BUDGET-AWARE LLM DISCOVERY VIA COST-CALIBRATED FRONTIER UTILITY

Yansen Zhang[1,*], **Yilu Liu**[1], **Tianyu Liu**[2], **Jiamin Chen**[1], **Xiaokun Zhang**[1], **Kai Xie**[3], **Xue Liu**[2], **Chen Ma**[1,†], **Yiyan Qi**[3,†]
[1]City University of Hong Kong
[2]Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
[3]International Digital Economy Academy (IDEA)
yanszhang7-c@my.cityu.edu.hk, chenma@cityu.edu.hk, qiyiyan@idea.edu.cn

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. We prove that cost-blind credit can forfeit all but a vanishing fraction of attainable quality as frontiers multiply and costs diverge. Under a fixed search-side token budget, the controller must decide which frontier is improving and whether its gain justifies the realized cost before the budget is exhausted. We introduce **CostAda**, a cost-calibrated adaptive controller built around *cost-calibrated frontier utility*. The utility values frontier progress relative to realized action cost and conditions that credit on the remaining budget. CostAda uses this signal to control local exploration intensity, frontier allocation, and budgeted tactic intervention. Cost and remaining budget therefore shape the search rather than serving only as accounting variables or a stopping rule. CostAda reaches the strongest baseline's full-budget quality with at most half the budget on twelve of sixteen benchmark–backbone pairs while achieving the strongest mean final quality on all eight benchmarks under GLM-5 and GPT-5.4.

1 INTRODUCTION

Large language models (LLMs) are increasingly used for scientific and algorithmic discovery through *inference-time search*, where candidate solutions are generated, evaluated, and refined sequentially. In mathematical optimization, algorithm design, and executable program discovery, deterministic evaluators provide repeatable quality signals for this search (Romera-Paredes et al., 2024; Novikov et al., 2025; Jiang et al., 2026; Cemri et al., 2026; Liu et al., 2026a). Once discovery becomes iterative, outcomes depend not only on model capability but also on how the controller spends its limited resources across competing search directions over time.

Existing LLM discovery systems adapt their search strategies to observed progress (Romera-Paredes et al., 2024; Novikov et al., 2025; Jiang et al., 2026; Lange et al., 2025; Cemri et al., 2026; Liu et al., 2026a), yet their controllers still credit each iteration or search action according to score progress alone. As a result, a short refinement, an invalid-code retry, and a long-context step receive comparable credit when they yield comparable score progress, although they consume different shares of a fixed token budget. Under token-based API pricing, this mismatch matters because a single long-context guidance call can cost several times as much as a short refinement.

Figure 1 illustrates why score progress alone is insufficient under a fixed token budget. Both actions improve the score by $+0.03$, but a short refinement costs one reference generation whereas a long-context step costs six. Progress-only credit values the two actions equally, while cost-calibrated credit values the expensive gain less. Yet the long-context action may justify its six-generation cost

[*]Work done as an intern at International Digital Economy Academy (IDEA).
[†]Corresponding authors: Chen Ma and Yiyan Qi.

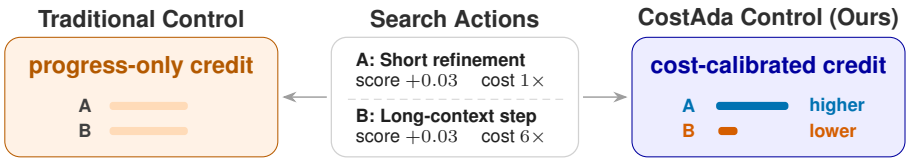

Figure 1: **Cost-calibrated credit for equal-gain, unequal-cost actions.** A short refinement and a long-context step both improve score by $+0.03$ but cost $1\times$ and $6\times$ a reference generation. Progress-only control assigns equal credit. Under a fixed budget, CostAda credits the expensive gain less.

early in a run if it opens a productive frontier, even though the same action may be too expensive near the horizon. The remaining budget therefore determines both the action's opportunity cost and the value of preserving resources for later search. Progress-only credit cannot represent this change because it measures neither return per realized cost nor how that return changes as the budget shrinks. Proposition 1 shows that, as frontier count and cost heterogeneity grow, a controller whose credit ignores realized cost can attain only a vanishing fraction of the optimal budgeted objective. Changing how progress is smoothed does not alter this separation under heterogeneous action costs.

To address this gap, we introduce **CostAda**, a cost-calibrated adaptive controller for LLM discovery under explicit *search-side token budgets*. CostAda replaces progress-only credit with *cost-calibrated frontier utility*, which combines local frontier progress, global best-so-far improvement, realized action cost, and remaining budget. Because realized cost enters the utility as a divisor, the equal gains in Figure 1 no longer receive equal credit. With ample budget, the utility retains more weight on local development because a promising frontier still has time to reach the global best. As the ledger drains, CostAda shifts credit toward global improvement and penalizes costly steps more strongly. The same long-context step can therefore justify its cost early but receive less credit near the horizon.

One principle drives all three control decisions of frontier-based search (Section 4). Local exploration intensity narrows sampling around strong candidates when a frontier returns progress efficiently and broadens the parent and context sets when utility is low. The remaining budget further suppresses expensive exploration near the horizon. Frontier allocation ranks frontiers by global gain per realized cost. Thus, a frontier that improves only against its own weak archive receives little allocation credit, while its exploration bonus shrinks with the remaining ledger. Budgeted tactic intervention purchases a higher-level guide only after local search stops converting budget into progress and the reserve can still support the generations needed to test it. CostAda treats realized action cost and remaining budget as control inputs, not as post-hoc statistics or a stopping rule.

Contributions. (1) We formulate LLM discovery under explicit search-side token budgets, where cumulative realized LLM spending induces the search horizon, and prove that cost-blind credit can forfeit the budgeted objective (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 budget-gated local exploration, cost-aware frontier allocation, and budgeted tactic intervention. (4) Our multi-budget protocol shows that CostAda reaches the strongest baseline's full-budget quality with at most half the budget in twelve of the sixteen benchmark–backbone pairs. Across GLM-5 and GPT-5.4, CostAda also records the strongest mean final quality on each of the eight benchmarks.

2 Related Work

LLM-guided evolutionary search. LLM-guided evolutionary search uses an LLM as a semantic proposal operator inside a proposal–evaluation–refinement loop (Liu et al., 2024; Ye et al., 2024; Hemberg et al., 2024). Inference-time search improves LLM outcomes beyond a single generation (Yao et al., 2023; Zhou et al., 2024). Deterministic scoring supports this search paradigm in mathematical discovery, algorithm design, equation discovery, code evolution, metaheuristic generation, and prompt optimization (Romera-Paredes et al., 2024; Novikov et al., 2025; Liu et al., 2024; Ye et al., 2024; van Stein & Bäck, 2025; Shojaee et al., 2025; Hemberg et al., 2024; Assumpção et al., 2025; Lange et al., 2025; Yang et al., 2024; Guo et al., 2024; Fernando et al., 2024; Agrawal et al., 2025). Recent systems also adapt search strategy to observed progress through momentum-style

variation, adaptive local/global control, strategy evolution, and multi-agent open-ended discovery (Jiang et al., 2026; Cemri et al., 2026; Liu et al., 2026a; Hu et al., 2025; Qu et al., 2026). Closest to our allocation question, recent work allocates compute across evolutionary search branches with bandit strategies and accelerates program evolution for efficiency (Xing et al., 2026; Yang et al., 2026b). These systems adapt where compute goes, but they still count iterations or samples as comparable units. CostAda instead ranks frontiers by progress per unit of realized cost and lets the remaining budget govern the cost penalty.

Budget-aware LLM systems. A parallel line makes budgets part of the decision process rather than only a post-hoc reporting statistic. The idea has classical roots in bandits with knapsacks, where budget-constrained selection concentrates on arms with the highest expected gain per unit cost (Badanidiyuru et al., 2013). Tool-use and multi-agent work expose remaining budget and explicit cost constraints to the agent (Liu et al., 2025; Yang et al., 2026a; Fang et al., 2026). Test-time-compute work studies token-budget-aware reasoning, adaptive token allocation, compute-optimal scaling, and tree-search policies aligned with fixed token budgets (Alomrani et al., 2025; Han et al., 2025; Li et al., 2025; Wen et al., 2025; Agarwal et al., 2025; Wang et al., 2025; Miyamoto et al., 2026). Evaluation-driven scaling for scientific discovery further shows the value of repeated evaluator-grounded generation (Ye et al., 2026). Existing budget-aware LLM methods allocate tokens within a single reasoning trajectory or among one agent's tool calls. By contrast, CostAda allocates variable-cost actions across a population of competing search frontiers.

3 Problem Formulation

We formalize this population-level allocation problem over a discrete space of executable candidates $\mathcal{P}$. A deterministic task evaluator assigns each candidate a scalar score $F : \mathcal{P} \to \mathbb{R}$, where higher is better. When an evaluator returns multiple metrics, or when the benchmark's native objective is minimized, F denotes the direction-adjusted scalar proxy exposed to the controller. At iteration t, the controller proposes a candidate $p_t \in \mathcal{P}$, observes its score $f_t = F(p_t)$, and maintains the global best-so-far score $y_t = \max_{\tau \le t} f_\tau$. No other quality feedback enters the controller.

The search state at iteration t contains K_t frontiers indexed by $k \in \{1, \dots, K_t\}$. Each frontier stores a local archive $D_t^{(k)} \subseteq \mathcal{P}$ with local best score $b_t^{(k)} = \max_{p \in D_t^{(k)}} F(p)$. At each step, the controller selects a frontier k_t, samples parent/context candidates from it, invokes the search-side LLM to generate a new candidate, evaluates it, and updates both local and global state. The frontier representation separates local progress from improvement in the global best-so-far solution.

3.1 Search Budget and Objective

Unlike iteration-limited discovery, our setting defines the effective horizon by cumulative search-side LLM cost. Let $B > 0$ be the nominal budget for a run. The realized cost of search-side LLM calls at iteration t is c_t, excluding deterministic evaluator execution (Appendix D). The controller tracks consumption through the cumulative cost $C_t = \sum_{s \le t} c_s$ and the remaining-budget ratio $\rho_t = \max(0, 1 - C_t/B) \in [0, 1]$. The remaining-budget ratio decreases from one to zero as spending approaches the nominal budget. For scale-invariant control, 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 within the benchmark family and $\epsilon_c > 0$ is a stabilizer. Control decisions depend on this normalized cost rather than the raw price.

The budgeted discovery objective is to maximize 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 controller policy that selects frontiers, sampling actions, and interventions, and τ_B is the last iteration whose cumulative cost stays within the budget. The objective is to obtain the strongest solution reachable within the budget-induced horizon, not global optimality over $\mathcal{P}$. During execution, B is a nominal rather than a strictly enforceable limit. The cost of an LLM call is observed only after it returns, so truncating a run exactly at τ_B would change the action being evaluated. Accordingly, a run completes the first iteration that reaches the boundary, $\hat{\tau}_B = \min\{t : C_t \ge B\}$, issues no further search-side calls, and is evaluated at that crossing.

