution framework for automatic heuristic design using large language models. *Applied Sciences*, 15(15):8735, 2025.

Ao Qu, Han Zheng, Zijian Zhou, Yihao Yan, Yihong Tang, Shao Yong Ong, Fenglu Hong, Kaichen Zhou, Chonghe Jiang, Minwei Kong, Jiacheng Zhu, Xuan Jiang, Sirui Li, Cathy Wu, Bryan Kian Hsiang Low, Jinhua Zhao, and Paul Pu Liang. Coral: Towards autonomous multi-agent evolution for open-ended discovery, 2026. URL https://arxiv.org/abs/2604.01658.

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.

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent. GitHub software repository, `https://github.com/algorithmicsuperintelligence/openevolve`, 2025. Accessed: 2026-07-12.

Weiwei Sun, Shengyu Feng, Shanda Li, and Yiming Yang. Co-bench: Benchmarking language model agents in algorithm search for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 33126–33134, 2026.

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

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.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 78338–78373, 2025.

A METHODOLOGICAL POSITIONING

Table 4: Positioning by the object being adapted and the evidence retained. Although prior work is adaptive in several respects, SAMR specifically focuses on routing diagnostic evidence to individual islands.

Method	Primary search control	Persistent evidence	Relation to SAMR
FunSearch/ OpenEvolve	Island evolution and reseeding	Scored program database	Island-specific code, but fixed evidence and no memory of failed edits
ReEvo	Reflective evolution	Accumulated verbal reflections	Multiple time scales, but no island-local edit-pattern statistics
MCTS-AHD	UCT search selection	Tree values and paths	State controls search and action templates
CORAL	Multi-agent exploration	Shared notes and skills	Greater autonomy and a large token budget
SAMR	Per-island trajectory controller	Local and cross-island evidence	Routes multi-level diagnostics by island state

B Additional Details of Multi-Scale Reflection

B.1 Summary of reflection scales

Table 5: The three reflection scales. State selection determines which evidence is exposed rather than accumulating every signal in every prompt.

Reflection	Time scale	Diagnostic level	Used in	Diagnostic question
Short-term	Current prompt	Case	EXPLOIT	Which cases are weak, and what did a stronger peer do differently?
Long-term	Island history	Pattern	EXPLOIT, EXPLORE	Which edit types have helped or hurt this island?
Cross-island	Population history	Population	EXPLORE, RESET	Which directions worked elsewhere but remain untried here?

B.2 Local knowledge-base update pipeline

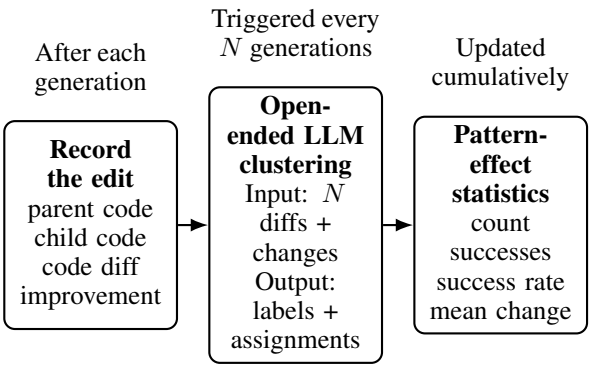

Figure 4: Local-knowledge-base pipeline.

C Task-Level Ablations and Interpretation

Table 6 presents five representative task-level comparisons highlighted in the experimental analysis. Periodic VRP and generalized assignment suffer substantial performance losses when long-term reflection is removed, whereas container loading with weight restrictions and crew scheduling are particularly sensitive to the removal of cross-island evidence. TSP benefits substantially from both components and therefore does not isolate the contribution of either mechanism.

Table 6: Selected problem-level ablations highlighting tasks where individual reflection components provide the largest gains. Bold marks the best configuration in each row.

Problem	Full	w/o long-term	w/o cross-island
TSP	**0.923**	0.595	0.579
Periodic VRP	**0.638**	0.188	0.338
Generalized assignment	**0.997**	0.843	0.942
Container loading with weight restrictions	**0.334**	0.320	0.073
Crew scheduling	**0.590**	0.453	0.339

Additional reflection is not universally beneficial. Table 7 reports all observed reversals in which an ablated variant slightly outperforms the full method. Based on the displayed values, these gains range from 0.005 to 0.050 and occur on tasks for which the full method already achieves high scores. These exceptions qualify the aggregate findings and motivate more fine-grained, task-dependent evidence gating.

Table 7: Reported counterexamples in which removing a reflection component outperforms the full method.

Problem	Full	w/o long-term	w/o cross-island
Aircraft landing	0.908	0.847	**0.919**
Constrained non-guillotine cutting	0.951	**0.956**	0.950
Flow-shop scheduling	0.875	0.913	**0.925**
Unconstrained guillotine cutting	0.969	**0.981**	0.972
Packing unequal circles	0.992	0.992	**1.000**

The task-level results suggest two post-hoc hypotheses. Long-term reflection may be especially useful when improvement arises through repeated refinement of an established heuristic structure. Cross-island reflection may be more valuable when a island requires a structural change rather than incremental local tuning. These interpretations are based on a limited set of selected tasks and have not been independently validated. We therefore present them as hypotheses for future study rather than as established task categories.