COSTADA: COST-CALIBRATED CREDIT FOR BUDGET-AWARE LLM DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models increasingly support scientific and algorithmic discovery through inference-time search over evaluated candidates. Existing adaptive discovery controllers assign credit based only on score progress, even though prompt length, retries, and guidance calls cause search actions to incur different token costs. Under a fixed search budget, the controller must decide which frontier to develop and whether its gain justifies the realized cost before the budget is exhausted. We prove that, in the worst case, cost-blind control can forfeit all but a vanishing fraction of attainable quality as frontier count and cost heterogeneity grow. To address this limitation, we introduce **CostAda**, an adaptive controller built on *cost-calibrated frontier utility*. By valuing progress relative to realized action cost and conditioning credit on the remaining budget, CostAda coordinates local exploration intensity, frontier allocation, and budgeted tactic intervention. Realized action cost and remaining budget therefore shape the search rather than serving only as accounting variables or a stopping rule. Across eight benchmarks with GLM-5 and GPT-5.4, CostAda reaches the strongest baseline's full-budget quality with at most half the budget on thirteen of sixteen benchmark–backbone pairs and achieves the strongest final quality on all sixteen pairs.

1 INTRODUCTION

Large language models (LLMs) increasingly support scientific and algorithmic discovery through inference-time search. This process iteratively generates, evaluates, and refines candidate solutions. Across mathematical optimization, algorithm design, and executable program discovery, automated evaluators score candidates to guide further refinement (Romera-Paredes et al., 2024; Novikov et al., 2025; Lange et al., 2026). Each search round incurs LLM costs, so solution quality depends on how a fixed search budget is allocated across competing search directions. Poor allocation can exhaust this budget before promising candidates receive enough testing and refinement.

Existing LLM discovery systems adapt their search strategies using evaluator feedback (Jiang et al., 2026; Lange et al., 2026; Cemri et al., 2026; Liu et al., 2026a). However, progress-only credit assigns value to score gains without accounting for realized action cost (Lange et al., 2026; Cemri et al., 2026). These costs vary with prompt length, response length, and retries under token-based API pricing. A short refinement and a long-context step can therefore receive comparable credit despite consuming different amounts of the search budget. When such credit guides later choices, costly actions can keep attracting search budget without delivering greater gains.

Figure 1 illustrates this distinction with a short refinement (action A) and a long-context step (action B). Both improve the score by 0.03, but A costs $1\times$ the reference cost and B costs $6\times$. Progress-only credit assigns equal value to A and B. At the same remaining budget, cost-calibrated credit instead gives A higher credit because it achieves the same gain at lower cost and preserves more resources for subsequent search. The value of B also depends on the remaining budget. Early in search, B may help develop a promising frontier, whereas near budget exhaustion its cost may leave too few resources to exploit that frontier. This example motivates credit that accounts for both realized action cost and the budget left for further improvements. For controllers that ignore cost, Proposition 1 shows that the worst-case approximation ratio can vanish as frontier count and cost heterogeneity grow. Smoothing progress alone cannot resolve this cost blindness.

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.

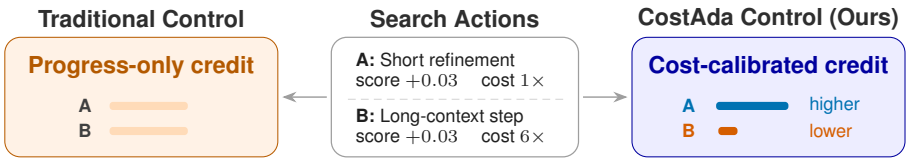


Figure 1: **Cost-calibrated credit for equal-gain, unequal-cost actions.** A short refinement (A) and a long-context step (B) yield the same gain ($+0.03$) at $1\times$ and $6\times$ the reference cost, respectively. Progress-only credit treats A and B equally, whereas CostAda gives A more credit than B.

To address this gap, we introduce CostAda, a cost-calibrated adaptive controller for budget-aware LLM discovery. CostAda replaces progress-only credit with *cost-calibrated frontier utility*, which values local frontier progress and global best-so-far improvement relative to realized action cost. Local progress indicates that a search direction is still developing, whereas global improvement measures its contribution to the strongest solution found so far. For comparable progress at the same remaining budget, a costlier action receives less credit. Early in search, CostAda gives more weight to local development because enough budget remains for a promising frontier to improve on the global best. As the budget shrinks, it shifts credit toward global improvement and penalizes costly steps more strongly, since a given call cost represents a larger fraction of the remaining budget. After each completed action, CostAda charges its realized cost and updates the corresponding frontier's credit to guide subsequent decisions using the updated budget state.

CostAda applies this credit principle through three coupled controls (Section 4). (I) Local exploration intensity favors focused refinement when recent utility is high and broader parent sampling when it is low, while narrowing exploration as the remaining budget shrinks. (II) Frontier allocation selects the next frontier by combining recent cost-calibrated global gain with an exploration bonus that decreases as the budget is spent. Local progress can guide refinement within a frontier, but only global improvement earns allocation reward. (III) Budgeted tactic intervention schedules a guide when global progress stagnates or low-yield spending accumulates, provided the remaining budget covers its estimated cost and a reserve for testing its tactics. These controls make cost and remaining budget part of search decisions, beyond their roles in accounting and stopping.

Contributions. (1) We formulate LLM discovery under a fixed search budget and prove a worst-case separation between cost-blind and cost-aware control (Proposition 1). (2) We introduce cost-calibrated frontier utility with remaining-budget conditioning as the core credit principle. (3) We instantiate this principle in CostAda, an adaptive controller that coordinates local exploration intensity, frontier allocation, and budgeted tactic intervention. (4) Across GLM-5 and GPT-5.4, CostAda reaches the strongest baseline's full-budget quality using at most half the budget on thirteen of sixteen benchmark–backbone pairs. It also achieves the strongest final quality on all sixteen pairs.

2 Related Work

2.1 LLM-Guided Evolutionary Search

LLM-guided evolutionary search uses evaluator feedback to refine candidate solutions (Hemberg et al., 2024). Systems such as FunSearch, AlphaEvolve, and SimpleTES apply this loop to mathematical and algorithmic discovery (Romera-Paredes et al., 2024; Novikov et al., 2025; Ye et al., 2026). Related methods extend it to heuristic design (Liu et al., 2024; Ye et al., 2024; van Stein & Bäck, 2025), scientific equation discovery (Shojaee et al., 2025), and prompt optimization (Yang et al., 2024; Fernando et al., 2024; Agrawal et al., 2026). Recent program-evolution frameworks study efficient sampling, candidate management, and context reuse (Lange et al., 2026; Assumpção et al., 2025; Jiang et al., 2026). TTT-Discover and PACEvolve++ further learn proposal or advisor policies during search (Yuksekgonul et al., 2026; Yan et al., 2026). EvoX co-evolves candidate solutions and their search strategies (Liu et al., 2026a). Progress-aware controllers such as AdaEvolve assign credit from score gains without incorporating realized action cost or the remaining budget (Cemri et al., 2026). In contrast, CostAda uses realized action cost to calibrate frontier credit and the remaining budget to adjust credit and subsequent search decisions.

2.2 BUDGET-AWARE LLM SYSTEMS

LLM reasoning and agent systems face a related budget allocation problem, as each reasoning step or tool call consumes resources needed for subsequent decisions. Classical bandits with knapsacks formalize such allocation as cumulative reward maximization under resource constraints (Badanidiyuru et al., 2013). Token-budget-aware reasoning methods control reasoning length or adapt token allocation to query difficulty (Han et al., 2025; Li et al., 2025; Wen et al., 2025). Agent methods expose remaining resources to guide tool use or jointly select models and budgets for multi-stage tasks (Liu et al., 2025; Wang et al., 2025). BG-MCTS conditions tree selection and expansion on the remaining output-token budget (Miyamoto et al., 2026). These reasoning and agent methods focus on completing tasks within budget, rather than continually improving candidate programs for scientific discovery. Recent work addresses budget allocation in evolutionary discovery (Xing et al., 2026; Yang et al., 2026b), but does not calibrate progress feedback by realized action cost. However, CostAda introduces cost-calibrated frontier utility, distinguishing equal gains by the cost of obtaining them and adjusting their credit with the remaining budget to guide subsequent search.

3 PROBLEM FORMULATION

We consider LLM discovery over a discrete space $\mathcal{P}$ of executable candidates. A deterministic evaluator $F : \mathcal{P} \to \mathbb{R}$ assigns each candidate a scalar score, where higher is better. For tasks with multiple metrics or a minimization objective, F is a direction-adjusted scalar proxy. At each iteration t, the LLM generates a candidate $p_t \in \mathcal{P}$, and the evaluator returns $f_t = F(p_t)$. Search reuses evaluated candidates to improve the best-so-far score $y_t = \max_{\tau \le t} f_\tau$.

We formulate this iterative refinement as a search over K_t frontiers, or local candidate populations, at iteration t. For frontier $k \in \{1, \dots, K_t\}$, the retained candidates form a local archive $D_t^{(k)} \subseteq \mathcal{P}$ with local best score $b_t^{(k)} = \max_{p \in D_t^{(k)}} F(p)$. Across all frontiers, y_t is the global best score found so far. A context builder combines a parent sampled from the selected frontier k_t with archived context candidates to form the next prompt. After evaluation, the selected archive is updated, and a search controller uses the score to select subsequent search actions.

3.1 SEARCH BUDGET AND OBJECTIVE

To account for differences in generation and guidance costs, we adopt a search budget $B > 0$ for search-side LLM expenditure (Badanidiyuru et al., 2013; Yang et al., 2026a). The realized cost c_t at iteration t covers candidate generation, retries, and guidance calls, excluding deterministic evaluator execution. The controller accumulates these costs as $C_t = \sum_{s \le t} c_s$ and represents the unspent budget by the remaining-budget ratio

$$\rho_t = \max(0, 1 - C_t/B) \in [0, 1].$$

For credit calibration, the controller uses the normalized step cost $\tilde{c}_t = c_t/(\bar{c} + \epsilon_c)$, where $\bar{c}$ is a fixed reference generation cost shared across benchmarks and $\epsilon_c > 0$ is a stabilizer. The budget constraint applies to actual cumulative expenditure C_t, with accounting details in Appendix E.

The budgeted discovery objective is to maximize expected best-so-far quality before the search-side budget is exhausted:

$$\max_{\pi} \ \mathbb{E}[y_{\tau_B}], \qquad \tau_B = \max\{t : C_t \le B\},$$

where π is the search policy and τ_B is the last iteration whose cumulative cost stays within budget B. The final output is the highest-scoring candidate found by τ_B.

3.2 FAILURE OF COST-BLIND CREDIT

Call a controller *cost-blind* if each action distribution depends only on action and score history, never on realized cost. A controller whose decisions rely only on progress credit and use realized cost nowhere else belongs to this class. Smoothing or combining progress signals preserves this cost-blindness. The following proposition isolates its effect when actions have different costs.

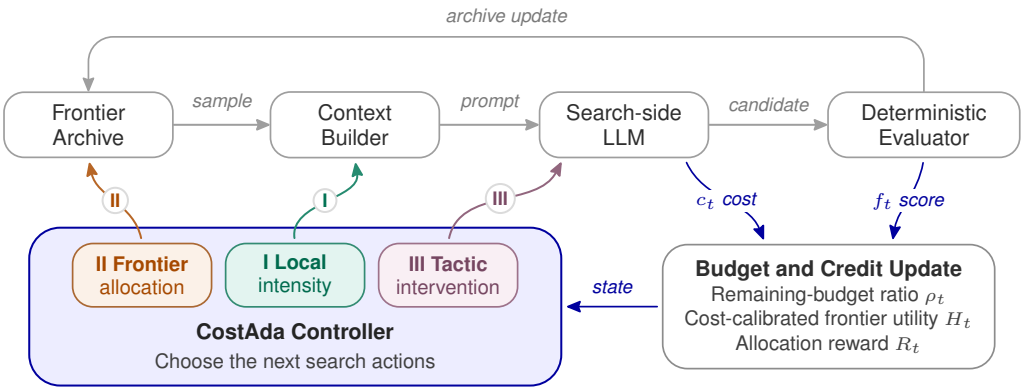


Figure 2: **The overall architecture of CostAda.** Gains over stored best scores are calibrated by realized cost and remaining budget. The lower-right block maintains the remaining-budget ratio ρ_t and smoothed credit signals H_t (frontier utility) and R_t (allocation reward). This state then guides local exploration intensity (I), frontier allocation (II), and budgeted tactic intervention (III).

Proposition 1 (Worst-case separation between cost-blind and cost-aware control) *Consider search over $K \geq 2$ frontiers with initial best score zero. Every selection increases the global best score by the same positive amount. One frontier costs $c > 0$ per selection, while the others cost κc, where $\kappa > 1$ is the cost ratio. For any cost-blind controller and budget $B \geq c$, some fixed placement of the cheap frontier limits the controller's expected budgeted value to at most $\left(\frac{1}{K} + \frac{1}{\kappa}\right) \frac{B/c}{\lfloor B/c \rfloor}$ OPT. Here, OPT is the optimal value attainable within budget B on that instance. A cost-aware policy that samples each frontier once and then selects by gain per realized cost attains $(1 - o(1))$ OPT on every placement as $B/c \to \infty$, with K, κ fixed.*

The proof is in Appendix B. For fixed K and κ, the finite-budget rounding factor tends to one as B/c grows. The limiting upper bound $1/K + 1/\kappa$ tends to zero as the frontier count K and the cost ratio κ increase. This separation motivates the cost-calibrated utility in Section 4, which restores cost-dependent credit and conditions subsequent decisions on the remaining budget.

4 Method

4.1 Overview

As shown in Figure 2, the CostAda controller combines cost-calibrated frontier utility with remaining-budget conditioning. Each frontier maintains a group of candidate programs in a local archive, and all frontiers share the search budget. The upper loop constructs a prompt from a selected candidate and archived context, generates and evaluates a new candidate, then updates the archive. The resulting score and realized action cost feed the lower-right block, which measures gains against previous local and global bests and updates the remaining budget and cost-calibrated credit. The controller in the lower-left block uses this updated state to choose the next search actions through three controls. Local exploration intensity responds to frontier utility, frontier allocation uses global gain relative to cost, and budgeted tactic intervention combines stagnation or low-yield spending with affordability. Decisions use the budget and credit available before each action, whose realized cost updates the state only after completion and thus informs later decisions.

4.2 Cost-Calibrated Frontier Utility

To assess a frontier's progress, we distinguish improvements within that frontier from improvements in the best solution found across the search. Because a candidate can improve its frontier's best score without exceeding the global best, local gain captures progress not reflected in global gain. At iteration t, the new candidate from frontier k_t receives score f_t. Relative to the pre-iteration local

best $b_{t-1}^{(k_t)}$ and global best y_{t-1}, its local gain $\delta_t^{(k_t)}$ and global gain g_t are

$$\delta_t^{(k_t)} = \max\left(\frac{f_t - b_{t-1}^{(k_t)}}{\max(|f_t|, |b_{t-1}^{(k_t)}|, 1)}, 0\right), \qquad g_t = \max\left(\frac{f_t - y_{t-1}}{\max(|f_t|, |y_{t-1}|, 1)}, 0\right).$$

Here, both gains quantify positive improvements normalized by the larger magnitude of the candidate and reference scores. A denominator floor of one prevents amplification near zero.