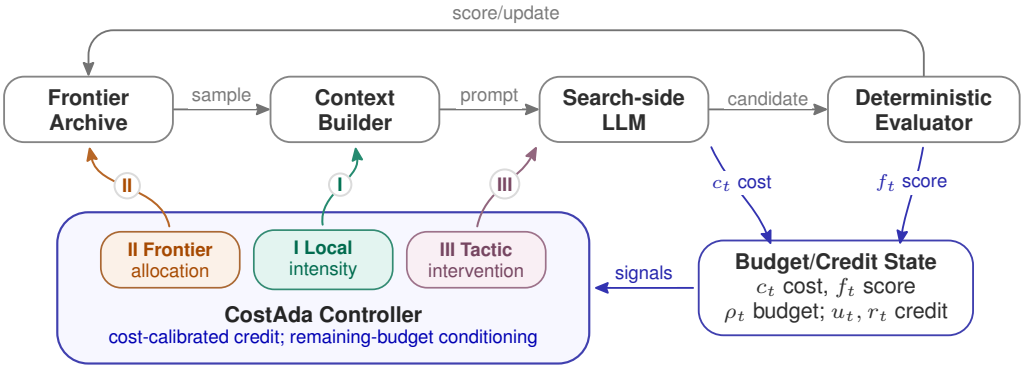

Figure 2: **The overall architecture of CostAda.** The shared discovery loop returns the candidate score and realized LLM cost to the budget/credit state. CostAda uses cost-calibrated frontier utility and remaining-budget conditioning to control local exploration intensity, frontier allocation, and budgeted tactic intervention. Numerals mark each control and its target module.

3.2 Failure of Cost-Blind Credit

Call a controller *cost-blind* if its action distribution at every iteration depends only on the history of actions and observed scores, never on realized costs. Progress-only credit rules are cost-blind regardless of how the progress signal is smoothed or combined, and there are heterogeneous-cost instances on which this class provably forfeits the budgeted objective.

Proposition 1 (Cost-blind control forfeits the budgeted objective) *For every $\kappa > 1$ and $K \geq 2$ there is a K-frontier instance with equal per-step gains, in which one frontier has per-selection search cost c and the others κc, such that as $B/c \to \infty$ every cost-blind controller attains at most $\left(\frac{1}{K} + \frac{1}{\kappa}\right)(1 + o(1))\,\mathrm{OPT}$ in expectation, where OPT is the optimal budgeted objective value on the instance, whereas ranking frontiers by gain per realized cost attains $(1 - o(1))\,\mathrm{OPT}$.*

The proof is in Appendix A. The attainable fraction vanishes as frontiers multiply and action costs diverge, mirroring discovery runs in which short refinements and long-context guidance calls share one ledger. The proposition characterizes cost-blind controllers rather than CostAda itself. 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

CostAda organizes three coupled controls around one credit principle. Throughout, *cost* is the realized search-side expenditure of an individual action, whereas *budget* is the finite ledger whose remaining balance sets that action's opportunity cost. CostAda calibrates progress credit by realized action cost and conditions subsequent decisions on the remaining budget. This coupling makes the opportunity cost of each action explicit. A gain worth buying early can be too costly near the horizon.

Figure 2 shows the overall architecture of CostAda. The upper loop passes a selected frontier through context construction, candidate generation, and deterministic evaluation. The resulting score and realized LLM cost update the lower budget/credit state, which supplies the three control signals. Frontier allocation chooses the next frontier, local exploration intensity sets its sampling breadth, and budgeted tactic intervention determines when higher-level guidance is worth its cost. Although they act at different points, the controls share one cost-calibrated credit principle. Pre-step control reads ρ_{t-1} and stored credit because the action precedes its realized cost. Post-step credit uses ρ_t only after the cost is charged. This ordering prevents post-generation information from influencing the action that produced it, while keeping it available for future decisions.

4.2 COST-CALIBRATED FRONTIER UTILITY

For the candidate produced at iteration t on frontier k_t, CostAda measures progress at two levels, a local gain and a global gain:

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

where f_t is the new candidate score, $b_{t-1}^{(k_t)}$ is the local best of the selected frontier, and y_{t-1} is the global best-so-far score. The symmetric denominator keeps the gain scale stable when a benchmark starts from zero or its scores change sign or pass near zero. Local progress identifies frontiers still developing, while only global progress justifies repeated budget allocation.

The utility weights local and global gain according to the remaining budget. Early in a run, enough budget remains for a productive frontier to challenge the global best. Near the horizon, further work on a locally improving but globally weak frontier carries a greater opportunity cost. The global weight is $\lambda_t = 1 - \rho_t$, which rises from zero to one as the ledger drains. The credit mixture shifts from local development toward global best-so-far improvement as λ_t rises. A fresh run credits local progress almost entirely, while a nearly exhausted run credits only global improvement.

Realized cost enters through the divisor $d_t = 1 + \lambda_t^c \phi_t$, where $\phi_t = \log(1 + \tilde{c}_t)$ and $\tilde{c}_t$ is the normalized step cost from Section 3. The cost weight $\lambda_t^c = \max(\lambda_t, \lambda_{\min})$ floors λ_t at a fixed positive $\lambda_{\min}$, keeping cost penalization active early in a run. The logarithm preserves cost ordering while limiting the influence of rare, very expensive calls. As λ_t^c increases, costly low-gain steps receive progressively less utility while cheap steps keep nearly full credit.

Combining remaining-budget-conditioned progress with the realized-cost divisor yields the cost-calibrated utility

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

The denominator calibrates credit by realized action cost, while λ_t and λ_t^c condition the progress mixture and cost pressure on the remaining budget. Both gains are nonnegative and $d_t \geq 1$, so the utility is nonnegative and reduces to the plain progress mixture only for a zero-cost step. CostAda smooths this utility with an exponential moving average $H_t^{(k_t)} = \alpha H_{t-1}^{(k_t)} + (1 - \alpha) u_t^{(k_t)}$, where $\alpha \in (0, 1)$ is a fixed smoothing coefficient. The exponential average limits the influence of any single step and remains unchanged for unselected frontiers, $H_t^{(k)} = H_{t-1}^{(k)}$. This statistic controls exploration intensity within a frontier, while allocation uses the reward defined below.

4.3 BUDGET-AWARE CONTROL

CostAda uses the cost-calibrated frontier utility together with the remaining ledger state to choose future actions at the local, allocation, and intervention levels.

Local exploration intensity. Because a newly opened frontier has too little evidence for its utility estimate to govern sampling, CostAda uses a short bootstrap stage before applying utility-controlled intensity. After bootstrap, the controller sets

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

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 indicates that the selected frontier is producing useful progress per cost, so the controller moves toward lower-intensity refinement around strong candidates. Low recent utility raises intensity, broadening the parent and context set before abandoning the frontier. The multiplier ρ_{t-1} suppresses expensive exploration as the remaining budget shrinks. The local sampler π_{local} is a fixed distribution over action modes conditioned on this intensity. The sampled mode $\tilde{a}_t \sim \pi_{\text{local}}(\cdot \mid I_t^{(k_t)})$ governs how broadly the parent candidate and context set are drawn from the frontier archive $D_{t-1}^{(k_t)}$ for the search-side LLM. A broader context set costs more to prompt.

Frontier allocation. The allocation layer decides where future budget should be spent. For this purpose, CostAda uses only global gain per realized cost:

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

The allocation reward measures whether frontier k_t advanced the overall best-so-far solution per realized cost. Local gain is excluded because improvement against a weak local archive does not by itself justify further global budget. CostAda maintains a smoothed allocation estimate

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

where $\gamma \in (0,1)$ is a fixed smoothing coefficient. CostAda selects the next frontier using an upper-confidence allocation rule (Auer et al., 2002):

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

where c_{ucb} is a fixed optimism scale, N_t counts frontier-selection decisions, and $n_t^{(k)}$ counts selections of frontier k. The first term exploits frontiers with recent global progress per cost. The second preserves exploration of under-sampled frontiers in proportion to the remaining budget. CostAda explores low-evidence frontiers early in a run. Late in a run, allocation relies on observed global progress per cost rather than spending scarce budget on uncertainty alone.

Budgeted tactic intervention. When local sampling and frontier allocation enter a low-yield regime, CostAda gates guide/tactic spending on three binary evidence predicates. The core criterion is $\mathcal{I}_t = A_t \wedge (P_t \vee Y_t)$. The affordability predicate A_t holds when the remaining ledger can pay for the guide/tactic call and the ordinary generations needed to test its tactics. P_t holds after a sustained absence of meaningful global progress, with patience adapting to the remaining budget and the frontier count. Y_t holds when low-yield spending accumulates before a long stagnation window forms, providing earlier evidence that local search is unproductive.

The complete decision additionally requires that no consolidation window, active tactic batch, or backoff suppression is in effect (Appendix B.4). CostAda schedules a guide/tactic only when local search no longer converts budget into useful progress and the guide will not exhaust the remaining horizon. The intervention mode follows the same cost-aware principle. Before guidance has produced incumbent-level global progress, CostAda favors breakthrough tactics. After such progress appears, CostAda favors refinement tactics that preserve the productive direction.

Guide calls stay on the same search-side ledger. Their credit is evaluated over the whole tactic cycle they induce rather than only the guide call. Successful cycles briefly consolidate search around the improving frontier. After an unproductive cycle, the controller requires fresh stagnation or low-yield evidence before another budgeted tactic intervention can proceed.

4.4 Overall Procedure

The controller starts iteration t by reading ρ_{t-1} together with each frontier's smoothed utility, allocation estimate, and selection count. Frontier allocation first chooses k_t. The selected frontier's smoothed utility determines local exploration intensity, which sets the sampling mode, parent, and context set. If $\mathcal{I}_t$ holds, the controller buys a guide/tactic instead of issuing an ordinary generation. On return, the realized cost enters the ledger and lowers ρ_t. The evaluator scores the candidate. CostAda then applies Eq. (1) to update the statistics used at the next step. The run ends at the first iteration whose cumulative cost reaches B. The controller constants are fixed within each benchmark family and are not retuned per benchmark. Appendix B gives the bootstrap action, progress predicates, guide-cycle bookkeeping, backoff rule, constants, and Algorithm 1. Appendix B.6 traces the same decisions on the run log used in the case study.

5 Experiments

5.1 Experimental Setup

Benchmarks. We evaluate on the eight-benchmark suite used by recent SkyDiscover, AdaEvolve, and EvoX studies (Liu et al., 2026b; Cemri et al., 2026; Liu et al., 2026a), spanning geometric packing,

Table 1: **Final benchmark quality at matched search budgets for GLM-5 and GPT-5.4.** Arrows indicate the preferred direction. All strongest entries at the reported precision are bolded, and tied entries are bolded jointly. Human reference values follow prior benchmark reports (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 ↑	–	0.6022 ± 0.0479	0.6545	0.6592 ± 0.0573	0.7162	**0.7054 ± 0.0124**	**0.7191**
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 ↑	–	0.7011 ± 0.0128	0.7107	0.6161 ± 0.0561	0.6808	**0.7965 ± 0.0346**	**0.8268**

extremal geometry, additive combinatorics, and executable program optimization. Benchmark definitions and score directions are given in Appendix C.

Baselines. We compare CostAda with AdaEvolve (Cemri et al., 2026), the closest progress-aware baseline, and EvoX (Liu et al., 2026a), a strategy-evolution controller from a different design family. AdaEvolve already controls local exploration, frontier allocation, and high-level intervention from progress signals. Neither baseline conditions its decisions on the remaining budget. Within each setting, all methods share the benchmark evaluator, search scaffold, backbone, prices, and budget protocol, so the comparison isolates the controller.

