FEEDBACK IS NOT A FREE LUNCH: THE EXPLORATION COST OF AUXILIARY INFORMATION IN AGENTIC ALGORITHM DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents discover algorithms by writing programs, testing them, and revising them based on the results. A test gives a score and can also report truthful details about a program's behavior. We call these extra details auxiliary feedback. They may suggest useful changes, but we find that even truthful feedback is not a free lunch. In a matched study of 15 algorithmic and engineering tasks, adding auxiliary feedback raises the best score found across repeated runs on five tasks, lowers it on eight, and leaves two unchanged. To understand these results, we examine the programs agents try in a data-center control task. For each of five models, the best run with sustained auxiliary feedback earns a lower score and tries fewer new ways of using information to choose control actions than the best run without the extra feedback. We call this the exploration cost of feedback. Diagnostics can help an agent improve a promising idea, but repeatedly following them can leave fewer opportunities to try other ideas. We bridge agentic discovery with auxiliary-guided Bayesian optimization, isolating how guidance strength turns useful feedback into restricted exploration. The signal helps when its influence is moderate, but causes search to miss a better region when its influence is too strong. Removing strong guidance later restores exploration. In four tasks where sustained auxiliary feedback lowered the best score, renewed-exploration runs find solutions exceeding the strongest baseline references. Together, these results suggest that feedback should help agents improve promising ideas while leaving room to explore alternatives. Code is available at `https://anonymous.4open.science/r/feedback-discovery-E31B/`.

1 INTRODUCTION

LLM agents increasingly search for better algorithms by writing programs, evaluating them, and using the results to decide what to try next (Tsoukalas et al., 2026; Novikov et al., 2025; Chen et al., 2026a; Yuksekgonul et al., 2026). Evaluations report both how well a candidate performs and, often, how it behaves. A data-center evaluator, for example, can describe how a controller schedules workloads, adjusts cooling, and uses a battery under different grid conditions. Such diagnostics suggest directions for the next revision.

We study truthful details added to the usual evaluation response, which we call *auxiliary feedback*. Similar diagnostics appear in kernel optimization, where hardware measurements help identify bottlenecks, in robotic reward design, where reports describe individual reward components, and in equation discovery, where error measurements describe the contributions of different terms (Zhang et al., 2025; Ma et al., 2024; Saveliev et al., 2026). A score tells the agent how well a program performed; diagnostics help it decide what to change. Yet improving individual revisions need not improve the best solution found over a complete run. With a finite search budget, pursuing one promising direction leaves fewer opportunities to explore others.

Prior work has examined several ways agents use feedback. Reflexion uses verbal feedback from earlier attempts to guide later decisions (Shinn et al., 2023). CausalGame studies how agents reason from the outcomes of experiments (Chen et al., 2026b). CUDAnalyst varies the feedback shown at fixed points in program evolution to examine its effect on planning decisions (Chong et al., 2026).

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

A theoretical account of how auxiliary feedback redirects exploration over a complete agentic discovery run, and when this helps or hurts the final outcome, remains missing. Addressing this gap requires connecting feedback-driven decisions to the search histories they create.

We first study the effect of auxiliary feedback on 15 algorithmic and engineering tasks. For each task, we compare the usual evaluation response with the same response plus auxiliary diagnostics. The objective, evaluator, agent setup, initial program, and per-run budget remain fixed between conditions. We repeat each condition and compare the best valid score found across its available runs.

Feedback is not a free lunch. Across 15 tasks, auxiliary feedback raises the best-found score on five, lowers it on eight, and leaves two unchanged.

These heterogeneous outcomes raise a deeper question: how does auxiliary feedback change the allocation of search effort?