These gains are weighted by the budget remaining after iteration t. Specifically, we set the global weight to $\lambda_t = 1 - \rho_t$, so that credit shifts from local development toward global improvement as the budget shrinks. We further account for realized action cost by dividing this weighted progress by the cost-calibration factor $d_t = 1 + \lambda_t^c \phi_t$. Here, $\phi_t = \log(1 + \tilde{c}_t)$ logarithmically scales the normalized step cost from Section 3, limiting the influence of unusually expensive calls. The corresponding cost weight $\lambda_t^c = \max(\lambda_t, \lambda_{\min})$ allows the penalty to strengthen as the budget is spent, while its fixed positive floor $\lambda_{\min}$ keeps it active from the start. Thus, the cost-calibrated frontier utility is given by

$$u_t^{(k_t)} = \frac{\lambda_t g_t + (1 - \lambda_t)\delta_t^{(k_t)}}{d_t}. \tag{1}$$

A step without improvement has zero utility even when the frontier has recently made progress. We therefore guide subsequent local exploration with an exponential moving average that retains recent credit while gradually discounting older observations:

$$H_t^{(k_t)} = \alpha H_{t-1}^{(k_t)} + (1 - \alpha) u_t^{(k_t)},$$

where $\alpha \in (0, 1)$ is the smoothing coefficient, with larger values placing more weight on past utility and smaller values on the latest utility. Only the selected frontier receives a new utility observation, so $H_t^{(k)} = H_{t-1}^{(k)}$ for $k \neq k_t$. When the observed utility is zero, the selected frontier's history is discounted by a factor of α, while the incurred cost reduces the remaining budget.

4.3 Budget-Aware Control

The controller uses progress and cost feedback together with the remaining budget to guide subsequent search through three controls. Local exploration intensity adjusts the breadth of sampling within a frontier, frontier allocation selects the next frontier, and budgeted tactic intervention determines when to purchase guidance that the remaining budget can cover.

Local exploration intensity. Local control adjusts sampling between strong candidates and more novel candidates within the selected frontier. Initial sampling gathers feedback when a frontier is first selected. For a previously selected frontier, the default sampling rule sets intensity at iteration t using its utility state and the remaining budget from iteration $t - 1$:

$$I_t^{(k_t)} = I_{\min} + (I_{\max} - I_{\min}) \cdot \frac{\rho_{t-1}}{1 + \sqrt{H_{t-1}^{(k_t)} + \epsilon_H}},$$

where $I_{\min}$ and $I_{\max}$ are fixed lower and upper bounds on local sampling intensity and $\epsilon_H > 0$ is a small stabilizer. High recent utility reflects cost-efficient progress and favors lower-intensity sampling around strong candidates. In contrast, lower recent utility raises intensity at a given remaining-budget ratio, broadening parent sampling toward novel candidates. Meanwhile, the multiplier ρ_{t-1} narrows exploration as the remaining budget shrinks.

We then map this intensity to probabilities over exploration, exploitation, and balanced sampling using the fixed sampler π_{local}. The resulting mode $a_t \sim \pi_{\text{local}}(\cdot \mid I_t^{(k_t)})$ sets the default parent distribution and prompt guidance. Across these modes, we request the same number of local and global context programs, with sampling and retrieval rules detailed in Appendix C.2.

Frontier allocation. Frontier allocation chooses which frontier receives the next search step. Its reward credits only improvements in the global best, which advance the overall discovery objective. After iteration t, we apply the same cost calibration to define its reward:

$$r_t^{(k_t)} = \frac{g_t}{d_t}.$$

To compare recent contributions across frontiers, CostAda smooths this reward as

$$R_t^{(k_t)} = \gamma R_{t-1}^{(k_t)} + (1-\gamma) r_t^{(k_t)},$$

where $\gamma \in (0,1)$ is a fixed smoothing coefficient. When guidance yields substantial global improvement, a consolidation window selects the frontier on which the guide was generated, uses exploitation mode, and suppresses new interventions. Outside this window, CostAda selects the next frontier using an upper-confidence allocation rule (Auer et al., 2002):

$$k_{t+1} = \arg\max_k \left[R_t^{(k)} + \rho_t \, c_{\mathrm{ucb}} \sqrt{\frac{\log(N_t+2)}{n_t^{(k)}+1}} \right],$$

where c_{ucb} is a fixed optimism scale, $n_t^{(k)}$ counts completed selections of frontier k, and $N_t = \sum_k n_t^{(k)}$ is the total number of completed frontier selections. The first term exploits frontiers with recent global gain relative to cost. The second gives under-sampled frontiers an exploration bonus that decreases with the remaining budget. As less budget remains to develop under-sampled frontiers, allocation increasingly favors observed global gain relative to cost.

Budgeted tactic intervention. When candidate refinement makes little global progress, a guide call can generate a batch of tactics. Because testing these tactics requires further candidate generation, we use the affordability indicator A_t to check that the remaining budget covers the estimated guide cost and a testing reserve. The stagnation indicator P_t equals one after a sustained absence of meaningful global progress, with patience adapting to the remaining budget and frontier count. In addition, Y_t detects accumulated low-yield spending, allowing intervention before the stagnation requirement is met. After iteration t, we schedule guidance for the next eligible iteration using

$$\mathcal{I}_t = A_t \wedge (P_t \vee Y_t),$$

with $\mathcal{I}_t = 0$ during consolidation, an active tactic batch, or a pause after unsuccessful guidance.

The candidate generations that test a guide's tactic batch form a guide cycle. Within this cycle, the guide call and all candidate generations that test its tactics are charged to the same search budget. Before guidance produces substantial global improvement, CostAda favors breakthrough tactics that propose new solution strategies. After successful guidance, refinement becomes the default mode for later guides, focusing on improvements to existing solutions. If the tactic batch is exhausted without substantial improvement, we pause new guide calls for K_t further iterations while search continues. Further details can be found in Appendix C.4.

4.4 Overall Procedure

In summary, each iteration selects a frontier and sampling mode from the remaining budget and stored credit, with any active consolidation window taking precedence. Any scheduled guide is generated before the candidate, whose prompt combines the selected parent program, archived context, and the active tactic when available. After evaluation, all search-side calls are charged, and gains are computed against the pre-step best scores before the archives and credit are updated. The resulting evidence determines whether to schedule an intervention for the next eligible iteration. The loop stops after reaching the budget threshold or iteration cap and returns the best candidate found. For reproducibility, Appendix C provides the pseudocode (Algorithm 1), supplementary control rules, and controller settings shared across all eight benchmarks.

5 Experiments

5.1 Experimental Setup

Benchmarks. Following prior LLM discovery studies (Liu et al., 2026b; Cemri et al., 2026; Liu et al., 2026a), we evaluate on eight benchmarks spanning geometric packing, extremal geometry, additive combinatorics, and executable program optimization. Together, these benchmarks cover diverse constraints and optimization objectives in iterative program discovery. Further details, including benchmark definitions and score directions, are provided in Appendix D.

Table 1: **Solution quality under equal budgets within each backbone.** Arrows indicate the preferred direction. The strongest displayed values, including ties, are bolded. We adopt the human reference values reported in prior work (Novikov et al., 2025; Cemri et al., 2026; Liu et al., 2026a).

Benchmark	Human	AdaEvolve		EvoX		CostAda	
		Mean ± Std	Best	Mean ± Std	Best	Mean ± Std	Best
Backbone: GLM-5							
Circle Packing ↑	2.6340	2.5617 ± 0.0773	2.6224	2.5717 ± 0.0752	2.6181	**2.6306 ± 0.0054**	**2.6360**
Circle Packing Rect ↑	2.3640	2.2089 ± 0.1635	2.3495	2.1989 ± 0.2563	2.3540	**2.3525 ± 0.0081**	**2.3572**
Heilbronn Convex ↑	0.0306	0.0229 ± 0.0015	0.0247	0.0235 ± 0.0012	0.0245	**0.0263 ± 0.0018**	**0.0280**
Heilbronn Triangle ↑	0.0360	0.0303 ± 0.0011	0.0314	0.0220 ± 0.0050	0.0257	**0.0314 ± 0.0003**	**0.0317**
MinMaxDist (n=16, d=2) ↓	12.89	14.06 ± 1.30	13.29	14.17 ± 1.92	13.06	**13.11 ± 0.29**	**12.89**
MinMaxDist (n=14, d=3) ↓	4.17	4.38 ± 0.08	4.31	4.20 ± 0.03	**4.17**	**4.17 ± 0.00**	**4.17**
Third Autocorrelation ↓	1.4581	1.4685 ± 0.0014	1.4674	1.5481 ± 0.1308	1.4720	**1.4658 ± 0.0037**	**1.4620**
Signal Processing ↑	–	0.6393 ± 0.0240	0.6656	0.6502 ± 0.0430	0.6927	**0.7222 ± 0.0127**	**0.7364**
Backbone: GPT-5.4							
Circle Packing ↑	2.6340	2.6196 ± 0.0030	2.6216	2.5851 ± 0.0510	2.6215	**2.6248 ± 0.0046**	**2.6280**
Circle Packing Rect ↑	2.3640	2.3148 ± 0.0357	2.3501	2.3474 ± 0.0011	2.3487	**2.3577 ± 0.0050**	**2.3621**
Heilbronn Convex ↑	0.0306	0.0197 ± 0.0020	0.0220	0.0223 ± 0.0031	**0.0252**	**0.0244 ± 0.0008**	**0.0252**
Heilbronn Triangle ↑	0.0360	0.0307 ± 0.0008	0.0314	0.0263 ± 0.0035	0.0304	**0.0330 ± 0.0009**	**0.0337**
MinMaxDist (n=16, d=2) ↓	12.89	13.00 ± 0.00	13.00	13.00 ± 0.00	13.00	**12.93 ± 0.06**	**12.89**
MinMaxDist (n=14, d=3) ↓	4.17	4.55 ± 0.15	4.46	4.44 ± 0.02	4.42	**4.42 ± 0.08**	**4.32**
Third Autocorrelation ↓	1.4581	1.5081 ± 0.0069	1.5012	1.4824 ± 0.0042	1.4788	**1.4717 ± 0.0074**	**1.4631**
Signal Processing ↑	–	0.7182 ± 0.0137	0.7334	0.6767 ± 0.0179	0.6968	**0.7672 ± 0.0292**	**0.7990**

Baselines. We compare CostAda with AdaEvolve (Cemri et al., 2026) and EvoX (Liu et al., 2026a), two recent leading methods for adaptive LLM discovery. AdaEvolve is the closest progress-aware controller, adapting local exploration, frontier allocation, and intervention from score improvements. EvoX jointly evolves candidates and their search strategies, providing a complementary approach to adapting discovery. Neither baseline conditions decisions on remaining budget. Within each setting, all methods share the evaluator, search scaffold, backbone, fixed token prices, and budget protocol. This shared setting focuses the comparison on search-control mechanisms.

Backbones and budgets. We use GLM-5 (GLM-5-Team: Aohan Zeng et al., 2026) and GPT-5.4 (OpenAI, 2026) with respective dollar budgets B=1 and B=5. Appendix E provides accounting details and fixed token prices. Each run stops after the first completed iteration with cumulative cost at or above B, or after 100 iterations (budget diagnostics in Appendix K). To track search progress, we record results from the same runs at the budget cutoffs $0.25B$, $0.5B$, $0.75B$, and B, using the first completed iteration reaching or exceeding each threshold.

Metrics. Following the same studies (Liu et al., 2026b; Cemri et al., 2026; Liu et al., 2026a), we measure final quality with the native benchmark objective of the best-so-far candidate, which we record at budget cutoffs rather than iteration counts (BESTSCORE@BUDGET). Table 1 lists the native units and score directions. Because these objectives are not comparable across benchmarks, we also record a direction-adjusted normalized score. To complement final quality, we compute normalized AUC to summarize the best-so-far normalized score over the budget path and capture how early strong solutions are found (Appendix I). Across three independent runs per setting, as in AdaEvolve and EvoX (Cemri et al., 2026; Liu et al., 2026a), we report the mean and sample standard deviation for both metrics, together with the strongest final objective (Best).

5.2 SOLUTION QUALITY UNDER EQUAL BUDGETS

As shown in Table 1, we make the following observations about final quality under equal budgets. First, CostAda's advantage over both adaptive baselines is consistent across benchmarks and backbones. It achieves the strongest mean final objective across all eight benchmarks with GLM-5 and GPT-5.4, and its Best is also strongest or tied in every setting. Moreover, CostAda's mean matches or exceeds each baseline's Best on six of the eight benchmarks under GLM-5 and seven under GPT-5.4, reflecting consistently strong runs. Second, CostAda improves program quality even when the strongest baseline changes with the backbone. On Signal Processing, it raises the mean final objective by 11.1% over EvoX under GLM-5 and by 6.8% over AdaEvolve under GPT-5.4. The ablations in Appendix H attribute much of this gain to cost calibration and remaining-budget condition-

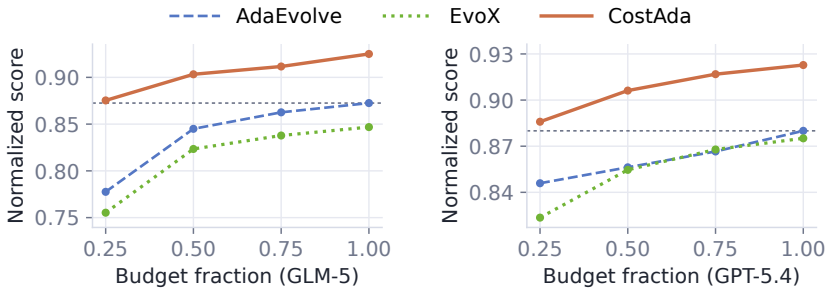


Figure 3: **Mean normalized score at different budget fractions.** At each cutoff, scores are averaged over eight benchmarks under GLM-5 (left) and GPT-5.4 (right). The panels use independent vertical scales. Dotted lines mark the strongest baseline's full-budget score in each panel. Per-benchmark normalized scores and AUC values appear in Appendix I.

ing. Removing remaining-budget conditioning under GLM-5, or either component under GPT-5.4, lowers the final Signal Processing objective below that of the strongest baseline.

Finally, CostAda matches the reported human references on both MinMaxDist tasks and exceeds the reference on Circle Packing. Under GLM-5, CostAda's runs on MinMaxDist (n=14, d=3) all reach the reference value of 4.17, and its Best matches the reference on MinMaxDist (n=16, d=2) under both backbones. On the latter task under GPT-5.4, AdaEvolve and EvoX remain at 13.00 in every run, whereas CostAda lowers the mean to 12.93. On Circle Packing under GLM-5, CostAda's Best exceeds the reference (2.6360 vs. 2.6340), and its standard deviation is about one fourteenth of those of AdaEvolve and EvoX. Because each baseline's Best comes within 0.013 of CostAda's mean, its mean advantage on this task stems mainly from consistently avoiding weak runs.

5.3 Budget Efficiency

Figure 3 reveals that CostAda finds strong solutions early and stays ahead as spending increases. First, CostAda needs substantially less budget to reach the strongest baseline's final quality. At a quarter of the budget, its mean normalized score under both backbones already exceeds the strongest baseline's full-budget score, marked by the dotted line. Per benchmark, it reaches the strongest baseline's full-budget quality with at most half the budget on thirteen of sixteen benchmark–backbone pairs (Appendix F). Cost calibration and remaining-budget conditioning both contribute to this early advantage, as the ablations in Section 5.4 show. Second, the lead persists along the entire budget path. CostAda's curve lies at least 0.04 above both baselines at each of the four cutoffs, and it achieves the highest normalized AUC on fifteen of sixteen pairs (Appendix I).

Finally, the early advantage also holds when a smaller budget is specified before search. We configure CostAda directly with B=0.5 under GLM-5 and compare methods at the same absolute costs (Appendix G). These runs outperform the strongest baseline result in all 16 benchmark–cutoff cells. They also match or improve on CostAda's corresponding cutoffs from runs configured with B=1 in 14 cells. Here, the smaller budget conditions credit and control from the start of search.

5.4 Ablation Study

We compare CostAda with three variants in Table 2, each keeping all three controls and removing one budget-aware component. Specifically, w/o cost calibration sets $d_t = 1$, leaving progress-only credit, and w/o budget conditioning fixes $\lambda_t = \lambda_t^c = 0.5$ while dropping ρ_t from local intensity and the exploration bonus. The variant w/o intervention gating schedules the same guidance without the A_t and Y_t gates. Each component raises normalized AUC in all six settings, with the largest effect varying by backbone and task. First, cost calibration matters most on Heilbronn Triangle and Signal Processing under GPT-5.4, where its removal produces the largest losses. Second, remaining-budget conditioning matters most on the same two tasks under GLM-5. On Heilbronn Triangle, removing it still yields higher final and half-budget objectives than removing cost calibration (Appendix H). The lower AUC of this variant (0.4073 vs. 0.4654) therefore reflects weaker quality earlier in the search,

Table 2: **Ablation study of CostAda.** Entries report normalized AUC as **Mean** $\pm$ **Std**. Higher is better, and the best means within each benchmark and backbone are bolded.