Backbones and budgets. We run the main comparisons with GLM-5 (Zeng et al., 2026) and GPT-5.4 (OpenAI, 2026), whose per-token prices differ by about fourfold on input and eightfold on output (OpenRouter, 2026a;b). The price gap motivates different nominal budgets and creates two distinct cost regimes. Appendix D lists the fixed OpenRouter prices and defines the shared cost ledger. Budgets are denominated in the provider-priced cost of input and output tokens rather than in raw token counts. In these dollar units, the nominal limit is B=1 for GLM-5 and B=5 for GPT-5.4. Runs stop under the crossing-completion convention of Section 3, at $\hat{\tau}_B$. We allow at most 100 search iterations under either backbone, although many runs reach the budget boundary earlier. To measure early progress, we read the same runs at budget cutoffs $q \in \{0.25, 0.5, 0.75, 1.0\}$, each at $\hat{\tau}_{qB}$. We repeat each benchmark–method setting three times independently.

Metrics. Final quality is BESTSCORE@BUDGET, the best-so-far score $y_{\hat{\tau}_B}$ at the budget crossing. At each cutoff, we report the *benchmark objective*, the headline value used in prior work (Liu et al., 2026b; Cemri et al., 2026; Liu et al., 2026a), with the units and score directions shown in Table 1. Because these objectives are not comparable across benchmarks, we record a direction-adjusted *normalized score*. We use normalized AUC to summarize the best-so-far normalized score over the full budget path and thus credit reaching a given quality level earlier (Appendix H).

5.2 FINAL QUALITY UNDER EQUAL BUDGETS

Table 1 reports final solution quality at the same nominal budget in benchmark-objective units. Human values are listed for reference only. Because their search-side token budgets are unreported, the budget-matched comparison covers the three adaptive methods.

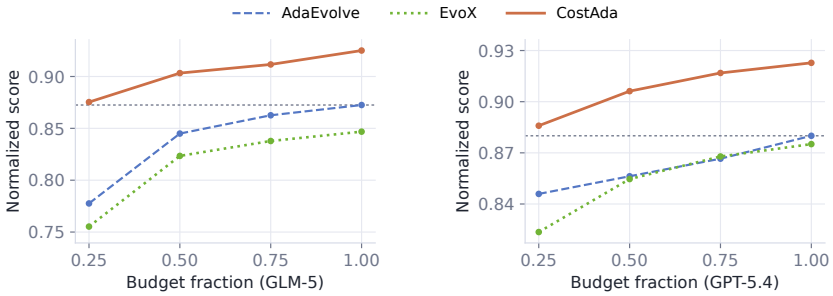

Figure 3: **Quality along the budget path.** Mean normalized score over the eight benchmarks at each cutoff, under GLM-5 (left) and GPT-5.4 (right). Points average the per-benchmark means of Appendix H. The panels use independent vertical scales. The dotted line marks the strongest baseline's value at the *full* budget, which CostAda exceeds at a quarter of that budget.

CostAda attains the strongest mean benchmark-objective value on all eight benchmarks and the strongest or tied single-run value at the reported precision in every setting. The only ties occur on MinMaxDist $(n{=}14, d{=}3)$ under GLM-5 at 4.17 and on Heilbronn Convex under GPT-5.4 at 0.0252, both with EvoX. CostAda remains strictly strongest in the corresponding means. The stronger baseline changes across benchmarks. Under GLM-5, AdaEvolve holds the better mean on four benchmarks and EvoX on the other four. Under GPT-5.4, AdaEvolve leads on three and EvoX on four, and the two tie on MinMaxDist $(n{=}16, d{=}2)$. CostAda's mean lead therefore holds against whichever baseline is stronger for each benchmark. The result spans both score directions and all four benchmark families. This breadth shows that cost calibration preserves endpoint quality under matched budgets despite substantial differences in model pricing.

Signal Processing gives the clearest separation between CostAda and the baselines. CostAda improves mean BESTSCORE@BUDGET by 7.0% under GLM-5 (0.7054 vs. EvoX 0.6592) and 13.6% under GPT-5.4 (0.7965 vs. AdaEvolve 0.7011). Under GPT-5.4, both baselines remain at 13.00 ± 0.00 on MinMaxDist $(n{=}16, d{=}2)$, whereas CostAda reaches 12.93 ± 0.06. CostAda's strongest single run reaches the human reference value of 12.89.

Because the methods share one dollar ledger, CostAda's endpoint advantage reflects spending allocation rather than a larger budget. Under GLM-5, CostAda also pairs the strongest mean with the smallest cross-run standard deviation on six of the eight benchmarks. On Circle Packing Rect under GLM-5, the two baselines report standard deviations of 0.1635 and 0.2563 against CostAda's 0.0081. Under progress-only credit, an expensive branch that yields no global improvement can continue to absorb budget because local gain is credited without regard to realized cost. CostAda instead assigns the branch less credit and allocates future budget according to global gain per realized cost.

Consistent with this allocation argument, component ablations show that removing cost calibration, remaining-budget conditioning, or intervention gating reduces quality at both budgets on three benchmarks under both backbones (Appendix G). The endpoint comparison establishes final quality, not the budget each controller needs. Section 5.3 measures that requirement.

5.3 QUALITY UNDER PARTIAL BUDGETS

Final scores do not show how quality accumulates as the budget is spent, so we read the same runs at four cutoffs. Under GLM-5, CostAda leads or ties 30 of the 32 benchmark-objective cells (Appendix E). Under GPT-5.4, CostAda leads 29 of the 32 benchmark-objective cells. At half the budget, CostAda's mean matches or exceeds the strongest baseline's full-budget mean on five benchmarks under GLM-5 and seven under GPT-5.4. In total, CostAda reaches the same quality for at most half the search-side dollar cost on twelve of sixteen benchmark–backbone pairs.

Figure 3 shows the aggregate pattern. At a quarter of the budget, CostAda averages 0.8753 under GLM-5 and 0.8859 under GPT-5.4. Both scores exceed the strongest baseline's full-budget averages of 0.8725 and 0.8800. Over the full budget path, CostAda attains the highest normalized AUC on fifteen of the sixteen benchmark–backbone pairs. CostAda averages 0.8626 against the strongest

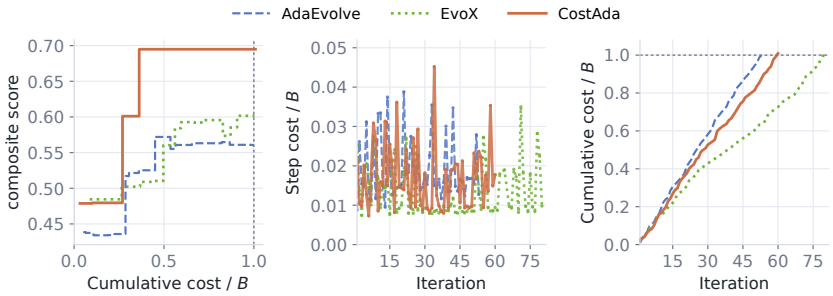

Figure 4: **Objective and cost trajectories for Signal Processing under GLM-5.** The left panel plots best-so-far benchmark objective against cumulative realized search-side cost. The middle panel shows realized step cost, and the right panel shows cumulative realized cost, both against iteration. Costs are divided by the nominal budget B, whose boundary is marked by the dotted line at 1. Curves show one run per method at the same nominal budget.

baseline's 0.8060 under GLM-5 and 0.8851 against 0.8437 under GPT-5.4. Together, the cutoff and AUC results show that CostAda reaches strong solutions early and maintains that advantage from the earliest cutoff through the full budget.

The half-budget result also holds under two robustness checks. First, we run CostAda directly at $B{=}0.5$ rather than read the midpoint of a full-budget trajectory. These independently budgeted runs beat the strongest baseline mean in all 16 cells (Appendix F). Second, spending at the crossing remains closely matched. Because runs complete the crossing iteration, all three methods exceed the nominal budget by only 1.1–2.5% on average under both backbones (Appendix J). Across both checks, the efficiency gains reflect how CostAda allocates budget along the run rather than where the search stops or how much it spends at the crossing.

5.4 Case Study

Aggregate tables do not show how each method spends its budget within a run. Figure 4 provides a single-run comparison on Signal Processing under the same nominal GLM-5 budget. The left panel compares methods by cumulative cost. In the run shown, CostAda surpasses both baselines' final levels within roughly the first third of the budget and maintains that margin to the boundary. The early lead agrees with the aggregate result in Section 5.3, where CostAda's quarter-budget mean on this benchmark (0.6210) already exceeds AdaEvolve's full-budget mean (0.6022). The middle and right panels show that realized step costs vary several-fold. Consequently, the methods reach the same budget after different numbers of iterations. Appendices I and K provide the corresponding views for every benchmark and backbone, while Appendix B.6 traces the CostAda decisions behind this curve against the logged controller state. In this run, the cost-based view reveals a budget-efficiency difference that an iteration-indexed trajectory would hide.

6 Conclusion

We presented CostAda, a cost-calibrated controller for LLM discovery under explicit search-side token budgets. Our analysis shows that cost-blind credit can be confined to a vanishing fraction of the budgeted objective on heterogeneous-cost instances. CostAda addresses this failure through cost-calibrated frontier utility and remaining-budget conditioning across local exploration, frontier allocation, and budgeted tactic intervention. With at most half the budget, CostAda reaches the strongest baseline's full-budget quality in twelve of sixteen benchmark–backbone pairs. Across GLM-5 and GPT-5.4, CostAda also achieves the strongest mean final quality on all eight benchmarks. Together, these results show that CostAda improves budget efficiency while preserving final solution quality across two distinct backbone cost regimes.

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. The complete source code and experimental scripts are available at `https://github.com/Forrest-Stone/CostAda`.

AI Use Statement

OpenAI Codex was used to polish the language and presentation of this paper.

References

Aradhye Agarwal, Ayan Sengupta, and Tanmoy Chakraborty. The art of scaling test-time compute for large language models. *arXiv preprint arXiv:2512.02008*, 2025.

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, Daniel Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, Peng Cheng, Yunzhou Wang, Pengyi Liao, Hanrui Huang, Bin Wang, Jianye Hao, and Mark Coates. Reasoning on a budget: A survey of adaptive and controllable test-time compute in llms. *arXiv preprint arXiv:2507.02076*, 2025.

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. doi: 10.1023/A:1013689704352.

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.

Zhengru Fang, Senkang Forest Hu, Zhonghao Chang, Yu Guo, Yihang Tao, Hongyao Liu, Mengzhe Ruan, Jun Huang, and Yuguang Fang. Inference-time budget control for LLM search agents. *arXiv preprint arXiv:2605.05701*, 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/`.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations*, 2024.

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.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

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. *arXiv preprint arXiv:2509.19349*, 2025.

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 Vu, 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.

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

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*, volume 169. 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 Rio. *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 in evolutionary search: From depth-breadth to multi-armed bandits. *arXiv preprint arXiv:2605.29268*, 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.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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, Caiyin Yang, Chang Su, Rahul Thapa, Rui Yang, Ruihua Liu, Zeyu Li, Chong Gao, Dachao Ding, Guangrong He, Miaolei Zhang, Lina Sun, Wenyang Wang, Yuchen Zhong, Zhuohao Shen, Di He, Jianfeng Ma, Stefano Ermon, Tongyang Li, Xiaowen Chu, James Z. Wang, and Yuzhi Xu. Evaluation-driven scaling for scientific discovery. *arXiv preprint arXiv:2604.19341*, 2026.

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, 2026.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 62138–62160, 2024.

A Proof of Proposition 1

Instance. Fix $g > 0$, $\kappa > 1$, and $K \geq 2$. Every frontier selection deterministically improves the global best score by g. The budgeted objective value is therefore g times the number of selections completed before cumulative search cost exceeds B. One frontier costs c per selection, while the other $K - 1$ frontiers cost κc per selection. The cheap frontier is placed adversarially and can be identified only from realized costs.

Optimum. Selecting the cheap frontier at every step completes $\lfloor B/c \rfloor$ selections, so $\mathrm{OPT} = g\lfloor B/c \rfloor$.

Upper bound for cost-blind controllers. Every selection returns the same gain g, so the action–score history visible to a cost-blind controller is identical under every placement of the cheap frontier. The controller's possibly random selection sequence σ therefore has the same distribution under every placement. Condition on σ. For placement j, let N_j be the number of selections completed within budget, and let $V_j(n)$ count selections of frontier j among the first n. The first n selections cost $c\,(\kappa n - (\kappa - 1)V_j(n))$, so the ledger constraint at $n = N_j$ rearranges to

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

Every selection costs at least c, so $N_{\max} = \max_j N_j \leq B/c$. Each V_j is nondecreasing, and $\sum_j V_j(N_{\max}) = N_{\max}$. Averaging the display over the K placements gives

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

The adversarial placement attains $\min_j N_j$, which is at most this average. Taking expectation over σ preserves the bound because it holds for every realization. Since the run value is $g\,N_j$, every cost-blind controller attains at most $\left(\frac{1}{K}+\frac{1}{\kappa}\right)(1+o(1))\,\mathrm{OPT}$ in expectation.

Achievability with cost-aware credit. 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. 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. 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. Observing the stopping event does not help because that signal arrives only after the budget is spent. 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. Our setting additionally includes frontier development, nonstationarity from growing archives, and intervention actions. The proposition isolates cost-blindness from these difficulties.

B CostAda Controller Details

B.1 Controller State and Overall Loop

We specify the experimental loop, local-sampling bootstrap, frontier-allocation bookkeeping, and evidence rules needed to reproduce CostAda. The controller maintains the evidence state

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

where each component supports budgeted tactic intervention while sharing the same search-side ledger as local exploration and frontier allocation. Table 2 summarizes these state variables.

Table 2: **CostAda evidence state.** The state variables are used by the budgeted tactic intervention layer and are updated from realized score and cost evidence.

State	Role
ν_t	Counts consecutive steps without meaningful global progress to detect global stagnation.
L_t	Accumulates realized cost over consecutive low-utility and low-allocation-reward steps.
Z_t	Records whether a guide/tactic cycle has produced incumbent-level global progress.
o_t	Stores a temporary intervention-mode correction after a failed guide cycle.
w_t	Stores the remaining consolidation window after successful incumbent evidence.
κ_t	Stores the frontier selected for consolidation while $w_t > 0$.
$\mathcal{G}_t$	Stores the active guidance-cycle state, including its baseline score and remaining tactic batch.

B.2 Local Exploration Control

After a frontier has been selected, the local controller chooses the action used to sample parent and context candidates from that frontier. The short bootstrap stage below handles low-evidence frontiers before the intensity rule is applied.

Algorithm 1: CostAda control loop

Input : Initial frontiers $\{D_0^{(k)}\}_{k=1}^{K_0}$, evaluator F, budget B, reference cost $\bar{c}$, controller constants Θ (Table 3), iteration cap $T_{\max}$

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

Initialize global best y_0, local bests $\{b_0^{(k)}\}$, 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**

`// Remaining-budget conditioning: form the pre-step budget ratio`

`// Allocation level: cost-aware frontier allocation`

Form the pre-step budget state $\rho_{t-1} \leftarrow \max(0, 1 - C_{t-1}/B)$ and select frontier k_t using the allocation statistic and optimism bonus

`// Local level: budget-gated exploration intensity`

Choose local action a_t using limited-evidence bootstrap or the budget-gated intensity rule; draw parent/context programs $p_t^{\text{par}}, \mathcal{C}_t$ from $D_{t-1}^{(k_t)}$

Generate candidate p_t using the selected context and any scheduled guide/tactic information; evaluate p_t and obtain scalar score f_t

`// Charge realized search-side token cost to the shared ledger`

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 frontier utility: the core credit signal`

Update local and global best scores and the best-so-far candidate $p^\star$; compute $\delta_t^{(k_t)}$, g_t, λ_t, λ_t^c, d_t, utility $u_t^{(k_t)}$, and allocation reward $r_t^{(k_t)}$

Update $H_t^{(k_t)}$, $R_t^{(k_t)}$, visit counts, and intervention evidence; carry forward states for unselected frontiers

Update active guidance-cycle credit if a guided step improves over its cycle baseline

Apply guide backoff if the active guidance cycle is exhausted without sufficient gain

`// Intervention level: budgeted tactic intervention`

if $\mathcal{I}_t = 1$ (Section B.4) **then**

Schedule a budgeted guide/tactic intervention with mode m_t

When the guide is generated, open a guidance cycle with the current best score as baseline

end

`// Budget stop: halt at the first crossing of the token budget`

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

break

end

end

Return $p^\star$

B.2.1 Bootstrap Action

Let $N_{t-1} = \sum_k n_{t-1}^{(k)}$ be the total number of frontier selections before step t. The local action is

$$a_t \leftarrow \begin{cases} \text{exploration}, & N_{t-1} = 0, \\ \text{balanced}, & N_{t-1} > 0 \wedge n_{t-1}^{(k_t)} = 0, \\ \tilde{a}_t, & \text{otherwise}, \end{cases}$$

where $\tilde{a}_t \sim \pi_{\text{local}}(\cdot \mid I_t^{(k_t)})$ is sampled from the budget-gated intensity rule of Section 4.

B.3 Frontier Allocation Control

The allocation state records whether a frontier advances the global best solution per realized cost. Local-only improvement is excluded because progress inside a weak frontier does not by itself justify more global budget. CostAda updates the allocation state only for the selected frontier. For every

unselected frontier, $R_t^{(k)} = R_{t-1}^{(k)}$. The controller uses the ordinary frontier allocation rule unless a consolidation window is active. During consolidation, CostAda selects the frontier κ_t defined below.

B.4 Budgeted Tactic Intervention

The intervention decision $\mathcal{I}_t$ uses the evidence rules below to determine whether a guide/tactic call is worth purchasing, which mode to use, and when later calls should be suppressed. Successful and failed guide cycles also feed back into frontier allocation and local sampling. Appendix B.6 traces both trigger paths and both modes in a recorded run log.

B.4.1 Stagnation and Low-Yield Evidence

The global stagnation counter is updated after observing the realized global gain:

$$\nu_t = \begin{cases} 0, & g_t > \epsilon_g, \\ \nu_{t-1} + 1, & \text{otherwise.} \end{cases}$$

The stagnation criterion compares this counter with a budget-adaptive patience requirement,

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

where ν_0 is the base patience scale. The term $2K_t$ requires evidence across roughly two active-frontier cycles and is determined by the number of active frontiers rather than tuned per benchmark.

CostAda also accumulates the realized cost of consecutive steps whose cost-calibrated utility and allocation reward are both negligible:

$$\ell_t = \mathbb{1}\left[u_t^{(k_t)} \leq \epsilon_g \wedge r_t^{(k_t)} \leq \epsilon_g\right], \qquad L_t = \begin{cases} L_{t-1} + c_t, & \ell_t = 1, \\ 0, & \ell_t = 0, \end{cases}$$

where ℓ_t flags a low-yield step and L_t accumulates realized cost over consecutive low-yield steps. The low-yield criterion is

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

The criterion permits an earlier intervention while at least half of the budget remains. A trigger requires one frontier cycle without global improvement and accumulated low-yield cost at least as large as an estimated guide call. By that point, the run has already spent intervention-scale budget without useful return. The stagnation path P_t remains available later in the run.

B.4.2 Progress Predicates

CostAda uses the same realized global gain with two evidence levels:

$$\text{Prog}_t = [g_t > \epsilon_g], \qquad \text{Inc}_t = [g_t > \epsilon_{\text{inc}}], \quad \epsilon_{\text{inc}} = 10\epsilon_g.$$

The first predicate resets stagnation and low-yield evidence. The second predicate is used for successful-incumbent evidence, consolidation, and the refinement prior.

B.4.3 Guidance-Cycle Credit and Mode Selection

When a budgeted guide/tactic call is generated, CostAda opens a guidance cycle $\mathcal{G}$. The cycle baseline $y_{\text{base}}^{\mathcal{G}}$ is the global best before the generated tactic batch is consumed. Let $G_t = 1$ indicate that an active guide/tactic batch remains. Let $\text{guided}_t = 1$ indicate that step t uses a tactic from the active guidance cycle. The cycle includes every candidate generated with the new tactics, including a same-step guided candidate when applicable and subsequent guided steps before the batch is exhausted. The cycle-level normalized gain is

$$g_t^{\mathcal{G}} = \max\left(\frac{y_t - y_{\text{base}}^{\mathcal{G}}}{\max(|y_t|, |y_{\text{base}}^{\mathcal{G}}|, 1)}, 0\right).$$

Successful-incumbent evidence is produced when a budgeted guide cycle yields incumbent-level global progress before the tactic batch is exhausted:

$$\zeta_t = \mathbb{1}\left[g_t^{\mathcal{G}} > \epsilon_{\text{inc}} \wedge \text{guided}_t = 1\right].$$

The persistent incumbent state is $Z_t = \max(Z_{t-1}, \zeta_t)$. The default intervention mode is

$$m_t^0 = \begin{cases} \text{breakthrough}, & Z_t = 0, \\ \text{refinement}, & Z_t = 1. \end{cases}$$

When an intervention is scheduled, the active mode also accounts for the temporary mode-credit correction:

$$m_t = \begin{cases} o_{t-1}, & o_{t-1} \neq \emptyset, \\ m_t^0, & \text{otherwise}. \end{cases}$$

B.4.4 CONSOLIDATION

After successful-incumbent evidence is observed ($\zeta_t = 1$), CostAda consolidates around the improving frontier for one active-frontier cycle:

$$w_t = \begin{cases} K_t, & \zeta_t = 1, \\ \max(w_{t-1} - 1, 0), & \zeta_t = 0, \end{cases} \qquad \kappa_t = \begin{cases} k_t, & \zeta_t = 1, \\ \kappa_{t-1}, & \zeta_t = 0, \end{cases}$$

where w_t is the remaining consolidation window and κ_t is the frontier held for consolidation. When $w_t > 0$, frontier allocation returns κ_t, guide calls are suppressed, and the local controller uses exploitation-oriented sampling for the consolidation window.