Backbone	Variant	Circle Packing	Heilbronn Triangle	Signal Processing
GLM-5	CostAda	**0.9517 $\pm$ 0.0095**	**0.7831 $\pm$ 0.0486**	**0.6546 $\pm$ 0.0444**
	w/o cost calibration	0.9053 $\pm$ 0.0376	0.4654 $\pm$ 0.0614	0.6355 $\pm$ 0.0200
	w/o budget conditioning	0.9161 $\pm$ 0.0316	0.4073 $\pm$ 0.0426	0.6146 $\pm$ 0.0134
	w/o intervention gating	0.8478 $\pm$ 0.0642	0.4494 $\pm$ 0.1622	0.6293 $\pm$ 0.0140
GPT-5.4	CostAda	**0.9825 $\pm$ 0.0050**	**0.7951 $\pm$ 0.0253**	**0.7022 $\pm$ 0.0224**
	w/o cost calibration	0.9648 $\pm$ 0.0276	0.6748 $\pm$ 0.0459	0.6333 $\pm$ 0.0028
	w/o budget conditioning	0.9673 $\pm$ 0.0079	0.7494 $\pm$ 0.0615	0.6557 $\pm$ 0.0192
	w/o intervention gating	0.9402 $\pm$ 0.0246	0.6949 $\pm$ 0.0259	0.6717 $\pm$ 0.0116

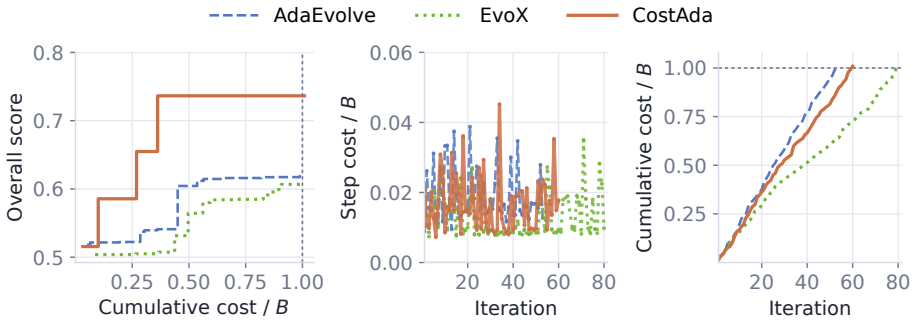


Figure 4: **Solution quality and search cost for Signal Processing under GLM-5.** Panels show the best-so-far benchmark objective versus cumulative realized cost (left), realized step cost versus iteration (middle), and cumulative realized cost versus iteration (right). Costs are normalized by the shared search budget B, with a dotted line at 1.

which final quality alone would miss. Finally, intervention gating matters most on Circle Packing, where its removal lowers AUC by 0.1039 under GLM-5 and 0.0423 under GPT-5.4.

5.5 Case Study

Figure 4 follows each method's search on Signal Processing under GLM-5 to relate solution quality to realized cost. In the left panel, CostAda surpasses both baselines' final objectives within roughly a third of the budget and keeps that lead. Across runs, CostAda's half-budget mean in Appendix F (0.6924) likewise exceeds the strongest baseline's full-budget mean (0.6502). The middle panel shows four- to sixfold variation in step cost within each run, which cost calibration accounts for when assigning credit. In the right panel, the same budget lasts 53 iterations for AdaEvolve, 60 for CostAda, and 80 for EvoX, which illustrates why we compare methods at budget cutoffs rather than at iteration counts. The largest rise of CostAda's objective, at about $0.36B$ in the left panel, comes from the tactics of a refinement guide. Low-yield spending triggered this guide before the stagnation threshold was reached, so realized cost moved the intervention earlier (Appendix C.5). Appendix J extends these trajectories to all eight benchmarks under both backbones.

6 Conclusion

We presented CostAda, a cost-calibrated controller for budget-aware LLM discovery. Our analysis shows that the worst-case approximation ratio of cost-blind control can vanish as frontier count and cost heterogeneity grow. This limitation motivates CostAda's cost-calibrated frontier utility and remaining-budget conditioning for local exploration intensity, frontier allocation, and budgeted tactic intervention. Across eight benchmarks with GLM-5 and GPT-5.4, CostAda reaches the strongest baseline's full-budget quality with at most half the budget on thirteen of sixteen benchmark–backbone pairs. It also achieves the strongest final quality on all sixteen pairs.

AI USE STATEMENT

Generative AI tools assisted with method implementation and refinements to mathematical statements and proofs. They also provided feedback on experimental comparisons and supported result interpretation, literature retrieval, and language polishing.

ETHICS STATEMENT

The authors affirm that this work adheres to the ICLR Code of Ethics. The study involves no human subjects or sensitive or private data, and all benchmarks are publicly available.

REPRODUCIBILITY STATEMENT

The main text and appendix provide the proof, implementation details, controller settings, experimental setup, and evaluation procedures. We provide the source code and experimental scripts at `https://anonymous.4open.science/r/CostAda`.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, 2026.

Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. CodeEvolve: An open source evolutionary coding agent for algorithm discovery and optimization. *arXiv preprint arXiv:2510.14150*, 2025.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, 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. *arXiv preprint arXiv:2602.20133*, 2026.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 13481–13544, 2024.

Erich Friedman. Erich's Packing Center. Online resource, 2025. URL `https://erich-friedman.github.io/packing/`.

GLM-5-Team: Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui,

Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. GLM-5: from vibe coding to agentic engineering. arXiv preprint arXiv:2602.15763, 2026.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Erik Hemberg, Stephen Moskal, and Una-May O'Reilly. Evolving code with a large language model. *Genetic Programming and Evolvable Machines*, 25(2):21, 2024.

Jiachen Jiang, Tianyu Ding, and Zhihui Zhu. DeltaEvolve: Accelerating scientific discovery through momentum-driven evolution. *arXiv preprint arXiv:2602.02919*, 2026.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations*, 2026.

Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. SelfBudgeter: Adaptive token allocation for efficient LLM reasoning. *arXiv preprint arXiv:2505.11274*, 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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 32201–32223, 2024.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, Qiuyang Mang, Ashwin Naren, Ethan Boneh, Audrey Cheng, Melissa Z. Pan, Alexander Du, Kurt Keutzer, Alexandros G. Dimakis, Koushik Sen, Matei Zaharia, and Ion Stoica. EvoX: Meta-evolution for automated discovery. *arXiv preprint arXiv:2602.23413*, 2026a.

Shu Liu, Mert Cemri, Shubham Agarwal, Alexander Krentsel, Ashwin Naren, Qiuyang Mang, Zhifei Li, Akshat Gupta, Monishwaran Maheswaran, Audrey Cheng, Melissa Z. Pan, Ethan Boneh, Kannan Ramchandran, Koushik Sen, Matei Zaharia, Alexandros G. Dimakis, and Ion Stoica. SkyDiscover: A flexible, adaptive framework for AI-driven scientific and algorithmic discovery. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 1223–1227, 2026b.

Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-aware tool-use enables effective agent scaling. *arXiv preprint arXiv:2511.17006*, 2025.

Sora Miyamoto, Daisuke Oba, and Naoaki Okazaki. Aligning tree-search policies with fixed token budgets in test-time scaling of LLMs. *arXiv preprint arXiv:2602.09574*, 2026.

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

OpenAI. GPT-5.4 thinking system card. https://openai.com/index/gpt-5-4-thinking-system-card/, 2026.

OpenRouter. Z.ai: GLM-5 API pricing and benchmarks. https://openrouter.ai/z-ai/glm-5, 2026a.

OpenRouter. OpenAI: GPT-5.4 API pricing and benchmarks. https://openrouter.ai/openai/gpt-5.4, 2026b.

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

Klaus F. Roth. On a problem of Heilbronn. *Journal of the London Mathematical Society*, 26:198–204, 1951.

Asankhaya Sharma. OpenEvolve: an open-source evolutionary coding agent. GitHub repository, https://github.com/algorithmicsuperintelligence/openevolve, 2025.

Belle A. Shenoi. *Introduction to Digital Signal Processing and Filter Design*. John Wiley & Sons, 2005.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan Reddy. LLM-SR: Scientific equation discovery via programming with large language models. In *International Conference on Learning Representations*, volume 2025, pp. 16054–16085, 2025.

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

Carlos Vinuesa del Río. *Generalized Sidon sets*. PhD thesis, Universidad Autónoma de Madrid, 2010.

Fali Wang, Hui Liu, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Zongyu Wu, Chen Luo, Zhen Li, Xianfeng Tang, Qi He, and Suhang Wang. AgentTTS: Large language model agent for test-time compute-optimal scaling strategy in complex tasks. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. BudgetThinker: Empowering budget-aware LLM reasoning with control tokens. *arXiv preprint arXiv:2508.17196*, 2025.

Sixue Xing, Haoyu He, Kerui Wu, Zhuo Yang, Haozheng Luo, Tianfan Fu, and Aarthy Nagarajan. Compute allocation for self-evolving LLMs: From depth-breadth to multi-armed bandits. *arXiv preprint arXiv:2605.29268*, 2026.

Minghao Yan, Bo Peng, Benjamin Coleman, Ziqi Chen, Zhouhang Xie, Shuo Chen, Zhankui He, Noveen Sachdeva, Weili Wang, Ed H. Chi, Shivaram Venkataraman, Wang-Cheng Kang, Derek Zhiyuan Cheng, and Beidou Wang. PACEvolve++: Improving test-time learning for evolutionary search agents. *arXiv preprint arXiv:2605.07039*, 2026.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

Liming Yang, Junyu Luo, Xuanzhe Liu, Yiling Lou, and Zhenpeng Chen. BAMAS: Structuring budget-aware multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29802–29810, 2026a.

Yang Yang, Zining Zhong, Jindong Li, Jiemin Wu, Kaishen Yuan, Wenshuo Chen, Menglin Yang, and Yutao Yue. TurboEvolve: Towards fast and robust LLM-driven program evolution. *arXiv preprint arXiv:2604.18607*, 2026b.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Haotian Ye, Haowei Lin, Jingyi Tang, Yizhen Luo, Rahul Thapa, Caiyin Yang, Chang Su, Rui Yang, Ruihua Liu, Rundao Li, Zeyu Li, Pengwei Sun, Chong Gao, Dachao Ding, Guangrong He, Miaolei Zhang, Lina Sun, Wenyang Wang, Yuchen Zhong, Zhuohao Shen, Puheng Li, Pan Lu, Bianxiao Cui, Di He, Jianzhu Ma, Junfeng Li, Hexi Baoyin, Yejin Choi, Stefano Ermon, Xiaowen Chu, Tongyang Li, Yuzhi Xu, and James Zou. Structured scaling of AI discovery across diverse scientific domains. *arXiv preprint arXiv:2604.19341*, 2026.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026.

A Scope, Limitations, and Future Work

CostAda controls search-side LLM spending under deterministic evaluators held fixed across methods. This spending covers candidate generation, retries, and guidance calls, but excludes evaluator execution, human review, and downstream deployment. Our efficiency claims therefore concern search-side LLM spending rather than end-to-end runtime. Reported dollar values depend on the fixed token prices used in our experiments, whereas CostAda acts on normalized costs and remaining-budget ratios. Jointly rescaling the token prices, search budget, reference cost, and guide-cost estimate leaves these quantities unchanged up to fixed numerical stabilizers. Because output length and retries become known only after a call completes, all methods finish slightly above the nominal budget, by 1.1% to 2.5% on average (Appendix K).

Future work. Three extensions follow directly from this scope. First, a multi-resource budget could jointly account for token cost, latency, and evaluator usage. Second, strictly feasible control could combine pre-call cost prediction with adaptive generation limits to avoid crossing the budget boundary. Third, credit rules that account for evaluator uncertainty could extend CostAda to noisy simulators, wet-lab feedback, and human judgment. In such settings, safety-constrained discovery would also require explicit constraints on generation and evaluation.

B Proof of Proposition 1

Instance. Fix $g > 0$ and the parameters of Proposition 1, with initial score $y_0 = 0$. Every frontier selection deterministically improves the global best score by g. The budgeted value is therefore g times the number of selections completed within budget B. Consider the K instances indexed by the position j of the cheap frontier. Frontier j costs c per selection, and all others cost κc.

Optimum. Selecting the cheap frontier at every step completes $\lfloor B/c \rfloor$ selections, so $\mathrm{OPT} = g\lfloor B/c \rfloor$. No policy can complete more equal-gain selections within the same budget.

Upper bound for cost-blind controllers. Fix any possibly randomized cost-blind controller and couple the instances using the same internal randomness. Every selection returns gain g, so its action–score history is independent of the cheap frontier's position until termination. Extend the action sequence past budget termination solely for the counting argument, and denote this common sequence by σ. Condition on σ. For placement j, let N_j count selections completed within budget, and let $V_j(n)$ count visits to frontier j among the first n selections. The first n selections cost $c\,(\kappa n - (\kappa - 1)V_j(n))$, so the budget constraint at $n = N_j$ gives

$$N_j \;\le\; \frac{B}{\kappa c} + \frac{\kappa - 1}{\kappa}\, V_j(N_j).$$

Every selection costs at least c, so $N_{\max} = \max_j N_j \le B/c$. Since each V_j is nondecreasing,

$$V_j(N_j) \le V_j(N_{\max})$$

for every j, and $\sum_j V_j(N_{\max}) = N_{\max}$. Summing the preceding bound over the K placements and dividing by K gives

$$\frac{1}{K}\sum_{j=1}^{K} N_j \;\le\; \frac{B}{\kappa c} + \frac{\kappa - 1}{\kappa}\cdot\frac{N_{\max}}{K} \;\le\; \left(\frac{1}{\kappa} + \frac{1}{K}\right)\frac{B}{c}.$$

The bound holds for every sequence, so taking expectation over this internal randomness gives

$$\frac{1}{K}\sum_{j=1}^{K} \mathbb{E}[N_j] \;\le\; \left(\frac{1}{\kappa} + \frac{1}{K}\right)\frac{B}{c}.$$

Hence some fixed placement $j^\star$ has expected selection count no larger than this average. This placement may depend on the controller and budget, but not on its realized randomness. Using the value $gN_{j^\star}$ and $\mathrm{OPT} = g\lfloor B/c \rfloor$ yields

$$\mathbb{E}[gN_{j^\star}] \;\le\; \left(\frac{1}{K} + \frac{1}{\kappa}\right)\frac{B/c}{\lfloor B/c \rfloor}\,\mathrm{OPT}.$$

The rounding factor tends to one as $B/c \to \infty$.

Achievability with cost-aware control. For sufficiently large B, select every frontier once at total cost $(1 + (K-1)\kappa)\, c$. The realized costs identify the cheap frontier, which is then selected for the remaining budget. This policy completes at least $\lfloor B/c \rfloor - (K-1)\kappa$ selections on every placement. Ranking by gain per realized cost therefore attains at least $\mathrm{OPT} - (K-1)\kappa g = (1 - o(1))\,\mathrm{OPT}$ as $B/c \to \infty$ with K and κ fixed. □

Remarks. When $K, \kappa \to \infty$ with $K\kappa c/B \to 0$, cost-blind control has a vanishing worst-case approximation ratio, while the constructed cost-aware policy is asymptotically optimal. The vanishing-ratio conclusion requires growing frontier count and cost ratio. Each frontier has a fixed cost, so the construction requires neither heavy-tailed costs nor nonlinear token pricing.