B.4.5 GUIDE BACKOFF AND MODE-CREDIT CORRECTION

A guide cycle exhausted without incumbent-level global progress indicates poor return on its cost. Let $\text{exhaust}_t^{\mathcal{G}} = 1$ indicate that the active guidance cycle is exhausted at step t. The backoff signal is

$$\text{Backoff}_t = \mathbb{1}\left[\text{exhaust}_t^{\mathcal{G}} = 1 \wedge \max_{s \in \mathcal{G}} \zeta_s = 0\right].$$

When $\text{Backoff}_t = 1$, CostAda resets ν_t and L_t before the next decision. A subsequent guide therefore requires fresh stagnation or low-yield evidence. The intervention-mode correction is updated as

$$o_t = \begin{cases} \emptyset, & \zeta_t = 1, \\ \text{breakthrough}, & \text{Backoff}_t = 1 \wedge Z_{t-1} = 1 \wedge m_t = \text{refinement}, \\ \emptyset, & \text{Backoff}_t = 1 \wedge o_{t-1} \neq \emptyset, \\ o_{t-1}, & \text{otherwise}. \end{cases}$$

A successful incumbent therefore gives refinement a prior. A failed refinement guide permits one breakthrough probe, while a failed alternative probe returns the controller to the incumbent prior.

B.4.6 AFFORDABILITY AND FINAL INTERVENTION DECISION

The reserve criterion is

$$A_t = [B - C_t \geq \hat{c}_{\text{guide}} + \bar{c}],$$

which requires enough remaining budget for the guide/tactic call and a small ordinary-generation reserve. The reserve allows the generated tactics to be attempted. The full intervention decision output is $(\mathcal{I}_t, m_t)$, where

$$\mathcal{I}_t = [w_t = 0] \wedge [G_t = 0] \wedge [\text{Backoff}_t = 0] \wedge A_t \wedge (P_t \vee Y_t).$$

The mode m_t is ignored when $\mathcal{I}_t = 0$ and attached to the scheduled guide/tactic call when $\mathcal{I}_t = 1$.

B.5 CONTROLLER CONSTANTS

Controller constants are fixed once per benchmark family and are not retuned per benchmark. CostAda's cost-calibration rule uses ledger-derived quantities: realized step cost, normalized cost, cumulative spending, and remaining budget ratio. Table 3 summarizes the constants that appear in the controller definitions above and in Algorithm 1.

Table 3: **CostAda controller constants.** Constants are fixed within each benchmark family and are not retuned per benchmark. Symbols match the definitions above.

Symbol	Role	Used for
$I_{\min}, I_{\max}$	Local intensity bounds	Lower and upper bounds for parent/context sampling breadth.
α	Utility smoothing rate	Exponential moving average for local utility $H_t^{(k)}$.
γ	Allocation smoothing rate	Exponential moving average for allocation reward $R_t^{(k)}$.
$\lambda_{\min}$	Cost-penalty floor	Early-run lower bound on cost penalization in d_t.
c_{ucb}	Optimism scale	Remaining-budget-gated frontier exploration bonus.
ϵ_g	Meaningful-progress threshold	Stagnation reset, low-yield evidence, and Prog_t.
ϵ_{inc}	Incumbent-level threshold	Successful guide evidence, consolidation, and refinement prior, with $\epsilon_{\text{inc}} = 10\epsilon_g$.
ν_0	Base patience scale	Remaining-budget-dependent stagnation requirement $\nu_{\text{req}}(\rho_t, K_t)$.
$\epsilon_c,\ \epsilon_H$	Numerical stabilizers	Stabilize cost normalization and the local intensity denominator.
$\hat{c}_{\text{guide}}$	Guide-cost estimate	Low-yield trigger and affordability reserve for guide/tactic calls.

B.6 Controller Trace from a Run Log

Table 4 reports the controller state over iterations 6–20 of the CostAda run plotted in Figure 4. The nominal search-side budget is $B = 1$, and the reference cost is $\bar{c} = 0.01$. The reported values come directly from the run log. The trace shows how cost-calibrated utility and remaining-budget conditioning govern local exploration intensity, frontier allocation, and budgeted tactic intervention as the budget state changes. The window covers three complete guide cycles.

Table 4: **Controller trace over iterations 6–20 of one run on Signal Processing under GLM-5.** Recorded quantities are the selected frontier, local sampling mode with intensity in parentheses, realized step cost, post-step remaining-budget ratio, cost-calibrated utility of Eq. (1), and allocation reward. Guide/tactic calls are charged to the same ledger and included in the step cost of the iteration that issues them. A blank Event cell marks an iteration with no controller event.

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

Realized action costs. Step costs inside this window range from $0.0073 to $0.0361, and the seven guide/tactic calls issued over the full run cost between $0.0074 and $0.0179 each. The observed cost range corresponds to the middle panel of Figure 4. The realized costs enter cost-calibrated utility through the divisor d_t. The costliest step here is about five times the cheapest.

Local utility and allocation reward. Iteration 10 provides a clear example of the distinction between local and global progress. The step produces a normalized local gain of 0.0692 against its frontier archive but does not improve the global best. Equation (1) assigns a positive cost-calibrated utility of 0.050, while the allocation reward is zero because it uses only global gain per realized cost. The local controller therefore keeps the frontier available for refinement, but the allocation layer records no evidence for sending it more global budget. Iterations 7, 15, and 20 show the complementary case. Their gains are global, so utility and allocation reward coincide at 0.056, 0.052, and 0.062. Only these steps add evidence for sending the frontier more budget.

Local control under a shrinking budget. Local intensity falls as the ledger drains, from 0.46 at $\rho_t = 0.92$ to 0.37 by the end of the displayed window. Later in the run, intensity reaches 0.26 at $\rho_t = 0.32$ and 0.15 at exhaustion. Consolidation creates sharper changes within this gradual decline. At iterations 8–9 and 16–17, intensity drops to 0.15 as CostAda switches to exploitation inside a consolidation window. Sampling returns to exploration when the window closes. The local rule therefore combines gradual budget-driven narrowing with an evidence-driven exploitation response.

Intervention triggers and modes. At iteration 6, the global stagnation counter reaches its requirement of four. CostAda schedules a breakthrough guide because no earlier guide has produced incumbent-level progress. 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. The two guides rely on different evidence. Iteration 12 follows a stagnation trigger, whereas iteration 17 follows a low-yield trigger at $\rho_t = 0.70$. By iteration 17, the run has accumulated \$0.0277 of low-yield spending, roughly two to three times the cost of one guide call. The stagnation counter is only two against a requirement of four, so the low-yield path intervenes before a longer stagnation window forms.

Increasing cost pressure. The required stagnation count remains four throughout the displayed window. The requirement rises to five when ρ_t reaches 0.32 and to six by $\rho_t = 0.16$ because patience scales with $1 - \rho_t$. Over the same span, the cost divisor d_t grows from about 1.14 early in the run to 1.72, 2.01, and 2.48 as λ_t^c increases. The same realized cost is therefore penalized more heavily late in the run. Near the horizon, a guide requires more evidence and an expensive step earns less credit than it would near the start. The run ends at the budget crossing with cumulative search-side cost of \$1.0090 against nominal $B = 1$, an overshoot of 0.90%.

Summary. The trace shows how return per realized cost and remaining budget jointly govern all three control levels. Local sampling narrows as the ledger drains, guidance changes from breakthrough to refinement after incumbent-level progress, and the intervention threshold rises near the horizon. All three behaviors follow the same cost-calibrated credit principle under different budget states. Section 5.4 plots the same run against its baselines.

C 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 Rio, 2010; Sharma, 2025; Shenoi, 2005). Table 5 lists each benchmark, its optimization direction, and the objective used in the result tables.

D Experimental Accounting Details

D.1 Search-Side Cost

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

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

Table 5: **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.
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 pairwise distance, following the reference squared-ratio convention (Novikov et al., 2025).
MinMaxDist (n=14, d=3)	↓	Place 14 points in three dimensions and minimize the ratio between maximum and minimum pairwise distance, again using the reference squared-ratio convention.
Third Autocorrelation	↓	Construct witness functions that improve the upper bound on the third autocorrelation constant C_3 in additive combinatorics.
Signal Processing	↑	Synthesize a causal online filtering program for noisy nonstationary time series, balancing fidelity, smoothness, lag, and false trend changes.

where μ_j is the model used by call j, $T_{\text{in}}^{(j)}$ and $T_{\text{out}}^{(j)}$ are provider-reported billable tokens, and $p_{\text{in}}^{(\mu_j)}$, $p_{\text{out}}^{(\mu_j)}$ are the corresponding per-token prices. We use fixed OpenRouter prices for all methods. 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). Search-side calls include ordinary candidate generation, recovery from failed generations, and guide/tactic generation. Deterministic evaluator execution is excluded. Budgets in Section 5 are denominated in this quantity. 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, shared across benchmarks, and used only for controller normalization rather than the evaluation ledger.

D.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 within each benchmark family, and the same pricing assumptions apply to every method in that family.

E Per-Benchmark Budget-Cutoff Results

The cutoff results below support the budget-path analysis in Section 5.3. Table 6 reports the GLM-5 ladder at $B \in \{0.25, 0.5, 0.75, 1\}$, while Table 7 gives the corresponding fractions of $B = 5$ under GPT-5.4. We evaluate all three methods at the same four cutoffs. We omit the $B = 0$ initialization because it precedes the first generated candidate and may not have a benchmark-objective value.

The early-budget advantage in the main-text GLM-5 analysis also appears under GPT-5.4. In benchmark-objective units, CostAda leads 29 of the 32 cutoff cells and all eight benchmarks from $0.75B$ onward. The three non-leading cells occur at $0.25B$ and $0.5B$ on Circle Packing Rect and at $0.25B$ on MinMaxDist (n=16, d=2). With half of the budget, CostAda's mean matches or exceeds the strongest baseline's full-budget mean on seven of the eight benchmarks.

F Small-Budget Reruns versus Cutoff Evaluation

Table 8 compares two views of early-budget behavior under GLM-5. Within each benchmark block, the first three rows read nominal B=1 trajectories at the first completed iteration reaching each absolute budget. CostAda$_{B=0.5}$ instead reports three independent runs configured with nominal

Table 6: **Benchmark objectives across GLM-5 budget cutoffs.** The nominal budget is $B = 1$, so the columns correspond to cutoff costs 0.25, 0.5, 0.75, and 1.0 in ledger dollars. Entries report **Mean** $\pm$ **Std** at the first completed iteration reaching each cutoff, using the units and score directions of Table 1. Arrows indicate the preferred direction. Best entries within each benchmark and cutoff are bolded. Ties at the reported precision are jointly bolded.

Benchmark	**Method**	B=0.25	B=0.5	B=0.75	B=1
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	**2.5535 ± 0.0585**	**2.6016 ± 0.0335**	**2.6252 ± 0.0094**	**2.6306 ± 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	**2.3478 ± 0.0081**	**2.3478 ± 0.0081**	**2.3478 ± 0.0081**	**2.3525 ± 0.0081**
Heilbronn Convex ↑	AdaEvolve	**0.0215 ± 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	**0.0215 ± 0.0003**	**0.0250 ± 0.0030**	**0.0253 ± 0.0033**	**0.0263 ± 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	**0.0296 ± 0.0017**	**0.0296 ± 0.0017**	**0.0296 ± 0.0017**	**0.0314 ± 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	**13.62 ± 1.02**	**13.57 ± 1.07**	**13.15 ± 0.36**	**13.11 ± 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	**4.31 ± 0.24**	**4.19 ± 0.04**	**4.18 ± 0.02**	**4.17 ± 0.00**
Third Autocorrelation ↓	AdaEvolve	**1.4711 ± 0.0040**	**1.4694 ± 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	**1.4679 ± 0.0054**	**1.4658 ± 0.0037**
Signal Processing ↑	AdaEvolve	0.5330 ± 0.0436	0.5775 ± 0.0704	0.5759 ± 0.0731	0.6022 ± 0.0479
	EvoX	0.4649 ± 0.0328	0.6454 ± 0.0789	0.6571 ± 0.0604	0.6592 ± 0.0573
	CostAda	**0.6210 ± 0.0860**	**0.6540 ± 0.0890**	**0.6717 ± 0.0609**	**0.7054 ± 0.0124**

budget B=0.5 and evaluated at the same absolute budgets. These runs can adapt their search to the smaller horizon, whereas a cutoff only truncates a trajectory planned for B=1.