The construction uses equal gains deliberately. Credit based only on score progress receives no signal that separates the frontiers, whether the signal is raw, smoothed, or optimism adjusted. The failure therefore belongs to the cost-blind class rather than a particular rule. Budget termination only fixes the realized prefix length N_j. It cannot alter an earlier action, and no later action contributes to the budgeted value. The allocation question is related to bandits with knapsacks (Badanidiyuru et al., 2013), where budget-constrained optimality also favors arms with high expected gain per unit cost. The broader discovery setting includes frontier development, nonstationarity from growing archives, and interventions. The proposition isolates cost-blindness from these difficulties.

C CostAda Controller Details

C.1 Controller State and Overall Loop

We detail how the three controls use cost-calibrated credit and the remaining budget, with their interaction summarized in Algorithm 1. To track intervention timing, guide outcomes, and temporary control overrides, the controller maintains the evidence state

$$E_t = (\nu_t, L_t, Z_t, o_t, w_t, \kappa_t, \omega_t, G_t, \mathcal{G}_t).$$

Table 3 summarizes the role of each state variable. The controller settings in Table 4 remain fixed across all eight benchmarks without per-task tuning.

Table 3: **CostAda evidence state.** State variables support budgeted tactic intervention and are updated from realized score and cost evidence.

State	Role
ν_t	Counts consecutive steps without meaningful global progress, providing evidence for the stagnation criterion.
L_t	Accumulates realized cost over consecutive steps with negligible utility and allocation reward to track low-yield spending.
Z_t	Records whether guidance has produced substantial global improvement during the run.
o_t	Stores a temporary intervention-mode correction after a failed guide cycle.
w_t	Counts the remaining search steps in the consolidation window.
κ_t	Stores the frontier selected for consolidation while $w_t > 0$.
ω_t	Counts post-step intervention decisions that remain suppressed by guide backoff at the start of iteration t.
G_t	Indicates whether the tactic batch has unused generation attempts remaining after step t.
$\mathcal{G}_t$	Stores an open guide-credit record with its baseline score, mode, and originating frontier; empty after success or unsuccessful exhaustion.

C.2 Local Exploration Control

The local controller favors high-scoring parents in exploitation mode, novel parents in exploration mode, and a mixture of the two in balanced mode. Here, novelty measures code dissimilarity relative

Algorithm 1: CostAda control loop

Input : Initial frontiers $\{D_0^{(k)}\}_{k=1}^{K_0}$, deterministic evaluator F, budget B, reference cost $\bar{c}$, controller constants Θ (Table 4), iteration cap $T_{\max}$

Output : Best-so-far candidate $p^\star$

Initialize $p^\star$, global best y_0, and local bests $\{b_0^{(k)}\}$ from the initial archives; set cumulative cost $C_0 \leftarrow 0$, utility states $\{H_0^{(k)} \leftarrow 0\}$, allocation states $\{R_0^{(k)} \leftarrow 0\}$, visit counts $\{n_0^{(k)} \leftarrow 0\}$, and intervention evidence states

for $t \leftarrow 1$ **to** $T_{\max}$ **do**

`// Choose search actions from the available budget and credit`

Form the pre-step budget state $\rho_{t-1} \leftarrow \max(0, 1 - C_{t-1}/B)$

if $w_{t-1} > 0$ **then**

Set $k_t \leftarrow \kappa_{t-1}$ and $a_t \leftarrow$ exploitation

else

Select k_t using $R_{t-1}^{(k)}$ and the ρ_{t-1}-scaled exploration bonus

Choose a_t using initial sampling or local intensity from $H_{t-1}^{(k_t)}$ and ρ_{t-1}

end

`// Generate and evaluate the candidate`

Execute any scheduled guide that remains available and affordable; open a credit record for a valid tactic batch, storing the baseline, mode, and originating frontier

Draw parent p_t^{par} according to a_t and retrieve context $\mathcal{C}_t$ from local and global archives; use the global best parent while a tactic batch is active

Generate p_t with the context and any active tactic, then evaluate f_t

`// Update the budget from realized search-side cost`

Charge all search-side LLM calls in the step to obtain c_t; update C_t, ρ_t, and normalized cost $\tilde{c}_t$

`// Cost-calibrated credit with remaining-budget conditioning`

Compute gains $\delta_t^{(k_t)}$ and g_t against $b_{t-1}^{(k_t)}$ and y_{t-1}

Compute λ_t, λ_t^c, and d_t; obtain frontier utility $u_t^{(k_t)}$ from Eq. (1) and allocation reward $r_t^{(k_t)} = g_t/d_t$

Update the selected archive $D_t^{(k_t)}$, local and global best scores, and $p^\star$

Smooth $u_t^{(k_t)}$ into $H_t^{(k_t)}$ and $r_t^{(k_t)}$ into $R_t^{(k_t)}$; update visit counts and stagnation and low-yield evidence; carry forward states for unselected frontiers

Evaluate guide success and exhaustion; retain the credited frontier, update Z_t and mode correction o_t, then close any resolved credit record while retaining available tactics

Update w_t, κ_t from guide success and its originating frontier; reset $L_t \leftarrow 0$ when consolidation starts

Apply guide backoff after unsuccessful tactic exhaustion

`// Schedule budgeted tactic intervention from updated evidence`

Compute $(\mathcal{I}_t, m_t)$ using the full intervention rule in Appendix C.4

if $\mathcal{I}_t = 1$ **then**

Schedule a budgeted guide/tactic intervention with mode m_t for the next eligible iteration

end

Update the start-of-next-iteration backoff counter ω_{t+1}

`// Budget stop`

if $C_t \geq B$ **then**

break

end

end

Return $p^\star$

to candidates in the local archive. Initial sampling uses exploration for the first search step and balanced sampling when another frontier is first selected. For a previously selected frontier, the fixed

Table 4: **CostAda controller constants.** The listed values govern local exploration intensity, frontier allocation, and budgeted tactic intervention and remain fixed across all eight benchmarks.

Symbol	Role	Value	Used for
$I_{\min}, I_{\max}$	Local intensity bounds	$0.15,\ 0.50$	Lower and upper bounds on local sampling intensity.
α	Utility smoothing	0.9	Exponential moving average that forms $H_t^{(k)}$ from utility.
γ	Allocation smoothing	0.9	Exponential moving average that forms $R_t^{(k)}$ from allocation reward.
$\lambda_{\min}$	Cost penalty floor	0.25	Early-run lower bound on the cost penalty in d_t.
c_{ucb}	Optimism scale	0.5	Frontier exploration bonus scaled by the remaining-budget ratio.
ϵ_g	Meaningful progress threshold	10^{-4}	Stagnation reset and detection of low-yield steps.
$\epsilon_{\mathrm{strong}}$	Substantial improvement threshold	10^{-3}	Guide success, consolidation window, and default refinement mode, with $\epsilon_{\mathrm{strong}} = 10\epsilon_g$.
ν_0	Base patience scale	6	Budget-dependent stagnation requirement $\nu_{\mathrm{req}}(\rho_t, K_t)$.
$\epsilon_c,\ \epsilon_H$	Numerical stabilizers	$10^{-8},\ 10^{-8}$	Cost normalization and the local intensity denominator.
$\hat{c}_{\mathrm{guide}}$	Guide cost estimate	$2\bar{c}$	Low-yield threshold and guide-cost term of the affordability check.

sampler π_{local} uses the intensity computed from $H_{t-1}^{(k_t)}$ and ρ_{t-1} in Section 4.3. A consolidation window takes precedence and uses exploitation. Outside initial sampling and consolidation, the probabilities are

$$\begin{aligned}\Pr(a_t = \text{exploration}) &= I_t^{(k_t)},\\ \Pr(a_t = \text{exploitation}) &= 0.7(1 - I_t^{(k_t)}),\\ \Pr(a_t = \text{balanced}) &= 0.3(1 - I_t^{(k_t)}).\end{aligned}$$

The mode changes parent selection and prompt guidance but leaves the context request unchanged. Every mode requests four context programs in addition to the parent. Two are high-scoring local candidates chosen for dissimilarity to the parent, and two are the highest-scoring candidates across frontiers. Retrieval excludes the sampled parent and returns fewer context programs when too few candidates are available. During an active tactic batch, the global best candidate replaces the sampled parent, while the retrieved context is retained.

C.3 Frontier Allocation Control

The allocation state records only global progress after cost calibration, so local-only improvement receives no allocation reward. A frontier with only local improvement can still be selected through the exploration bonus. We update this state only for the selected frontier, leaving the other frontier states unchanged. The allocation rule in Section 4.3 selects the next frontier unless a consolidation window directs search to the successful guide's originating frontier. The shared archive scaffold starts with two frontiers and periodically exchanges high-scoring candidates between them. No new frontier was opened in the reported main runs, so $K_t = 2$ throughout.

C.4 Budgeted Tactic Intervention

Budgeted tactic intervention purchases guidance after global stagnation or accumulated low-yield spending, provided that the remaining budget covers the estimated guide cost and a testing reserve. The following rules specify this decision and how guide outcomes affect subsequent search.

C.4.1 Stagnation and Low-Yield Evidence

The stagnation counter ν_t counts consecutive steps with global gain $g_t \leq \epsilon_g$ and resets to zero when $g_t > \epsilon_g$. The stagnation indicator compares this count with a budget-adaptive patience requirement:

$$P_t = [\nu_t \geq \nu_{\mathrm{req}}(\rho_t, K_t)], \qquad \nu_{\mathrm{req}}(\rho_t, K_t) = \max\left(\lceil \nu_0(1-\rho_t) \rceil, 2K_t\right),$$

where $[\cdot]$ equals one when its condition holds and zero otherwise, and ν_0 is the base patience scale. The first term raises the requirement as the budget is spent, so guidance late in the run needs longer stagnation evidence. The lower bound requires at least $2K_t$ consecutive steps without meaningful global improvement, twice the frontier count.

Low-yield spending provides a complementary trigger before the longer stagnation requirement is met. The accumulator L_t sums realized costs c_t over consecutive steps satisfying both $u_t^{(k_t)} \leq \epsilon_g$ and $r_t^{(k_t)} \leq \epsilon_g$, resetting to zero otherwise. This route requires at least half the budget to remain and at least K_t steps without meaningful global improvement, while the stagnation route remains available later. Using the estimated guide cost $\hat{c}_{\mathrm{guide}}$, the low-yield indicator is

$$Y_t = [\rho_t \geq 1/2] \wedge [\nu_t \geq K_t] \wedge [L_t \geq \hat{c}_{\mathrm{guide}}].$$

C.4.2 Guide-Cycle Credit

A guide requests three tactics, which are tested in round-robin order with at most two generation attempts per returned tactic, including retries. When a valid batch is generated, its guide cycle keeps a credit record $\mathcal{G}$ that stores the current global best $y^{\mathcal{G}}_{\mathrm{base}}$, originating frontier $k^{\mathcal{G}}$, and guide mode. While this record is open, accumulated improvement and guide success are measured by

$$g_t^{\mathcal{G}} = \max\left(\frac{y_t - y^{\mathcal{G}}_{\mathrm{base}}}{\max(|y_t|, |y^{\mathcal{G}}_{\mathrm{base}}|, 1)}, 0\right), \qquad \zeta_t = [g_t^{\mathcal{G}} > \epsilon_{\mathrm{strong}}],$$

where $\epsilon_{\mathrm{strong}} = 10\epsilon_g$. We set $g_t^{\mathcal{G}} = 0$ when no credit record is open. The credit record closes at the first success or when the tactics are exhausted without success. Remaining tactics can still be used after success, so G_t may remain one while the post-step credit record $\mathcal{G}_t$ is empty. These later tests do not generate another success event for the resolved record.

C.4.3 Guide Outcomes

A successful guide cycle sets $Z_t = 1$ and opens a consolidation window of K_t subsequent iterations on its originating frontier. During this window, κ_t identifies that frontier and w_t counts the remaining iterations, which use exploitation and suppress new guidance. Success also resets L_t to zero and clears the temporary mode correction and backoff. Remaining tactics can still be tested within this window, but later improvements do not restart it.

If the batch is exhausted without success, new guidance is suppressed on that step and for the next K_t iterations. The backoff counter ω_t tracks the subsequent pause while candidate search and updates to ν_t and L_t continue. We choose the next guide mode m_t after these outcome updates. The default is breakthrough while $Z_t = 0$ and refinement once $Z_t = 1$. A failed refinement cycle sets the temporary correction o_t to request one breakthrough probe. If that probe fails, the correction clears and refinement resumes. When present, the updated correction overrides the default mode.

C.4.4 Affordability and Intervention Decision

Guidance requires budget for the estimated guide cost and a testing reserve of $K_t\bar{c}$, the reference cost of K_t ordinary generations. The affordability indicator is therefore

$$A_t = [B - C_t \geq \hat{c}_{\mathrm{guide}} + K_t\bar{c}].$$

With guidance enabled, the decision combines affordability with stagnation or low-yield evidence:

$$\mathcal{I}_t = A_t \wedge (P_t \vee Y_t),$$

with $\mathcal{I}_t = 0$ during consolidation, while tactics or unresolved guide credit remain, or during the backoff interval defined above. When $\mathcal{I}_t = 1$, mode m_t is attached to the next eligible guide call. Appendix C.5 traces both trigger paths and both modes in one controller run.

C.5 Controller Trace

Table 5 reports the controller state over iterations 6–20 of the CostAda run plotted in Figure 4. The search budget is B=1, and the reference cost is $\bar{c} = 0.01$. The window shows the scheduling of three guide calls and the global improvements that follow. Within this window, uneven action costs and local-only gains produce distinct utility and allocation signals. Local intensity and guide timing reflect both recent progress and the remaining budget.

Table 5: **Controller trace over iterations 6–20 on Signal Processing under GLM-5.** Parentheses give local sampling intensity, and ρ_t is the post-step remaining-budget ratio. Utility follows Eq. (1). Step cost includes guide/tactic calls issued in that iteration. Blank Event cells indicate no controller event during the corresponding search iteration.

t	k_t	Local action	c_t	ρ_t	u_t	r_t	Event
6	1	exploration (0.46)	0.0073	0.92	0.000	0.000	stagnation met, guide scheduled (breakthrough)
7	2	balanced (0.46)	0.0179	0.90	0.056	0.056	guide charged \$0.0084, consolidation opens
8	2	exploitation (0.15)	0.0309	0.87	0.000	0.000	consolidation
9	2	exploitation (0.15)	0.0266	0.84	0.000	0.000	consolidation
10	1	exploration (0.44)	0.0085	0.83	0.050	0.000	local gain, no global gain
11	1	exploration (0.42)	0.0160	0.82	0.000	0.000	
12	1	exploration (0.42)	0.0151	0.80	0.000	0.000	stagnation met, guide scheduled (refinement)
13	2	exploration (0.41)	0.0313	0.77	0.000	0.000	guide charged \$0.0087
14	1	balanced (0.40)	0.0149	0.76	0.000	0.000	
15	2	balanced (0.40)	0.0255	0.73	0.052	0.052	consolidation opens
16	2	exploitation (0.15)	0.0196	0.71	0.000	0.000	consolidation
17	2	exploitation (0.15)	0.0081	0.70	0.000	0.000	low-yield met, guide scheduled (refinement)
18	1	exploration (0.38)	0.0361	0.67	0.000	0.000	guide charged \$0.0116
19	1	exploration (0.37)	0.0146	0.65	0.000	0.000	
20	1	exploration (0.37)	0.0144	0.64	0.062	0.062	consolidation opens

First, realized action costs vary substantially within the same run. Step costs inside the displayed window range from \$0.0073 to \$0.0361, making the costliest step about five times the cheapest. Across the full run, the seven guide/tactic calls cost between \$0.0074 and \$0.0179 each. This variation also appears in the middle panel of Figure 4. Through the cost-calibration factor d_t, more expensive actions receive less credit for a fixed progress mixture and remaining budget.

Second, local refinement and global improvement produce different evidence for subsequent control. At iteration 10, a normalized local gain of 0.0692 improves the frontier archive without improving the global best. Equation (1) assigns a positive cost-calibrated utility of 0.050. The allocation reward is zero because frontier allocation credits only global gain. The positive utility updates $H_t^{(k_t)}$ and thereby affects local sampling intensity, without adding global-gain reward to the allocation state. By contrast, local and global gains coincide at iterations 7, 15, and 20. Utility and allocation reward are equal at 0.056, 0.052, and 0.062, respectively.

Local control combines gradual narrowing as the budget shrinks with temporary exploitation during consolidation. Intensity falls from 0.46 at $\rho_t = 0.92$ to 0.37 by the end of the displayed window. Later in the run, it reaches 0.26 at $\rho_t = 0.32$ and 0.15 at exhaustion. At iterations 8–9 and 16–17, consolidation forces exploitation with intensity 0.15, producing sharper changes within the gradual decline. Sampling returns to exploration when each consolidation window closes, linking these temporary overrides to successful guide cycles.

Intervention responds to both stagnation and low-yield spending, while previous guide outcomes determine whether its default mode seeks breakthrough or refinement. At iteration 6, the global stagnation counter reaches its requirement of four. CostAda schedules a breakthrough guide because

no earlier guide has produced substantial global improvement. The guide is charged \$0.0084 at iteration 7 and is followed by a normalized global gain of 0.0700, which opens a consolidation window. After this success, the guides scheduled at iterations 12 and 17 use refinement mode. Their triggers differ, with stagnation prompting the guide at iteration 12 and low-yield spending prompting it at iteration 17, when $\rho_t = 0.70$. By iteration 17, the run has accumulated \$0.0277 of low-yield spending, above the estimated guide cost of \$0.02 ($2\bar{c}$). The stagnation count of two meets the low-yield requirement of $K_t = 2$ steps but remains below the stagnation requirement of four. Accumulated spending therefore triggers the earlier guide schedule.

Near budget exhaustion, CostAda increases both the required stagnation evidence and the cost penalty on progress. The required stagnation count remains four throughout the displayed window. It rises to five when ρ_t reaches 0.32 and to six by $\rho_t = 0.16$ because patience scales with $1 - \rho_t$. The cost-calibration factor d_t rises from about 1.14 early in the run to 1.72 at $\rho_t = 0.32$, 2.01 at $\rho_t = 0.16$, and 2.48 at $\rho_t = 0.02$. These values reflect both realized step costs and the increasing weight λ_t^c. For a fixed progress mixture and realized cost, a larger λ_t^c reduces utility as the remaining budget shrinks. The run ends with final spending of \$1.0090, exceeding nominal $B{=}1$ by 0.90%.

D Benchmark Details

We follow the benchmark definitions and evaluation setup used by SkyDiscover, AdaEvolve, and EvoX (Liu et al., 2026b; Cemri et al., 2026; Liu et al., 2026a). The underlying tasks and reference values draw on AlphaEvolve, packing records, Heilbronn's problem, generalized Sidon sets, OpenEvolve, and standard signal-processing references (Novikov et al., 2025; Friedman, 2025; Roth, 1951; Vinuesa del Río, 2010; Sharma, 2025; Shenoi, 2005). The controller uses the higher-is-better evaluator score $F(p)$ to compute local and global gains in Section 4. Table 6 lists each benchmark's native objective and optimization direction for reporting results. Appendix L shows final CostAda programs for Circle Packing and MinMaxDist ($n{=}16, d{=}2$).

Table 6: **Benchmark objectives and optimization directions.** Rows state the benchmark objective and whether higher or lower values are preferred.

Benchmark	Direction	Objective
Circle Packing	↑	Pack 26 disjoint circles in a unit square and maximize the sum of radii.
Circle Packing Rect	↑	Pack 21 disjoint circles in a rectangle of perimeter 4 and maximize the sum of radii over all feasible circle configurations.
Heilbronn Triangle	↑	Place 11 points in a unit area triangle and maximize the minimum area over all triangles formed by triples of points.
Heilbronn Convex	↑	Place 13 points in a unit area convex region and maximize the minimum triangle area induced by any three points.
MinMaxDist ($n{=}16, d{=}2$)	↓	Place 16 points in two dimensions and minimize the ratio between maximum and minimum squared pairwise distances (Novikov et al., 2025).
MinMaxDist ($n{=}14, d{=}3$)	↓	Place 14 points in three dimensions and minimize the ratio between maximum and minimum squared pairwise distances.
Third Autocorrelation	↓	Construct witness functions that improve the upper bound on the third autocorrelation constant C_3 in additive combinatorics.
Signal Processing	↑	Synthesize a filtering program for noisy nonstationary time series, balancing fidelity, smoothness, lag, and false trend changes.

For Signal Processing, candidates are ranked by `combined_score`, which is identical to `overall_score`. The evaluator combines the original composite score, smoothness score, correlation/accuracy score, noise-reduction score, and success rate with weights 0.4, 0.2, 0.2, 0.1, and 0.1, respectively. The score is set to zero when the correlation/accuracy score is below 0.1. We use this score for all Signal Processing cutoff and normalized-AUC results.

E Experimental Accounting Details

E.1 Search-Side Cost

If iteration t issues a set of search-side LLM calls $\mathcal{J}_t$, its realized cost is

$$c_t = \sum_{j \in \mathcal{J}_t} \left(p_{\text{in}}^{(\mu_j)} T_{\text{in}}^{(j)} + p_{\text{out}}^{(\mu_j)} T_{\text{out}}^{(j)} \right),$$

where μ_j is the model used by call j, $T_{\text{in}}^{(j)}$ and $T_{\text{out}}^{(j)}$ are provider-reported input and output token counts, and $p_{\text{in}}^{(\mu_j)}$, $p_{\text{out}}^{(\mu_j)}$ are the corresponding fixed per-token prices. We use the same OpenRouter prices for all methods within each backbone. GLM-5 input/output prices are \$0.60/\$1.92 per million tokens (OpenRouter, 2026a), and GPT-5.4 prices are \$2.50/\$15.00 per million tokens (OpenRouter, 2026b). We apply these fixed rates without cache discounts.

Search-side calls include ordinary candidate generation, recovery from failed generations, and guide/tactic generation. The scalar controller updates require no additional LLM calls. Their CPU time and deterministic evaluator execution are excluded. All reported budgets use this ledger.

The normalized step cost $\tilde{c}_t$ uses reference generation costs of $\bar{c}{=}0.01$ for GLM-5 and $\bar{c}{=}0.1$ for GPT-5.4. We measured these values before the main comparisons as the average realized search-side cost of an ordinary generation call. The reference cost is fixed per backbone and shared across benchmarks. It scales the controller's cost credit without changing the monetary evaluation ledger.

E.2 Shared Experimental Scaffold

We instantiate all methods in the same discovery framework. Candidate-generation infrastructure, parsing, deterministic evaluator execution, and archive insertion are shared across methods. For the main comparisons, generator and guide model choices are fixed across all eight benchmarks under each backbone. Every method uses the same fixed prices.

F Budget-Cutoff Results

The cutoff results below support the budget-path analysis in Section 5.3. Tables 7 and 8 report the results at the budget cutoffs $0.25B$, $0.5B$, $0.75B$, and B under GLM-5 and GPT-5.4, respectively. We evaluate all three methods at the same four cutoffs. We omit the zero-cost initialization because it precedes candidate generation and may lack a benchmark-objective value.

In Tables 7 and 8, CostAda reaches the strongest baseline's full-budget quality with at most half the budget on six of the eight benchmarks under GLM-5 and seven under GPT-5.4. CostAda also leads or ties 30 of the 32 benchmark-objective cells under GLM-5. Under GPT-5.4, it leads 29 of the 32 cells. From $0.75B$ onward, it leads on all eight benchmarks under both backbones. These per-benchmark results agree with the early and persistent lead in Figure 3.

G Small-Budget Evaluation

To validate the early-budget comparisons under GLM-5, we additionally evaluate CostAda in independent runs with a smaller search budget (Table 9). Within each benchmark block, the first three rows report cutoff values from $B{=}1$ trajectories at the corresponding absolute costs. CostAda$_{B=0.5}$ instead reports three independent runs configured with search budget $B{=}0.5$ and evaluated at the same absolute costs. These runs can adapt their search to the smaller horizon, whereas a cutoff only truncates a trajectory planned for $B{=}1$.

CostAda's early-budget advantage persists when the smaller budget is configured before search. In Table 9, the independently budgeted CostAda$_{B=0.5}$ runs outperform the strongest baseline result in all 16 benchmark–cutoff cells. This comparison is made at the same absolute costs, with baseline values read from $B{=}1$ trajectories. The CostAda$_{B=0.5}$ runs also match or improve on the corresponding CostAda cutoff values from $B{=}1$ trajectories in 14 cells. These independent runs therefore support the budget-cutoff comparisons under GLM-5.

Table 7: **Benchmark objectives across GLM-5 budget cutoffs.** With search budget B=1, the columns correspond to costs 0.25, 0.5, 0.75, and 1.0 in ledger dollars. Entries report **Mean** $\pm$ **Std** in the units of Table 1. Arrows indicate the preferred direction. Best entries per benchmark and cutoff are bolded, including ties at the reported precision.

Benchmark	Method	$0.25B$	$0.5B$	$0.75B$	B
Circle Packing ↑	AdaEvolve	1.8225 ± 0.2223	2.4690 ± 0.1562	2.5617 ± 0.0773	2.5617 ± 0.0773
	EvoX	2.1815 ± 0.0801	2.5128 ± 0.1269	2.5717 ± 0.0752	2.5717 ± 0.0752
	CostAda	$\mathbf{2.5535 \pm 0.0585}$	$\mathbf{2.6016 \pm 0.0335}$	$\mathbf{2.6252 \pm 0.0094}$	$\mathbf{2.6306 \pm 0.0054}$
Circle Packing Rect ↑	AdaEvolve	2.1182 ± 0.1146	2.2043 ± 0.1578	2.2089 ± 0.1635	2.2089 ± 0.1635
	EvoX	2.1985 ± 0.2569	2.1989 ± 0.2563	2.1989 ± 0.2563	2.1989 ± 0.2563
	CostAda	$\mathbf{2.3478 \pm 0.0081}$	$\mathbf{2.3478 \pm 0.0081}$	$\mathbf{2.3478 \pm 0.0081}$	$\mathbf{2.3525 \pm 0.0081}$
Heilbronn Convex ↑	AdaEvolve	$\mathbf{0.0215 \pm 0.0030}$	0.0215 ± 0.0030	0.0229 ± 0.0015	0.0229 ± 0.0015
	EvoX	0.0203 ± 0.0036	0.0222 ± 0.0024	0.0234 ± 0.0012	0.0235 ± 0.0012
	CostAda	$\mathbf{0.0215 \pm 0.0003}$	$\mathbf{0.0250 \pm 0.0030}$	$\mathbf{0.0253 \pm 0.0033}$	$\mathbf{0.0263 \pm 0.0018}$
Heilbronn Triangle ↑	AdaEvolve	0.0212 ± 0.0075	0.0283 ± 0.0020	0.0283 ± 0.0020	0.0303 ± 0.0011
	EvoX	0.0149 ± 0.0027	0.0200 ± 0.0038	0.0200 ± 0.0038	0.0220 ± 0.0050
	CostAda	$\mathbf{0.0296 \pm 0.0017}$	$\mathbf{0.0296 \pm 0.0017}$	$\mathbf{0.0296 \pm 0.0017}$	$\mathbf{0.0314 \pm 0.0003}$
MinMaxDist (n=16, d=2) ↓	AdaEvolve	14.59 ± 1.03	14.54 ± 1.02	14.06 ± 1.30	14.06 ± 1.30
	EvoX	15.01 ± 1.69	14.37 ± 2.11	14.20 ± 1.90	14.17 ± 1.92
	CostAda	$\mathbf{13.62 \pm 1.02}$	$\mathbf{13.57 \pm 1.07}$	$\mathbf{13.15 \pm 0.36}$	$\mathbf{13.11 \pm 0.29}$
MinMaxDist (n=14, d=3) ↓	AdaEvolve	4.57 ± 0.08	4.48 ± 0.13	4.38 ± 0.08	4.38 ± 0.08
	EvoX	4.59 ± 0.37	4.34 ± 0.27	4.22 ± 0.06	4.20 ± 0.03
	CostAda	$\mathbf{4.31 \pm 0.24}$	$\mathbf{4.19 \pm 0.04}$	$\mathbf{4.18 \pm 0.02}$	$\mathbf{4.17 \pm 0.00}$
Third Autocorrelation ↓	AdaEvolve	$\mathbf{1.4711 \pm 0.0040}$	$\mathbf{1.4694 \pm 0.0017}$	1.4694 ± 0.0017	1.4685 ± 0.0014
	EvoX	1.5760 ± 0.1729	1.5719 ± 0.1658	1.5481 ± 0.1308	1.5481 ± 0.1308
	CostAda	1.4858 ± 0.0256	1.4746 ± 0.0064	$\mathbf{1.4679 \pm 0.0054}$	$\mathbf{1.4658 \pm 0.0037}$
Signal Processing ↑	AdaEvolve	0.5692 ± 0.0301	0.6126 ± 0.0161	0.6173 ± 0.0158	0.6393 ± 0.0240
	EvoX	0.5122 ± 0.0177	0.6361 ± 0.0655	0.6429 ± 0.0544	0.6502 ± 0.0430
	CostAda	$\mathbf{0.6377 \pm 0.0805}$	$\mathbf{0.6924 \pm 0.0536}$	$\mathbf{0.7069 \pm 0.0318}$	$\mathbf{0.7222 \pm 0.0127}$

H Ablation Study

We detail the ablation study of Section 5.4 on Circle Packing, Heilbronn Triangle, and Signal Processing under both backbones, covering packing, geometric point configuration, and executable program synthesis. The study compares CostAda with three variants. The variant w/o cost calibration sets the cost-calibration factor d_t to one while retaining the remaining-budget and intervention rules. The variant w/o budget conditioning fixes $\lambda_t = \lambda_t^c = 0.5$ and removes ρ_t from local exploration intensity and the exploration bonus, while leaving intervention gating unchanged. The variant w/o intervention gating schedules guidance from stagnation evidence without the A_t and Y_t gates, while retaining guide backoff, the consolidation window, and the single-active-batch constraint.

For each run, BestScore@$0.5B$ and BestScore@B report the benchmark objective of the best-so-far candidate at the corresponding cutoff, with candidates ranked by normalized score. Normalized AUC is the average stepwise best-so-far normalized score over $[0, B]$, obtained by integrating over realized cost and dividing by B. Each action is charged before its score takes effect.

Tables 10 and 11 show that the AUC gaps in Table 2 extend to the benchmark objectives at both cutoffs. Every displayed ablation result is lower than CostAda under both backbones. In particular, the lower cutoff objectives and AUC of w/o intervention gating occur while guidance remains available, tying this comparison to when guides are purchased and tested. On Signal Processing under GLM-5, the w/o cost calibration values are 4.86%, 5.13%, and 2.91% below CostAda for BestScore@$0.5B$, BestScore@B, and normalized AUC, respectively. Under GLM-5, removing remaining-budget conditioning lowers the final Signal Processing objective below the strongest baseline mean in Table 1. Under GPT-5.4, removing either cost calibration or remaining-budget conditioning also lowers it below the strongest baseline mean.

Table 8: **Benchmark objectives across GPT-5.4 budget cutoffs.** With search budget B=5, the columns correspond to costs 1.25, 2.5, 3.75, and 5.0 in ledger dollars. Entries report **Mean** $\pm$ **Std** in the units of Table 1. Arrows indicate the preferred direction. Best entries per benchmark and cutoff are bolded, including ties at the reported precision.