The independently budgeted CostAda$_{B=0.5}$ runs outperform the strongest baseline mean in all 16 benchmark–budget cells. The reruns also match or improve on the corresponding cutoff values from nominal B=1 trajectories in 13 cells. The rerun results confirm that the early-budget advantage in Section 5.3 persists when CostAda plans directly for the smaller horizon.

G Component Ablations

We study Circle Packing, Heilbronn Triangle, and Signal Processing under both backbones, covering packing, geometric point configuration, and executable program synthesis. The study compares Full CostAda with three variants. *No cost calibration* sets the cost denominator to one while retaining the remaining-budget and intervention rules. *No remaining budget* fixes the credit mixture and removes the remaining-budget factors from local intensity and frontier optimism, while leaving intervention gating unchanged. *No intervention gating* triggers guide/tactic intervention from stagnation evidence without the affordability and low-yield gates, while retaining guide backoff, consolidation, and the single-active-batch constraint. Together, the variants isolate cost calibration, remaining-budget conditioning, and intervention gating.

We perform three independent runs per variant, backbone, and benchmark. The three runs support all metrics reported for that setting, while the benchmark, evaluator, backbone, nominal budget, and stopping rule remain fixed across variants.

For each run, BestScore@$0.5B$ and BestScore@B report the benchmark objective of the best candidate available at the first completed iteration that reaches the cutoff, with candidates ranked by normalized score. We compute normalized AUC by integrating the stepwise best-so-far normalized

Table 7: **Benchmark objectives across GPT-5.4 budget cutoffs.** The nominal budget is $B = 5$, so the columns correspond to cutoff costs 1.25, 2.5, 3.75, and 5.0. Entries report **Mean** $\pm$ **Std** at the first completed iteration reaching each cutoff. Arrows indicate the preferred direction. Best entries within each benchmark and cutoff are bolded. Ties at the reported precision are jointly bolded.

Benchmark	**Method**	$0.25B$	$0.5B$	$0.75B$	$1B$
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	0.5998 ± 0.0071	0.6400 ± 0.0886	0.6604 ± 0.0400	0.7011 ± 0.0128
	EvoX	0.5156 ± 0.0441	0.5808 ± 0.0835	0.6159 ± 0.0570	0.6161 ± 0.0561
	CostAda	**0.6994 ± 0.0971**	**0.7681 ± 0.0434**	**0.7681 ± 0.0434**	**0.7965 ± 0.0346**

score over $[0, B]$, charging each action before its score takes effect. All standard deviations are sample standard deviations over the three runs. Tables 9 and 10 report the component ablations under GLM-5 at B=1 and GPT-5.4 at B=5.

Full CostAda gives the strongest displayed mean for every benchmark–metric pair under both backbones. On GLM-5 Signal Processing, removing cost calibration lowers BESTSCORE@$0.5B$, BESTSCORE@B, and normalized AUC by 4.93%, 3.89%, and 3.45%, respectively. Removing either remaining-budget conditioning or intervention gating also lowers all displayed GLM-5 means. Under GPT-5.4, the full controller leads every reported column. The consistent ordering supports the contribution of all three components within the settings examined here.

H 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 stack 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. Reference values follow prior benchmark reports (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. The fixed normalization preserves method rankings within each benchmark while placing scores from different benchmarks on a common scale.

Tables 12 and 13 report best-so-far normalized scores at the same cutoffs as the benchmark-objective tables. Table 11 complements these cutoff tables with per-benchmark normalized AUC at the nominal budget, and its *Average* row gives the per-backbone values summarized in Section 5.3. Cutoff scores

Table 8: **Independent GLM-5 small-budget runs versus cutoffs.** The first three rows of each benchmark block evaluate nominal $B{=}1$ trajectories at absolute costs 0.25 and 0.5. CostAda$_{B=0.5}$ denotes runs configured with the smaller nominal budget. Entries are **Mean** $\pm$ **Std** in the objective units of Table 1. Arrows show direction, and boldface marks the best value per benchmark and cost, including ties.

Benchmark	**Method**	$B{=}0.25$	$B{=}0.5$
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	0.5330 ± 0.0436	0.5775 ± 0.0704
	EvoX	0.4649 ± 0.0328	0.6454 ± 0.0789
	CostAda	**0.6210 ± 0.0860**	0.6540 ± 0.0890
	CostAda$_{B=0.5}$	0.6207 ± 0.0339	**0.7099 ± 0.0426**

Table 9: **Component ablations under GLM-5.** Entries report **Mean** $\pm$ **Std**. Benchmark objectives are used for the two cutoff columns, and normalized AUC is higher-better. Arrows indicate the preferred objective direction, and the strongest entry in each benchmark–metric column is bolded.

Benchmark	**Variant**	**BestScore@**$0.5B$	**BestScore@**B	**Norm. AUC** ↑
Circle Packing ↑	Full CostAda	**2.6195 ± 0.0128**	**2.6281 ± 0.0067**	**0.9517 ± 0.0095**
	No cost calibration	2.5389 ± 0.1153	2.5738 ± 0.0554	0.9053 ± 0.0376
	No remaining budget	2.6074 ± 0.0125	2.6172 ± 0.0057	0.9161 ± 0.0316
	No intervention gating	2.3594 ± 0.2251	2.5291 ± 0.0949	0.8478 ± 0.0642
Heilbronn Triangle ↑	Full CostAda	**0.0306 ± 0.0009**	**0.0313 ± 0.0004**	**0.7831 ± 0.0486**
	No cost calibration	0.0167 ± 0.0032	0.0228 ± 0.0040	0.4654 ± 0.0614
	No remaining budget	0.0212 ± 0.0048	0.0251 ± 0.0049	0.4073 ± 0.0426
	No intervention gating	0.0206 ± 0.0085	0.0244 ± 0.0035	0.4494 ± 0.1622
Signal Processing ↑	Full CostAda	**0.6682 ± 0.0645**	**0.7121 ± 0.0627**	**0.6583 ± 0.0309**
	No cost calibration	0.6352 ± 0.0200	0.6844 ± 0.0601	0.6355 ± 0.0200
	No remaining budget	0.6101 ± 0.0503	0.6258 ± 0.0443	0.6146 ± 0.0134
	No intervention gating	0.5986 ± 0.0142	0.6938 ± 0.0691	0.6336 ± 0.0214

Table 10: **Component ablations under GPT-5.4.** Entries report **Mean** $\pm$ **Std**. Benchmark objectives are used for the two cutoff columns, and normalized AUC is higher-better. Arrows indicate the preferred objective direction, and the strongest entry in each benchmark–metric column is bolded.

Benchmark	Variant	BESTSCORE@$0.5B$	BESTSCORE@B	Norm. AUC ↑
Circle Packing ↑	Full CostAda	**2.6197 ± 0.0080**	**2.6212 ± 0.0073**	**0.9825 ± 0.0050**
	No cost calibration	2.5963 ± 0.0293	2.6047 ± 0.0364	0.9648 ± 0.0276
	No remaining budget	2.5898 ± 0.0421	2.5942 ± 0.0458	0.9673 ± 0.0079
	No intervention gating	2.5183 ± 0.0455	2.5223 ± 0.0386	0.9402 ± 0.0246
Heilbronn Triangle ↑	Full CostAda	**0.0309 ± 0.0010**	**0.0326 ± 0.0008**	**0.7951 ± 0.0253**
	No cost calibration	0.0252 ± 0.0021	0.0274 ± 0.0018	0.6748 ± 0.0459
	No remaining budget	0.0290 ± 0.0033	0.0291 ± 0.0033	0.7494 ± 0.0615
	No intervention gating	0.0266 ± 0.0009	0.0268 ± 0.0007	0.6949 ± 0.0259
Signal Processing ↑	Full CostAda	**0.7838 ± 0.0870**	**0.8161 ± 0.0908**	**0.7019 ± 0.0480**
	No cost calibration	0.5893 ± 0.0216	0.6521 ± 0.0685	0.6405 ± 0.0150
	No remaining budget	0.6906 ± 0.0956	0.6734 ± 0.0559	0.6557 ± 0.0192
	No intervention gating	0.6396 ± 0.0583	0.7208 ± 0.0856	0.6645 ± 0.0032

are taken from the first completed iteration that reaches each budget level. The common scale makes cells comparable across benchmarks and methods, with higher values preferred after normalization.

Table 11: **Normalized best-so-far AUC over the full budget.** Entries give **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 within each benchmark and backbone are bolded. Ties at the reported precision are jointly bolded. The *Average* row reports the mean of 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**

Under GLM-5, CostAda is best or tied in 30 of the 32 cutoff cells. The two exceptions are Third Autocorrelation at the first two cutoffs, matching the benchmark-objective analysis in Section 5.3. Under GPT-5.4, CostAda is best or tied in 29 of the 32 cutoff cells, consistent with the benchmark-objective comparison in Appendix E. The ordering therefore does not depend on the reporting scale.

Table 12: **Best-so-far normalized scores across GLM-5 budget cutoffs.** The nominal budget is $B{=}1$, with the same budget fractions as Table 6. Best entries within each benchmark and cutoff are bolded. Ties at the reported precision are jointly bolded.

Benchmark	**Method**	$B{=}0.25$	$B{=}0.5$	$B{=}0.75$	$B{=}1$
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	$\mathbf{0.9691 \pm 0.0222}$	$\mathbf{0.9873 \pm 0.0127}$	$\mathbf{0.9963 \pm 0.0036}$	$\mathbf{0.9983 \pm 0.0021}$
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	$\mathbf{0.9924 \pm 0.0034}$	$\mathbf{0.9924 \pm 0.0034}$	$\mathbf{0.9924 \pm 0.0034}$	$\mathbf{0.9944 \pm 0.0034}$
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	$\mathbf{0.6957 \pm 0.0104}$	$\mathbf{0.8096 \pm 0.0974}$	$\mathbf{0.8190 \pm 0.1078}$	$\mathbf{0.8496 \pm 0.0575}$
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	$\mathbf{0.8098 \pm 0.0452}$	$\mathbf{0.8098 \pm 0.0452}$	$\mathbf{0.8098 \pm 0.0452}$	$\mathbf{0.8595 \pm 0.0085}$
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	$\mathbf{0.9495 \pm 0.0681}$	$\mathbf{0.9540 \pm 0.0721}$	$\mathbf{0.9805 \pm 0.0264}$	$\mathbf{0.9838 \pm 0.0212}$
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	$\mathbf{0.9685 \pm 0.0522}$	$\mathbf{0.9936 \pm 0.0094}$	$\mathbf{0.9964 \pm 0.0045}$	$\mathbf{0.9994 \pm 0.0006}$
Third Autocorrelation	AdaEvolve	$\mathbf{0.9895 \pm 0.0027}$	$\mathbf{0.9907 \pm 0.0011}$	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	$\mathbf{0.9916 \pm 0.0036}$	$\mathbf{0.9931 \pm 0.0025}$
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	$\mathbf{0.6377 \pm 0.0805}$	$\mathbf{0.6924 \pm 0.0536}$	$\mathbf{0.7069 \pm 0.0318}$	$\mathbf{0.7222 \pm 0.0127}$

Table 13: **Best-so-far normalized scores across GPT-5.4 budget cutoffs.** The nominal budget is B=5, with the same budget fractions as Table 7. Best entries within each benchmark and cutoff are bolded. Ties at the reported precision are jointly bolded.