Benchmark	Method	$0.25B$	$0.5B$	$0.75B$	B
Circle Packing ↑	AdaEvolve	2.5632 ± 0.0784	2.5827 ± 0.0510	2.5865 ± 0.0535	2.6196 ± 0.0030
	EvoX	2.5225 ± 0.0328	2.5441 ± 0.0609	2.5818 ± 0.0476	2.5851 ± 0.0510
	CostAda	**2.6143 ± 0.0140**	**2.6211 ± 0.0033**	**2.6218 ± 0.0044**	**2.6248 ± 0.0046**
Circle Packing Rect ↑	AdaEvolve	2.2826 ± 0.0410	2.2910 ± 0.0507	2.3125 ± 0.0370	2.3148 ± 0.0357
	EvoX	**2.3221 ± 0.0428**	**2.3474 ± 0.0011**	2.3474 ± 0.0011	2.3474 ± 0.0011
	CostAda	2.2689 ± 0.0313	2.3111 ± 0.0462	**2.3491 ± 0.0128**	**2.3577 ± 0.0050**
Heilbronn Convex ↑	AdaEvolve	0.0187 ± 0.0007	0.0187 ± 0.0007	0.0187 ± 0.0007	0.0197 ± 0.0020
	EvoX	0.0166 ± 0.0018	0.0197 ± 0.0026	0.0212 ± 0.0021	0.0223 ± 0.0031
	CostAda	**0.0234 ± 0.0010**	**0.0234 ± 0.0010**	**0.0244 ± 0.0008**	**0.0244 ± 0.0008**
Heilbronn Triangle ↑	AdaEvolve	0.0280 ± 0.0003	0.0290 ± 0.0016	0.0299 ± 0.0014	0.0307 ± 0.0008
	EvoX	0.0240 ± 0.0022	0.0254 ± 0.0019	0.0260 ± 0.0030	0.0263 ± 0.0035
	CostAda	**0.0288 ± 0.0039**	**0.0317 ± 0.0014**	**0.0329 ± 0.0012**	**0.0330 ± 0.0009**
MinMaxDist (n=16, d=2) ↓	AdaEvolve	13.49 ± 0.85	13.37 ± 0.65	13.14 ± 0.24	13.00 ± 0.00
	EvoX	**13.00 ± 0.00**	13.00 ± 0.00	13.00 ± 0.00	13.00 ± 0.00
	CostAda	13.16 ± 0.20	**12.98 ± 0.09**	**12.93 ± 0.06**	**12.93 ± 0.06**
MinMaxDist (n=14, d=3) ↓	AdaEvolve	4.64 ± 0.15	4.61 ± 0.14	4.61 ± 0.14	4.55 ± 0.15
	EvoX	4.64 ± 0.16	4.46 ± 0.01	4.44 ± 0.02	4.44 ± 0.02
	CostAda	**4.55 ± 0.15**	**4.42 ± 0.08**	**4.42 ± 0.08**	**4.42 ± 0.08**
Third Autocorrelation ↓	AdaEvolve	1.5433 ± 0.0278	1.5157 ± 0.0079	1.5157 ± 0.0079	1.5081 ± 0.0069
	EvoX	1.4899 ± 0.0028	1.4853 ± 0.0043	1.4838 ± 0.0029	1.4824 ± 0.0042
	CostAda	**1.4748 ± 0.0056**	**1.4738 ± 0.0048**	**1.4717 ± 0.0074**	**1.4717 ± 0.0074**
Signal Processing ↑	AdaEvolve	0.6617 ± 0.0040	0.6739 ± 0.0145	0.7043 ± 0.0024	0.7182 ± 0.0137
	EvoX	0.5890 ± 0.0447	0.6436 ± 0.0509	0.6644 ± 0.0299	0.6767 ± 0.0179
	CostAda	**0.7069 ± 0.0402**	**0.7289 ± 0.0262**	**0.7289 ± 0.0262**	**0.7672 ± 0.0292**

I Normalized Score Results

Benchmark objectives differ in units, magnitude, and direction across the eight benchmarks. We therefore use a normalized score for the cross-benchmark cutoff and AUC analyses in Section 5.3. The shared evaluator reports this direction-adjusted quality measure for every candidate. For benchmarks with a fixed reference value, the normalized score is the ratio between the achieved objective and that reference. We invert the ratio for minimization benchmarks so that higher is always better. We adopt the reference values reported in prior work (Novikov et al., 2025; Cemri et al., 2026; Liu et al., 2026a). A value of 1 matches the reference, values above 1 exceed it, and invalid candidates receive a score of zero. For Signal Processing, the evaluator already produces a bounded composite quality measure, so we use its overall program score in $[0, 1]$. Normalization constants are fixed per benchmark and shared by all methods. For individual candidates, normalization preserves each benchmark's native preference ordering.

Table 12 reports per-benchmark normalized AUC over the full search budget, which underlies the AUC comparison in Section 5.3. Tables 13 and 14 report best-so-far normalized scores at the same cutoffs as the benchmark-objective tables.

CostAda achieves the highest normalized AUC on fifteen of the sixteen benchmark–backbone pairs in Table 12. Its eight-benchmark average is also the highest under GLM-5 and GPT-5.4, at 0.8626 and 0.8851 compared with 0.8060 and 0.8437 for AdaEvolve. CostAda's cutoff-cell rankings also agree between normalized-score and benchmark-objective reporting. In Tables 13 and 14, CostAda is best or tied in 30 of the 32 cells under GLM-5 and in 29 under GPT-5.4, the same counts as in Appendix F. This agreement spans both maximization and minimization benchmarks after their score directions are aligned, so normalized scores support the aggregate comparison.

Table 9: **Benchmark objectives for independent B=0.5 runs and B=1 trajectories under GLM-5.** The first three rows of each benchmark block come from B=1 trajectories. CostAda$_{B=0.5}$ denotes runs configured with the smaller search budget and evaluated at the same costs, \$0.25 and \$0.5. Entries report **Mean ± Std** in the units of Table 1. Arrows indicate the preferred direction. Best entries per benchmark and cutoff are bolded, including ties at the reported precision.

Benchmark	Method	$0.25B$	$0.5B$
Circle Packing ↑	AdaEvolve	1.8225 ± 0.2223	2.4690 ± 0.1562
	EvoX	2.1815 ± 0.0801	2.5128 ± 0.1269
	CostAda	2.5535 ± 0.0585	2.6016 ± 0.0335
	CostAda$_{B=0.5}$	**2.6146 ± 0.0176**	**2.6233 ± 0.0044**
Circle Packing Rect ↑	AdaEvolve	2.1182 ± 0.1146	2.2043 ± 0.1578
	EvoX	2.1985 ± 0.2569	2.1989 ± 0.2563
	CostAda	**2.3478 ± 0.0081**	**2.3478 ± 0.0081**
	CostAda$_{B=0.5}$	2.3404 ± 0.0008	2.3404 ± 0.0008
Heilbronn Convex ↑	AdaEvolve	0.0215 ± 0.0030	0.0215 ± 0.0030
	EvoX	0.0203 ± 0.0036	0.0222 ± 0.0024
	CostAda	0.0215 ± 0.0003	0.0250 ± 0.0030
	CostAda$_{B=0.5}$	**0.0253 ± 0.0011**	**0.0253 ± 0.0011**
Heilbronn Triangle ↑	AdaEvolve	0.0212 ± 0.0075	0.0283 ± 0.0020
	EvoX	0.0149 ± 0.0027	0.0200 ± 0.0038
	CostAda	0.0296 ± 0.0017	0.0296 ± 0.0017
	CostAda$_{B=0.5}$	**0.0302 ± 0.0036**	**0.0310 ± 0.0023**
MinMaxDist (n=16, d=2) ↓	AdaEvolve	14.59 ± 1.03	14.54 ± 1.02
	EvoX	15.01 ± 1.69	14.37 ± 2.11
	CostAda	13.62 ± 1.02	13.57 ± 1.07
	CostAda$_{B=0.5}$	**13.41 ± 0.46**	**13.13 ± 0.21**
MinMaxDist (n=14, d=3) ↓	AdaEvolve	4.57 ± 0.08	4.48 ± 0.13
	EvoX	4.59 ± 0.37	4.34 ± 0.27
	CostAda	4.31 ± 0.24	4.19 ± 0.04
	CostAda$_{B=0.5}$	**4.23 ± 0.09**	**4.17 ± 0.00**
Third Autocorrelation ↓	AdaEvolve	1.4711 ± 0.0040	1.4694 ± 0.0017
	EvoX	1.5760 ± 0.1729	1.5719 ± 0.1658
	CostAda	1.4858 ± 0.0256	1.4746 ± 0.0064
	CostAda$_{B=0.5}$	**1.4685 ± 0.0062**	**1.4685 ± 0.0062**
Signal Processing ↑	AdaEvolve	0.5692 ± 0.0301	0.6126 ± 0.0161
	EvoX	0.5122 ± 0.0177	0.6361 ± 0.0655
	CostAda	0.6377 ± 0.0805	0.6924 ± 0.0536
	CostAda$_{B=0.5}$	**0.6690 ± 0.0095**	**0.7120 ± 0.0169**

Table 10: **Ablation study of CostAda under GLM-5.** Entries report **Mean ± Std**. Benchmark objectives are used for the two cutoff columns, and higher normalized AUC is better. Arrows indicate the preferred direction. Best entries per benchmark and metric among CostAda and its variants are bolded, including ties at the reported precision.

Benchmark	Variant	BestScore@$0.5B$	BestScore@B	Norm. AUC ↑
Circle Packing ↑	CostAda	**2.6195 ± 0.0128**	**2.6281 ± 0.0067**	**0.9517 ± 0.0095**
	w/o cost calibration	2.5389 ± 0.1153	2.5738 ± 0.0554	0.9053 ± 0.0376
	w/o budget conditioning	2.6074 ± 0.0125	2.6172 ± 0.0057	0.9161 ± 0.0316
	w/o intervention gating	2.3594 ± 0.2251	2.5291 ± 0.0949	0.8478 ± 0.0642
Heilbronn Triangle ↑	CostAda	**0.0306 ± 0.0009**	**0.0313 ± 0.0004**	**0.7831 ± 0.0486**
	w/o cost calibration	0.0167 ± 0.0032	0.0228 ± 0.0040	0.4654 ± 0.0614
	w/o budget conditioning	0.0212 ± 0.0048	0.0251 ± 0.0049	0.4073 ± 0.0426
	w/o intervention gating	0.0206 ± 0.0085	0.0244 ± 0.0035	0.4494 ± 0.1622
Signal Processing ↑	CostAda	**0.6924 ± 0.0536**	**0.7222 ± 0.0127**	**0.6546 ± 0.0444**
	w/o cost calibration	0.6588 ± 0.0254	0.6851 ± 0.0331	0.6355 ± 0.0200
	w/o budget conditioning	0.6322 ± 0.0130	0.6470 ± 0.0029	0.6146 ± 0.0134
	w/o intervention gating	0.6763 ± 0.0059	0.7059 ± 0.0220	0.6293 ± 0.0140

Table 11: **Ablation study of CostAda under GPT-5.4.** Entries report **Mean** $\pm$ **Std**. Benchmark objectives are used for the two cutoff columns, and higher normalized AUC is better. Arrows indicate the preferred direction. Best entries per benchmark and metric among CostAda and its variants are bolded, including ties at the reported precision.

Benchmark	Variant	BESTSCORE@$0.5B$	BESTSCORE@B	Norm. AUC ↑
Circle Packing ↑	CostAda	**2.6197 ± 0.0080**	**2.6212 ± 0.0073**	**0.9825 ± 0.0050**
	w/o cost calibration	2.5963 ± 0.0293	2.6047 ± 0.0364	0.9648 ± 0.0276
	w/o budget conditioning	2.5898 ± 0.0421	2.5942 ± 0.0458	0.9673 ± 0.0079
	w/o intervention gating	2.5183 ± 0.0455	2.5223 ± 0.0386	0.9402 ± 0.0246
Heilbronn Triangle ↑	CostAda	**0.0309 ± 0.0010**	**0.0326 ± 0.0008**	**0.7951 ± 0.0253**
	w/o cost calibration	0.0252 ± 0.0021	0.0274 ± 0.0018	0.6748 ± 0.0459
	w/o budget conditioning	0.0290 ± 0.0033	0.0291 ± 0.0033	0.7494 ± 0.0615
	w/o intervention gating	0.0266 ± 0.0009	0.0268 ± 0.0007	0.6949 ± 0.0259
Signal Processing ↑	CostAda	**0.7289 ± 0.0262**	**0.7672 ± 0.0292**	**0.7022 ± 0.0224**
	w/o cost calibration	0.6617 ± 0.0085	0.7059 ± 0.0191	0.6333 ± 0.0028
	w/o budget conditioning	0.6889 ± 0.0240	0.7086 ± 0.0321	0.6557 ± 0.0192
	w/o intervention gating	0.6931 ± 0.0306	0.7139 ± 0.0156	0.6717 ± 0.0116

Table 12: **Normalized AUC over the full search budget.** Entries report **Mean** $\pm$ **Std** of the average best-so-far normalized score over $[0, B]$, with B=1 for GLM-5 and B=5 for GPT-5.4. Higher is better. Best entries per benchmark and backbone are bolded, including ties at the reported precision. The Average row averages the eight per-benchmark entries.

Benchmark	GLM-5 (B=1)			GPT-5.4 (B=5)		
	AdaEvolve	EvoX	CostAda	AdaEvolve	EvoX	CostAda
Circle Packing	0.8341 ± 0.0273	0.8752 ± 0.0289	**0.9295 ± 0.0134**	0.9668 ± 0.0197	0.9548 ± 0.0172	**0.9771 ± 0.0096**
Circle Packing Rect	0.9009 ± 0.0629	0.8386 ± 0.1738	**0.9551 ± 0.0133**	0.9630 ± 0.0139	0.9668 ± 0.0124	**0.9672 ± 0.0058**
Heilbronn Convex	0.6812 ± 0.0840	0.6579 ± 0.0560	**0.7606 ± 0.0700**	0.5896 ± 0.0243	0.6103 ± 0.0645	**0.7499 ± 0.0144**
Heilbronn Triangle	0.6571 ± 0.0658	0.4700 ± 0.0372	**0.7613 ± 0.0531**	0.7825 ± 0.0132	0.6463 ± 0.0632	**0.8229 ± 0.0387**
MinMaxDist (n=16, d=2)	0.8864 ± 0.0588	0.8530 ± 0.0907	**0.9241 ± 0.0370**	0.9556 ± 0.0343	0.9651 ± 0.0165	**0.9654 ± 0.0098**
MinMaxDist (n=14, d=3)	0.9172 ± 0.0068	0.9232 ± 0.0473	**0.9502 ± 0.0082**	0.8947 ± 0.0240	0.9031 ± 0.0041	**0.9186 ± 0.0077**
Third Autocorrelation	**0.9820 ± 0.0027**	0.9177 ± 0.1009	0.9658 ± 0.0050	0.9311 ± 0.0102	0.9609 ± 0.0041	**0.9776 ± 0.0081**
Signal Processing	0.5891 ± 0.0149	0.5873 ± 0.0366	**0.6546 ± 0.0444**	0.6660 ± 0.0024	0.6216 ± 0.0292	**0.7022 ± 0.0224**
Average	0.8060	0.7654	**0.8626**	0.8437	0.8286	**0.8851**

Table 13: **Best-so-far normalized scores across GLM-5 budget cutoffs.** The search budget is B=1, with the same budget fractions as Table 7. Entries report **Mean** ± **Std**. Higher is better. Best entries per benchmark and cutoff are bolded, including ties at the reported precision.

Benchmark	Method	$0.25B$	$0.5B$	$0.75B$	B
Circle Packing	AdaEvolve	0.6917 ± 0.0844	0.9370 ± 0.0593	0.9722 ± 0.0293	0.9722 ± 0.0293
	EvoX	0.8279 ± 0.0304	0.9536 ± 0.0481	0.9760 ± 0.0285	0.9760 ± 0.0285
	CostAda	**0.9691 ± 0.0222**	**0.9873 ± 0.0127**	**0.9963 ± 0.0036**	**0.9983 ± 0.0021**
Circle Packing Rect	AdaEvolve	0.8953 ± 0.0484	0.9317 ± 0.0667	0.9336 ± 0.0691	0.9336 ± 0.0691
	EvoX	0.9293 ± 0.1086	0.9295 ± 0.1083	0.9295 ± 0.1083	0.9295 ± 0.1083
	CostAda	**0.9924 ± 0.0034**	**0.9924 ± 0.0034**	**0.9924 ± 0.0034**	**0.9944 ± 0.0034**
Heilbronn Convex	AdaEvolve	0.6954 ± 0.0963	0.6954 ± 0.0963	0.7410 ± 0.0497	0.7410 ± 0.0497
	EvoX	0.6556 ± 0.1173	0.7166 ± 0.0790	0.7563 ± 0.0372	0.7608 ± 0.0386
	CostAda	**0.6957 ± 0.0104**	**0.8096 ± 0.0974**	**0.8190 ± 0.1078**	**0.8496 ± 0.0575**
Heilbronn Triangle	AdaEvolve	0.5816 ± 0.2042	0.7734 ± 0.0535	0.7734 ± 0.0535	0.8300 ± 0.0309
	EvoX	0.4088 ± 0.0730	0.5488 ± 0.1039	0.5488 ± 0.1039	0.6024 ± 0.1364
	CostAda	**0.8098 ± 0.0452**	**0.8098 ± 0.0452**	**0.8098 ± 0.0452**	**0.8595 ± 0.0085**
MinMaxDist (n=16, d=2)	AdaEvolve	0.8861 ± 0.0630	0.8891 ± 0.0626	0.9214 ± 0.0811	0.9214 ± 0.0811
	EvoX	0.8657 ± 0.0957	0.9090 ± 0.1232	0.9179 ± 0.1138	0.9201 ± 0.1156
	CostAda	**0.9495 ± 0.0681**	**0.9540 ± 0.0721**	**0.9805 ± 0.0264**	**0.9838 ± 0.0212**
MinMaxDist (n=14, d=3)	AdaEvolve	0.9119 ± 0.0170	0.9298 ± 0.0264	0.9512 ± 0.0168	0.9512 ± 0.0168
	EvoX	0.9120 ± 0.0766	0.9612 ± 0.0568	0.9864 ± 0.0142	0.9909 ± 0.0076
	CostAda	**0.9685 ± 0.0522**	**0.9936 ± 0.0094**	**0.9964 ± 0.0045**	**0.9994 ± 0.0006**
Third Autocorrelation	AdaEvolve	**0.9895 ± 0.0027**	**0.9907 ± 0.0011**	0.9907 ± 0.0011	0.9913 ± 0.0009
	EvoX	0.9307 ± 0.0960	0.9326 ± 0.0927	0.9446 ± 0.0761	0.9446 ± 0.0761
	CostAda	0.9799 ± 0.0167	0.9871 ± 0.0043	**0.9916 ± 0.0036**	**0.9931 ± 0.0025**
Signal Processing	AdaEvolve	0.5692 ± 0.0301	0.6126 ± 0.0161	0.6173 ± 0.0158	0.6393 ± 0.0240
	EvoX	0.5122 ± 0.0177	0.6361 ± 0.0655	0.6429 ± 0.0544	0.6502 ± 0.0430
	CostAda	**0.6377 ± 0.0805**	**0.6924 ± 0.0536**	**0.7069 ± 0.0318**	**0.7222 ± 0.0127**

Table 14: **Best-so-far normalized scores across GPT-5.4 budget cutoffs.** The search budget is B=5, with the same budget fractions as Table 8. Entries report **Mean** ± **Std**. Higher is better. Best entries per benchmark and cutoff are bolded, including ties at the reported precision.

Benchmark	Method	$0.25B$	$0.5B$	$0.75B$	B
Circle Packing	AdaEvolve	0.9727 ± 0.0297	0.9801 ± 0.0194	0.9816 ± 0.0203	0.9941 ± 0.0011
	EvoX	0.9573 ± 0.0125	0.9655 ± 0.0231	0.9798 ± 0.0181	0.9811 ± 0.0193
	CostAda	**0.9921 ± 0.0053**	**0.9947 ± 0.0012**	**0.9950 ± 0.0017**	**0.9961 ± 0.0017**
Circle Packing Rect	AdaEvolve	0.9648 ± 0.0173	0.9684 ± 0.0214	0.9775 ± 0.0156	0.9784 ± 0.0151
	EvoX	**0.9815 ± 0.0181**	**0.9922 ± 0.0004**	0.9922 ± 0.0004	0.9922 ± 0.0004
	CostAda	0.9590 ± 0.0132	0.9769 ± 0.0195	**0.9929 ± 0.0054**	**0.9966 ± 0.0021**
Heilbronn Convex	AdaEvolve	0.6034 ± 0.0242	0.6034 ± 0.0242	0.6049 ± 0.0222	0.6376 ± 0.0640
	EvoX	0.5353 ± 0.0594	0.6354 ± 0.0829	0.6840 ± 0.0663	0.7194 ± 0.1014
	CostAda	**0.7574 ± 0.0319**	**0.7574 ± 0.0319**	**0.7885 ± 0.0243**	**0.7885 ± 0.0243**
Heilbronn Triangle	AdaEvolve	0.7654 ± 0.0088	0.7941 ± 0.0425	0.8183 ± 0.0382	0.8395 ± 0.0219
	EvoX	0.6577 ± 0.0599	0.6940 ± 0.0507	0.7124 ± 0.0824	0.7205 ± 0.0956
	CostAda	**0.7897 ± 0.1072**	**0.8675 ± 0.0376**	**0.8997 ± 0.0334**	**0.9042 ± 0.0258**
MinMaxDist (n=16, d=2)	AdaEvolve	0.9579 ± 0.0582	0.9653 ± 0.0453	0.9811 ± 0.0180	0.9914 ± 0.0002
	EvoX	**0.9915 ± 0.0000**	0.9915 ± 0.0000	0.9915 ± 0.0000	0.9915 ± 0.0000
	CostAda	0.9796 ± 0.0149	**0.9929 ± 0.0065**	**0.9972 ± 0.0049**	**0.9972 ± 0.0049**
MinMaxDist (n=14, d=3)	AdaEvolve	0.8980 ± 0.0305	0.9045 ± 0.0267	0.9045 ± 0.0267	0.9156 ± 0.0305
	EvoX	0.8991 ± 0.0325	0.9347 ± 0.0018	0.9376 ± 0.0050	0.9376 ± 0.0050
	CostAda	**0.9156 ± 0.0305**	**0.9434 ± 0.0177**	**0.9434 ± 0.0177**	**0.9434 ± 0.0177**
Third Autocorrelation	AdaEvolve	0.9434 ± 0.0169	0.9604 ± 0.0050	0.9604 ± 0.0050	0.9652 ± 0.0044
	EvoX	0.9770 ± 0.0018	0.9801 ± 0.0028	0.9810 ± 0.0019	0.9820 ± 0.0028
	CostAda	**0.9870 ± 0.0038**	**0.9877 ± 0.0032**	**0.9891 ± 0.0050**	**0.9891 ± 0.0050**
Signal Processing	AdaEvolve	0.6617 ± 0.0040	0.6739 ± 0.0145	0.7043 ± 0.0024	0.7182 ± 0.0137
	EvoX	0.5890 ± 0.0447	0.6436 ± 0.0509	0.6644 ± 0.0299	0.6767 ± 0.0179
	CostAda	**0.7069 ± 0.0402**	**0.7289 ± 0.0262**	**0.7289 ± 0.0262**	**0.7672 ± 0.0292**

J Objective and Cost Trajectories

Figures 5–20 extend the main-text case study to all eight benchmarks under GLM-5 and GPT-5.4. Each figure covers one benchmark under the same search budget. The left panel compares best-so-far benchmark objective with cumulative realized search-side cost. The middle panel shows realized step cost against iteration, and the right panel shows cumulative realized cost. Costs are divided by the search budget B, whose boundary is marked by the dotted line at 1. Objective curves are non-increasing for minimization and non-decreasing for maximization.

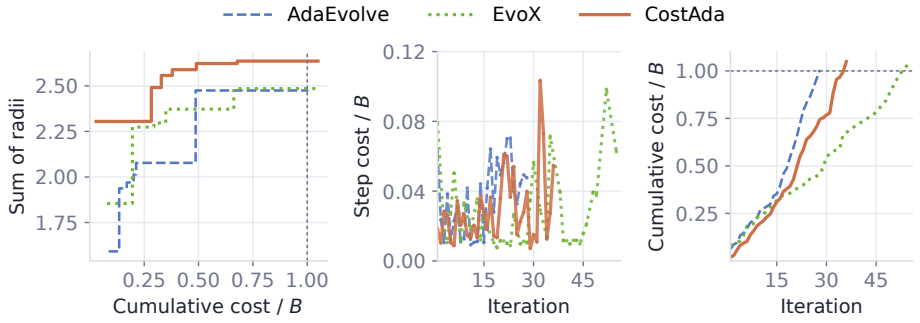


Figure 5: **Objective and cost trajectories for Circle Packing under GLM-5.**

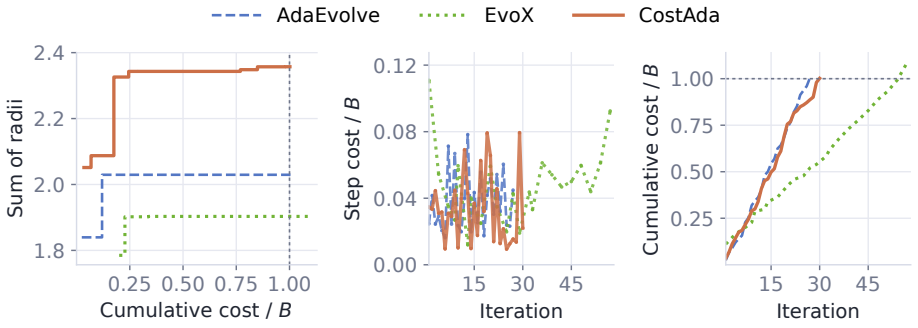


Figure 6: **Objective and cost trajectories for Circle Packing Rect under GLM-5.**

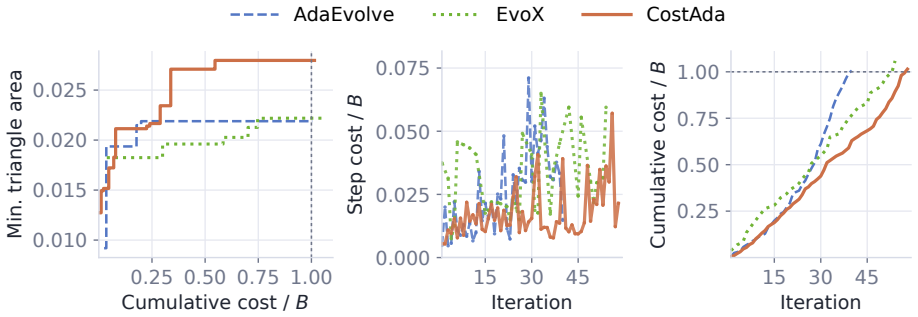


Figure 7: **Objective and cost trajectories for Heilbronn Convex under GLM-5.**

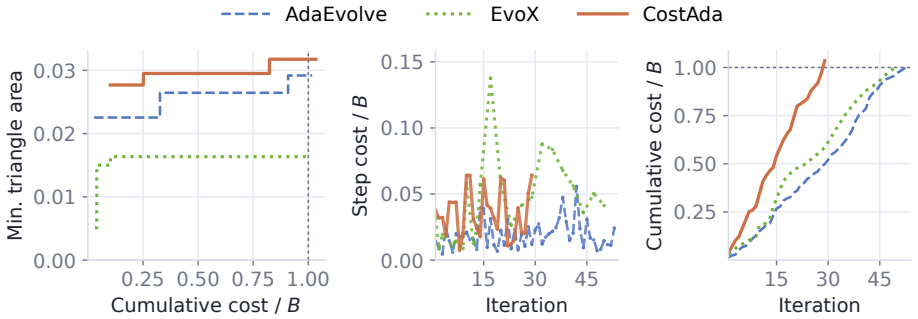


Figure 8: **Objective and cost trajectories for Heilbronn Triangle under GLM-5.**

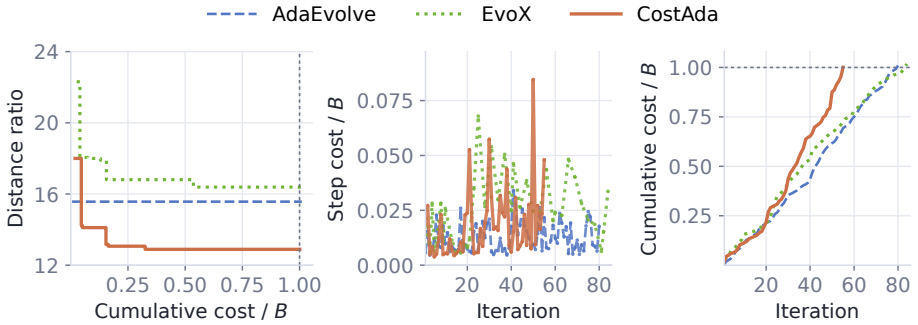


Figure 9: **Objective and cost trajectories for MinMaxDist** $(n{=}16, d{=}2)$ **under GLM-5.**

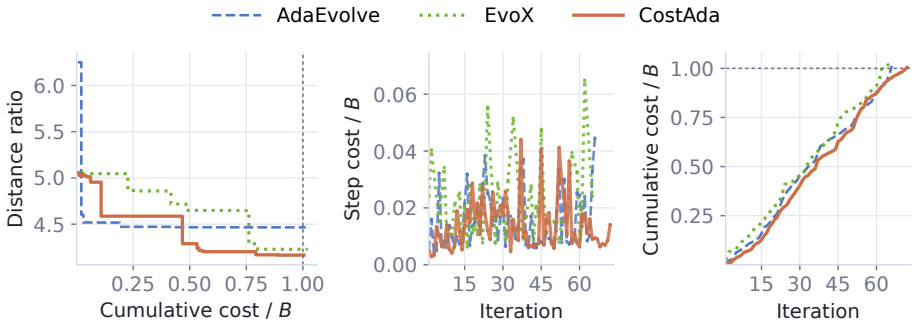


Figure 10: **Objective and cost trajectories for MinMaxDist** $(n{=}14, d{=}3)$ **under GLM-5.**

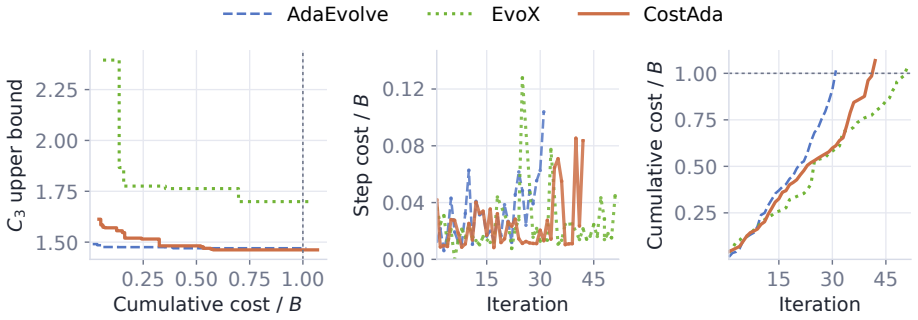


Figure 11: **Objective and cost trajectories for Third Autocorrelation under GLM-5.**

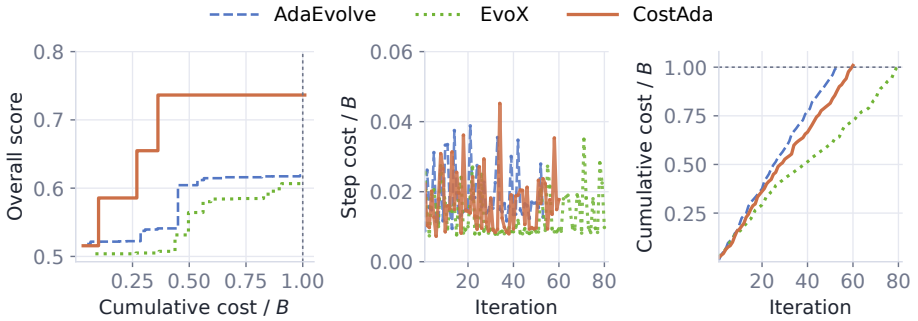


Figure 12: **Objective and cost trajectories for Signal Processing under GLM-5.**