Benchmark	Method	$0.25B$	$0.5B$	$0.75B$	$1B$
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**

I Objective and Cost Trajectories Across Benchmarks

Figures 5–20 extend the main-text case study to all eight benchmarks under GLM-5 and GPT-5.4. The figures use actual iteration logs, with one run per method and one benchmark per figure under the same nominal 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 nominal budget B, whose boundary is marked by the dotted line at 1.

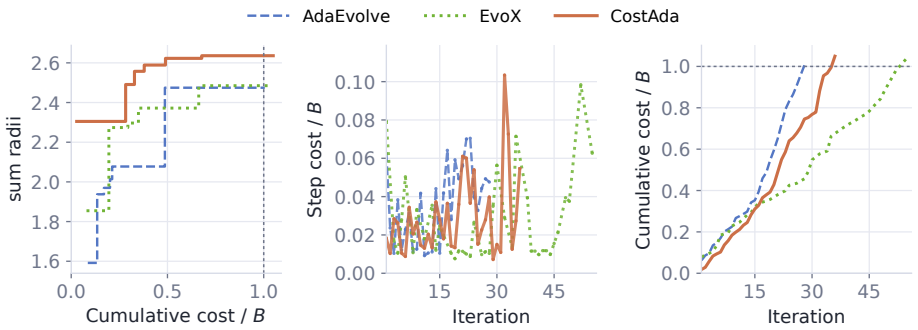

Figure 5: **Objective and cost trajectories for Circle Packing under GLM-5.**

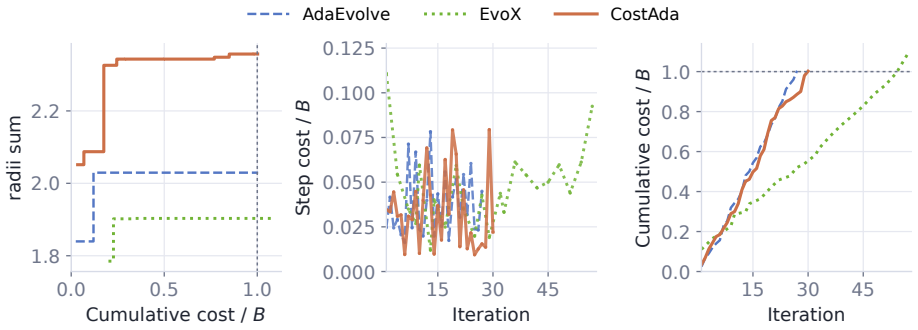

Figure 6: **Objective and cost trajectories for Circle Packing Rect under GLM-5.**

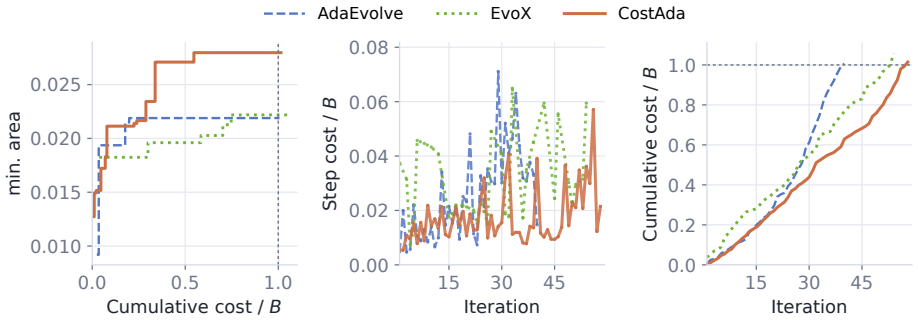

Figure 7: **Objective and cost trajectories for Heilbronn Convex under GLM-5.**

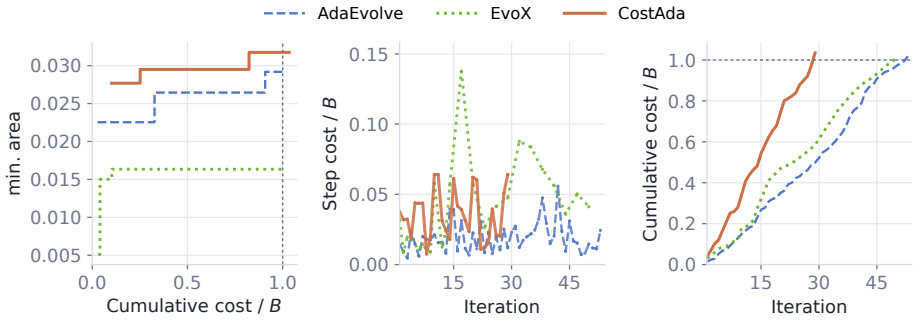

Figure 8: **Objective and cost trajectories for Heilbronn Triangle under GLM-5.**

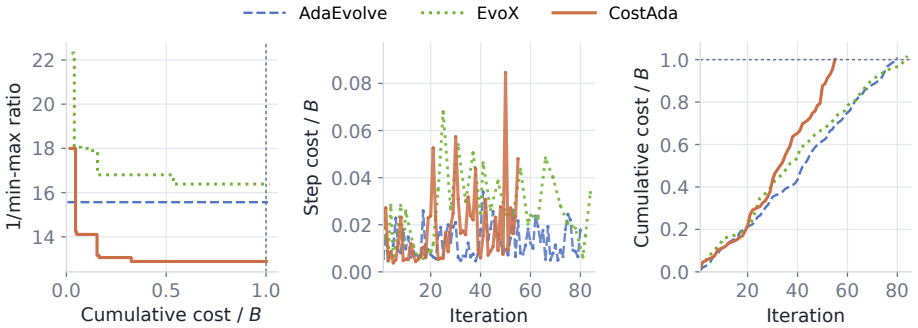

Figure 9: **Objective and cost trajectories for MinMaxDist** $(n{=}16, d{=}2)$ **under GLM-5.**

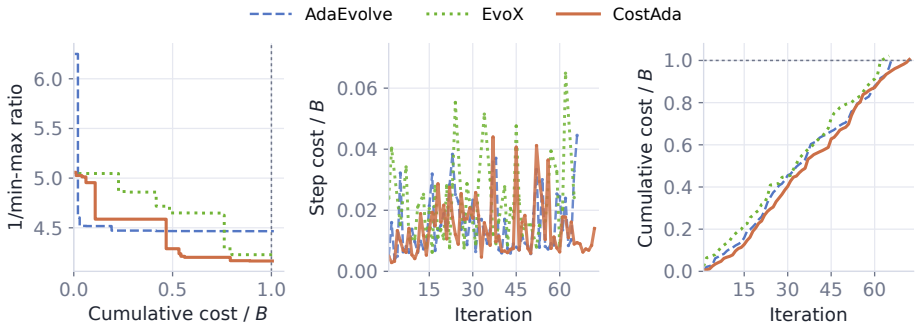

Figure 10: **Objective and cost trajectories for MinMaxDist** $(n{=}14, d{=}3)$ **under GLM-5.**

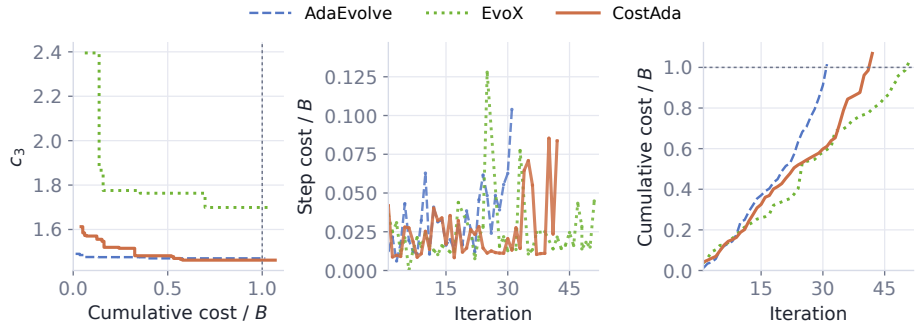

Figure 11: **Objective and cost trajectories for Third Autocorrelation under GLM-5.**

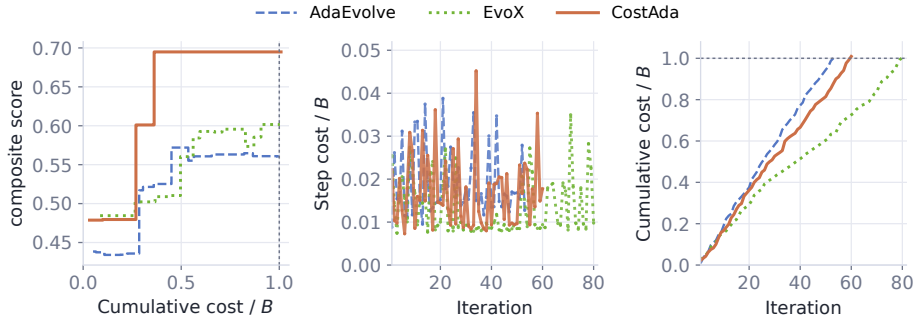

Figure 12: **Objective and cost trajectories for Signal Processing under GLM-5.**

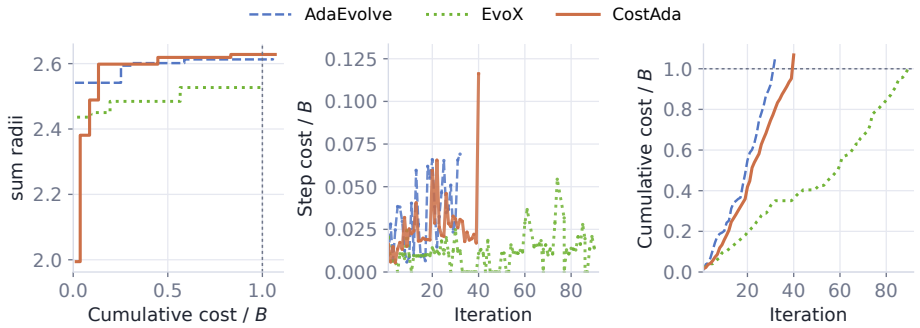

Figure 13: **Objective and cost trajectories for Circle Packing under GPT-5.4.**

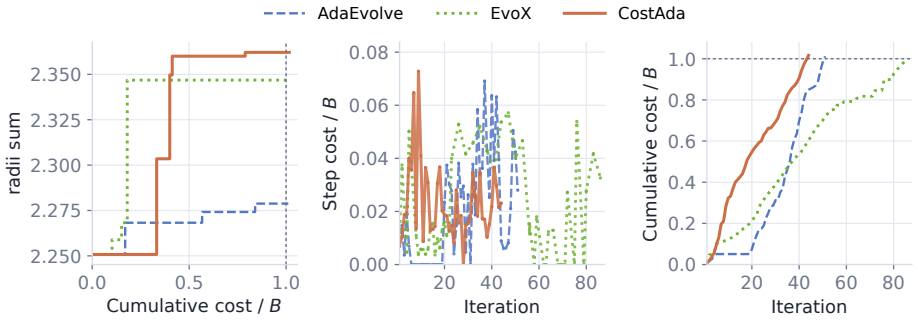

Figure 14: **Objective and cost trajectories for Circle Packing Rect under GPT-5.4.**