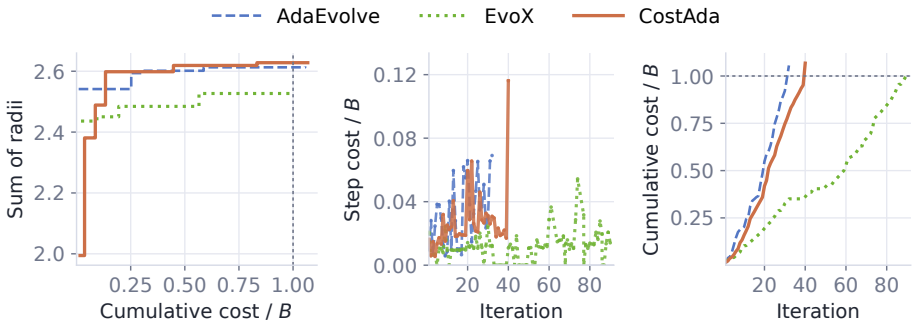


Figure 13: **Objective and cost trajectories for Circle Packing under GPT-5.4.**

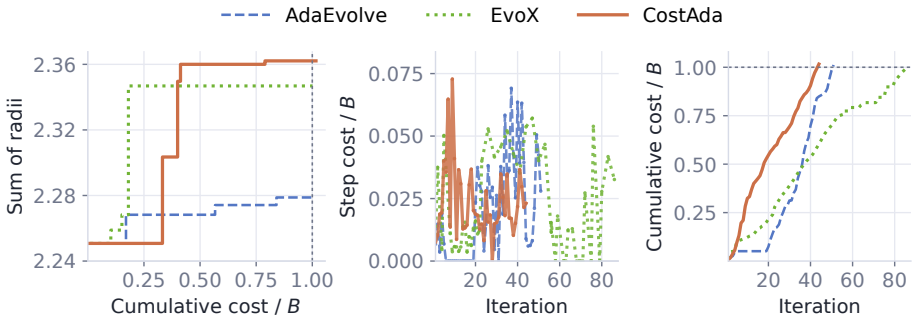


Figure 14: **Objective and cost trajectories for Circle Packing Rect under GPT-5.4.**

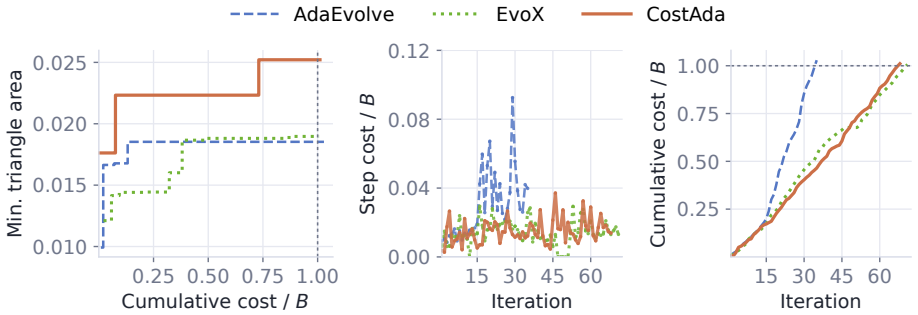


Figure 15: **Objective and cost trajectories for Heilbronn Convex under GPT-5.4.**

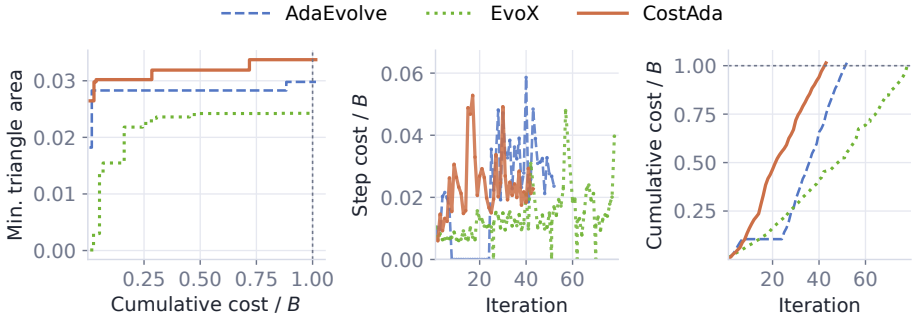


Figure 16: **Objective and cost trajectories for Heilbronn Triangle under GPT-5.4.**

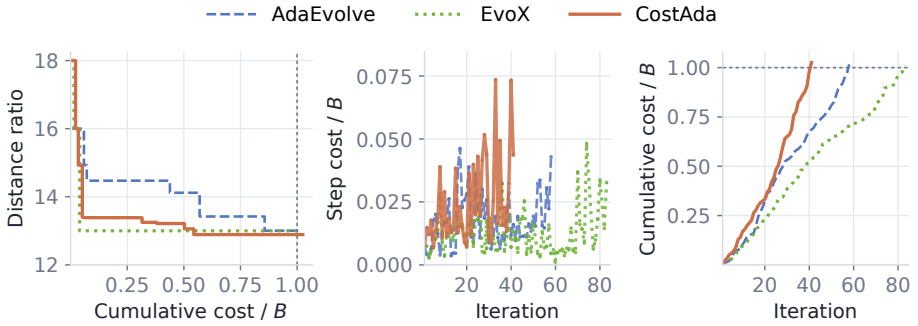


Figure 17: **Objective and cost trajectories for MinMaxDist** $(n{=}16, d{=}2)$ **under GPT-5.4.**

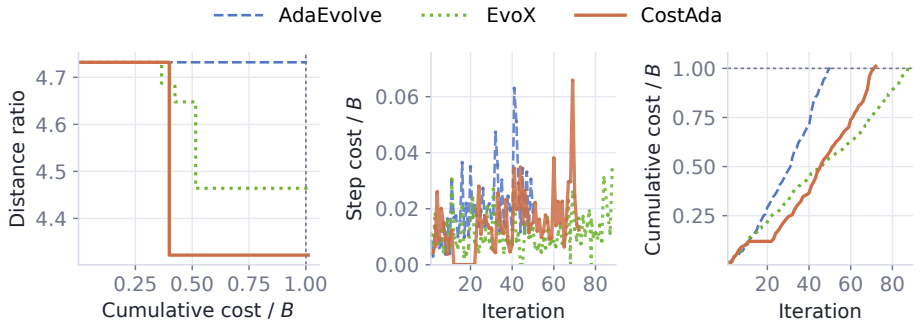


Figure 18: **Objective and cost trajectories for MinMaxDist** $(n{=}14, d{=}3)$ **under GPT-5.4.**

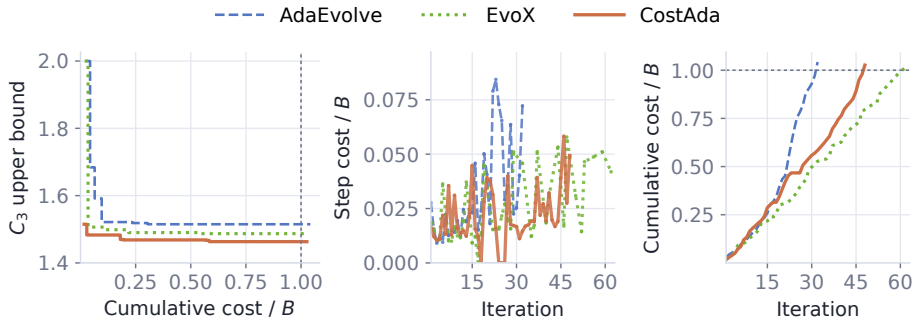


Figure 19: **Objective and cost trajectories for Third Autocorrelation under GPT-5.4.**

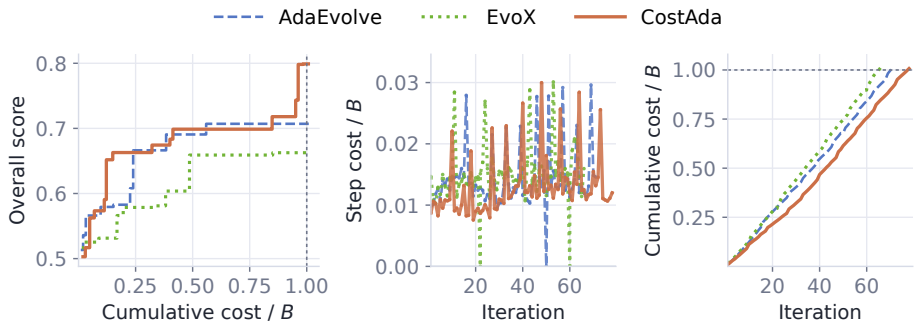


Figure 20: **Objective and cost trajectories for Signal Processing under GPT-5.4.**

K Budget Adherence

Final realized spending remains close to the nominal budgets across methods and backbones (Table 15). We report final spending as AvgCost and the relative excess as OvershootRatio. The average excess ranges from 1.1% to 2.5%, and no run exceeds the budget by more than 8.6%. Moreover, CostAda has the smallest average final spending and overshoot under GLM-5. The full-budget quality comparison in Section 5.2 therefore uses closely matched realized spending.

L Discovered Programs

Two final CostAda programs illustrate the human-reference results in Section 5.2. Figure 21 shows the configurations returned by the GLM-5 programs whose final objectives are the Circle Packing

Table 15: **Final realized spending and budget overshoot.** AVGCOST reports **Mean** $\pm$ **Std** of final cumulative search-side cost. OVERSHOOTRATIO is the relative excess over the nominal budget. For each backbone, statistics aggregate all eight benchmarks (24 runs per method). Entries are not ranked, since spending alone does not measure solution quality.

Backbone	Method	AVGCOST	Mean OVERSHOOTRATIO	Max OVERSHOOTRATIO
GLM-5 (B=1)	AdaEvolve	1.021 ± 0.019	2.1%	8.0%
	EvoX	1.025 ± 0.022	2.5%	8.6%
	CostAda	1.017 ± 0.022	1.7%	7.1%
GPT-5.4 (B=5)	AdaEvolve	5.092 ± 0.068	1.8%	5.6%
	EvoX	5.053 ± 0.064	1.1%	6.2%
	CostAda	5.103 ± 0.081	2.1%	6.8%

and MinMaxDist $(n{=}16, d{=}2)$ Best entries in Table 1. The Circle Packing program places 26 disjoint circles in the unit square with a sum of radii of 2.6360, above the reported reference of 2.6340. The MinMaxDist program returns 16 points whose maximum and minimum squared pairwise distances have a ratio of 12.89, matching the reference value. Both are deterministic Python programs, so executing them reproduces these values.

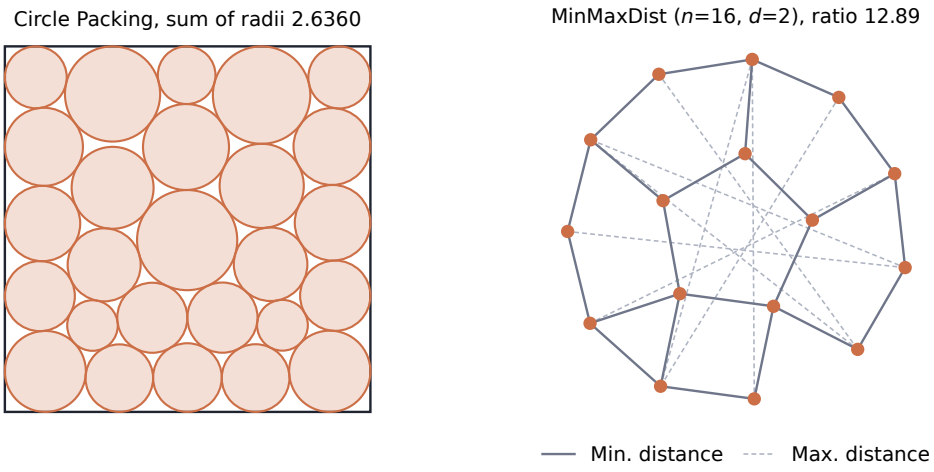


Figure 21: **Configurations returned by two final CostAda programs under GLM-5.** The left panel shows the Circle Packing solution in the unit square. In the right panel, solid lines join point pairs at the minimum distance, and dashed lines join pairs at the maximum distance.

The two programs reach these values through different uses of constrained optimization. The Circle Packing program starts from a corner-weighted layout with circles at the corners, along the edges, and in the interior. Basin hopping then perturbs this layout, and a constrained local solver (SLSQP) optimizes all centers and radii jointly. The solver's inequality constraints keep each circle inside the square and every pair of circles disjoint. The MinMaxDist program instead introduces an auxiliary variable t and maximizes it subject to every squared pairwise distance lying between t and one. This reformulation turns the ratio objective into a smooth constrained problem, which the program solves from hexagonal, grid, and circular initializations before keeping the best configuration. Figure 22 lists the core of both programs, including their constraint functions and solver calls.

Circle Packing

```
def construct_packing():
    n = 26
    best_sum, best_x = 0, None
    # [...] boundary_cons(x): circle inside the unit square; overlap_cons(x): d_ij - r_i - r_j
    for seed in [42, 123, 456]:
        np.random.seed(seed)
        centers0 = _init_config(n)
        radii0 = _compute_radii(centers0)
        x0 = np.concatenate([centers0.flatten(), radii0])

        bounds = [(0.01, 0.99)] * (2*n) + [(0.001, 0.45)] * n
        minimizer_kwargs = {
            'method': 'SLSQP',
            'bounds': bounds,
            'constraints': [{'type': 'ineq', 'fun': boundary_cons},
                            {'type': 'ineq', 'fun': overlap_cons}],
            'options': {'maxiter': 500, 'ftol': 1e-9}
        }
        result = basinhopping(
            lambda x: -np.sum(x[2*n:]), x0,
            niter=300, T=1.0, stepsize=0.08,
            minimizer_kwargs=minimizer_kwargs,
            niter_success=50
        )
        if -result.fun > best_sum:
            best_sum, best_x = -result.fun, result.x
```

MinMaxDist (n=16, d=2)

```
def min_max_dist_dim2_16() -> np.ndarray:
    n = 16
    np.random.seed(42)
    # [...] compute_ratio(pts): min/max distance ratio of a configuration
    def objective(x):
        return -x[-1]

    def min_dist_constraints(x):
        pts = x[:-1].reshape(n, 2)
        t = x[-1]
        vals = []
        for i in range(n):
            for j in range(i + 1, n):
                d_sq = (pts[i, 0] - pts[j, 0])**2 + (pts[i, 1] - pts[j, 1])**2
                vals.append(t - d_sq)
        return np.array(vals)
    # [...] max_dist_constraints(x) returns d_sq - 1.0 for every pair
    constraints = [
        {'type': 'ineq', 'fun': lambda x: -min_dist_constraints(x)},
        {'type': 'ineq', 'fun': lambda x: -max_dist_constraints(x)}
    ]
    bounds = [(0.0, 1.0)] * (2 * n) + [(0.01, 0.5)]
    # [...] init_hexagonal, init_grid, init_circular
    best_pts, best_ratio = None, -1.0
    for init_func in [init_hexagonal, init_grid, init_circular]:
        pts0 = init_func()
        # [...] t0 is 0.9 times the smallest initial squared distance
        x0 = np.concatenate([pts0.flatten(), [t0]])
        result = minimize(objective, x0, method='SLSQP', bounds=bounds,
                          constraints=constraints,
                          options={'ftol': 1e-9, 'maxiter': 1000})
        # [...] keep the configuration with the largest distance ratio
```

Figure 22: **Core of the two discovered programs.** Code lines are verbatim excerpts from the final programs, shown without their comments and docstrings. Comments beginning with `[...]` mark omitted code, such as helper functions and initializations described in the text.