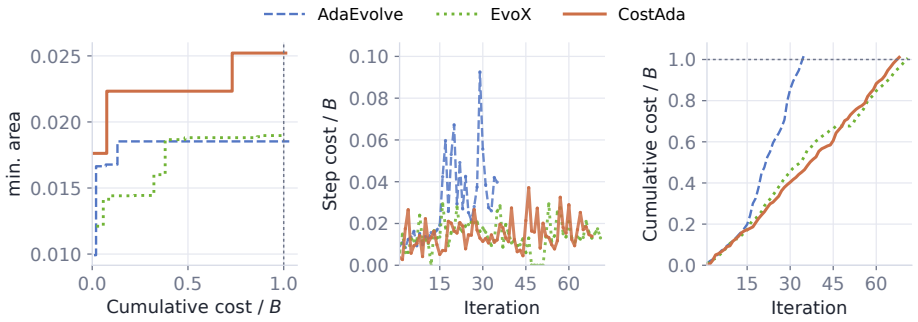

Figure 15: **Objective and cost trajectories for Heilbronn Convex under GPT-5.4.**

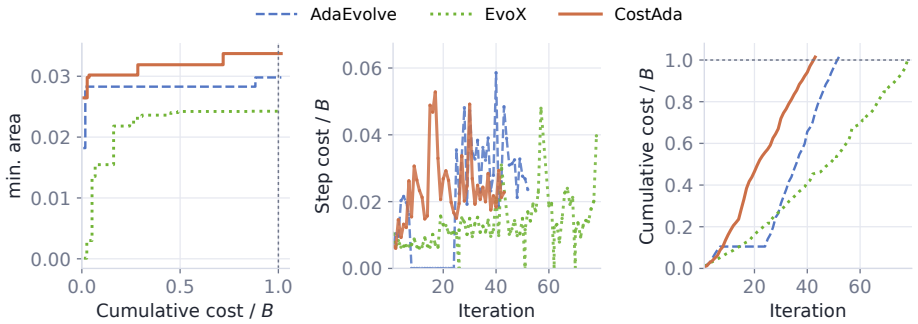

Figure 16: **Objective and cost trajectories for Heilbronn Triangle under GPT-5.4.**

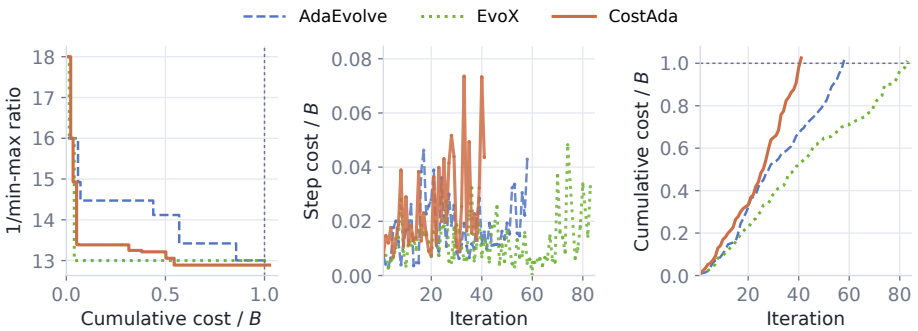

Figure 17: **Objective and cost trajectories for MinMaxDist** $(n{=}16, d{=}2)$ **under GPT-5.4.**

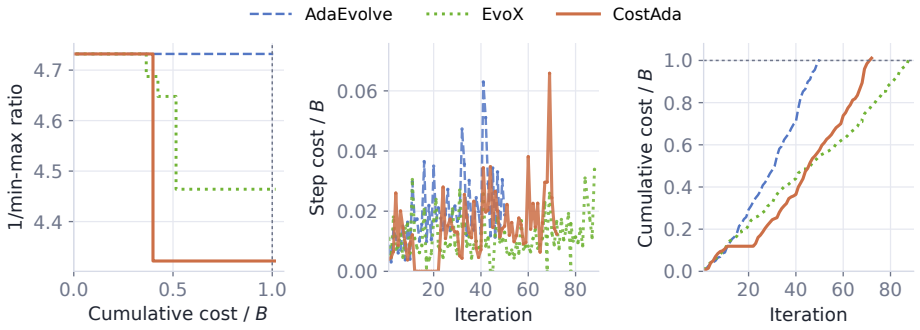

Figure 18: **Objective and cost trajectories for MinMaxDist** $(n{=}14, d{=}3)$ **under GPT-5.4.**

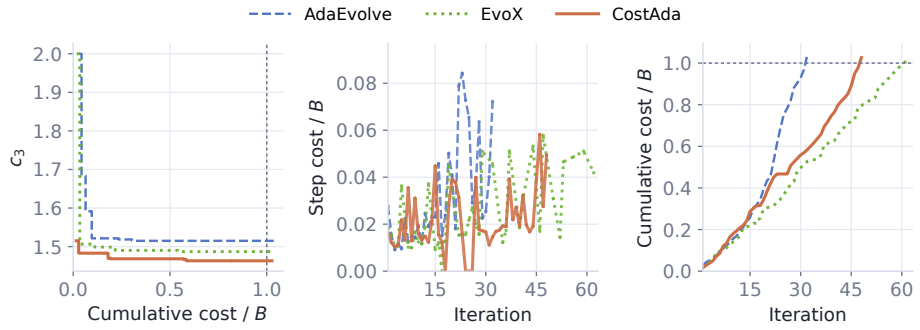

Figure 19: **Objective and cost trajectories for Third Autocorrelation under GPT-5.4.**

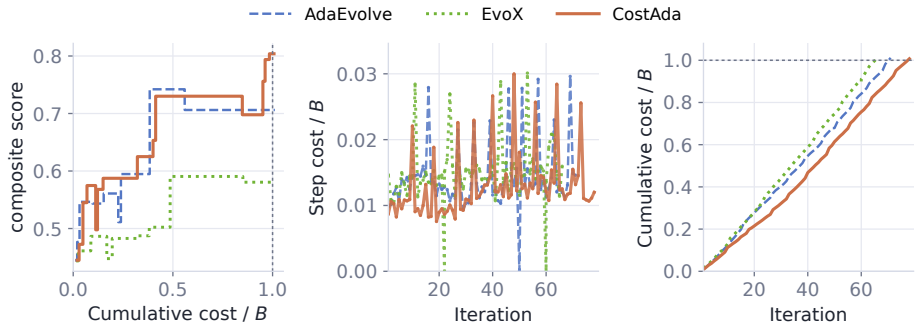

Figure 20: **Objective and cost trajectories for Signal Processing under GPT-5.4.**

J Budget Adherence Diagnostics

The budget protocol evaluates each run at the first completed iteration whose cumulative search-side cost reaches the nominal budget. Realized spending therefore includes the tail of the crossing iteration. A binary out-of-budget indicator is always one at this point and carries no information. We instead report the realized crossing cost AvgCost and the relative excess OvershootRatio. Table 14 gives both statistics for each backbone and method. All three methods cross within 1.1–2.5% of the nominal budget on average, and no run exceeds it by more than 8.6%. Under GLM-5, CostAda has the smallest average crossing cost and overshoot. The closely matched realized spending rules out unequal cost at the measurement point as an explanation for the quality differences in Section 5.3. The residual excess reflects the tail of one crossing iteration.

Table 14: **Realized spending at the nominal-budget crossing.** AVGCOST reports **Mean ± Std** of the realized cumulative search-side cost at the first completed iteration reaching the nominal budget. OVERSHOOTRATIO is the relative excess over that budget. GLM-5 and GPT-5.4 statistics each aggregate all eight benchmarks (24 runs per method). The table is a validity diagnostic for the budget-matched comparison rather than a performance ranking, so no entry is marked as best.

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%

K MARGINAL STEP-EFFICIENCY DIAGNOSTICS

Figures 21 and 22 relate realized search-side spending to best-so-far normalized-score improvements. Unlike the cumulative trajectories, these diagnostics ask whether a method spends budget on steps that change the incumbent. A lower share indicates less spending without improvement.

Across the single runs shown under GLM-5, CostAda spends the smallest share of normalized search cost on steps without a best-so-far improvement. The share is 0.84, compared with 0.87 for EvoX and 0.88 for AdaEvolve. CostAda also attains the highest total gain per unit of normalized budget at 0.63, compared with 0.54 for both baselines. Under GPT-5.4, CostAda again has the smallest no-improvement cost share at 0.88, compared with 0.91 for EvoX and 0.89 for AdaEvolve. CostAda's total gain per unit of normalized budget is 0.53, compared with 0.54 for EvoX and 0.48 for AdaEvolve. Across both backbones, the no-improvement cost share provides the consistent separation among the three controllers.

The plotted values diagnose individual runs rather than provide run-averaged estimates. The cutoff and normalized-AUC tables carry the aggregate budget-efficiency claim. In both figures, the left panel reports the share of realized search cost spent on steps without a best-so-far normalized-score improvement, where lower is better. The right panel reports total best-so-far normalized-score gain per unit of nominal budget, where higher is better. Because the panels compare point estimates rather than bar lengths, their labeled axes are zoomed to the range of the three method means to make the small between-method differences directly visible.

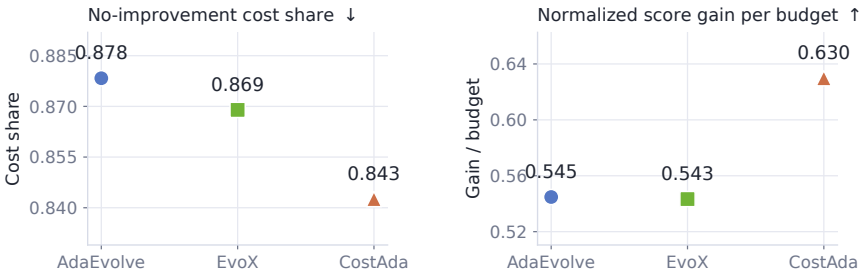

Figure 21: **Step-level budget efficiency under GLM-5.** Method means over eight benchmarks, with exact values annotated, from one run per benchmark at nominal budget B=1.

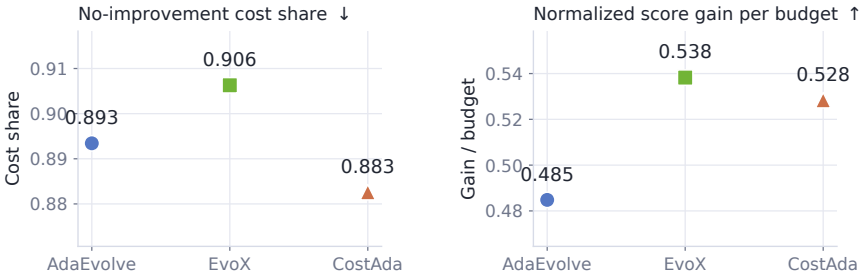

Figure 22: **Step-level budget efficiency under GPT-5.4.** Method means over eight benchmarks, with exact values annotated, from one run per benchmark at nominal budget $B{=}5$.