We examine this question in data-center control, where the agent writes rules for scheduling workloads, cooling servers, and charging or discharging a battery. Auxiliary feedback reports these actions under low and high grid carbon intensity. Code revisions distinguish refinement of existing rules, such as tuning a carbon-intensity threshold for battery discharge, from exploration of new dependencies on temperature or the time remaining in an episode. Across regular runs, agents receiving auxiliary feedback devote more search to refining carbon-intensity rules and less to exploring new links between information and actions. For each of five models, the highest-scoring regular run with auxiliary feedback explores fewer such links and reaches a lower score than the highest-scoring baseline run. These observations suggest an ***exploration cost of feedback***. Diagnostics can make a promising direction easier to improve, while repeated attention to that direction leaves fewer opportunities to develop alternatives.

To investigate this mechanism, we bridge agentic algorithm discovery and classical black-box optimization through auxiliary-guided Bayesian optimization (BO). Both select candidates from an evolving evaluation history: guidance shapes the next choice and, through that choice, the evidence available for future search. This shared structure provides a tractable model of how feedback reallocates exploration. Our BO setting contains a broad, accessible region and a narrower region with a better solution. Holding the objective and truthful auxiliary measurement fixed, we vary the strength of guidance. Moderate guidance improves search, whereas strong guidance concentrates evaluations in the inferior region and leaves the better one undiscovered. Interrupting strong guidance restores exploration while retaining the accumulated objective history.

We then examine renewed exploration in agentic search as the marginal gains from familiar refinements diminish. Across four tasks where sustained auxiliary feedback lowered the best-found score, renewed-exploration runs discover solutions exceeding the strongest baseline references. In data-center control, new time- and temperature-dependent rules create opportunities to reuse earlier carbon-intensity tuning experience within a richer policy. Exploration and exploitation thus build on each other: new strategies make prior refinement experience productive again. The resulting controller reaches 33.7400, exceeding the best data-center score reported in Frontier-Eng by 3.5527 points (Appendix G).

Our study makes three contributions.

1. We show that truthful auxiliary feedback can improve or impair best-found solutions through matched comparisons of complete discovery runs across 15 tasks.
2. We identify an exploration cost in data-center traces and bridge agentic algorithm discovery with auxiliary-guided BO, providing a controlled model that isolates how reliance on feedback shapes search.
3. We examine renewed exploration as a simple, initial probe of the proposed mechanism. Across four tasks, the resulting trajectories illustrate how new strategies can build on earlier feedback-guided experience, motivating more systematic methods for balancing exploitation and exploration.

The value of feedback thus depends on the search it encourages: effective discovery requires both exploiting useful diagnostics and exploring beyond the directions they suggest.

2 Related Work

Agentic discovery. Systems such as FunSearch, Evolution of Heuristics, and AlphaEvolve combine candidate generation, executable evaluation, and reuse of promising solutions (Romera-Paredes et al., 2024; Liu et al., 2024; Novikov et al., 2025). AVO extends this loop with autonomous coding agents (Chen et al., 2026a), while TTT-Discover updates model parameters during search (Yuksekgonul et al., 2026). These systems use evaluation to shape subsequent attempts. We study how adding truthful diagnostics to evaluation feedback affects discovery under a fixed objective.

Feedback and its effects. Feedback can include evaluator measurements, model-generated critiques, and reflections on previous trials. CudaForge, Eureka, and influence-guided symbolic regression expose detailed diagnostics for kernel, reward, and equation search, respectively (Zhang et al., 2025; Ma et al., 2024; Saveliev et al., 2026). Self-Refine and Reflexion use linguistic feedback to guide revisions (Madaan et al., 2023; Shinn et al., 2023), while Self-Distillation Policy Optimization uses rich feedback for learning (Hübotter et al., 2026). Recent studies examine feedback use directly: CausalGame studies reasoning from experimental observations (Chen et al., 2026b), and CUDAnalyst intervenes on feedback sources at fixed evolutionary states (Chong et al., 2026). We examine complete discovery trajectories, focusing on the trade-off between exploiting auxiliary diagnostics and preserving exploration.

Auxiliary information and guidance in optimization. BO incorporates additional information through richer objective models (Astudillo & Frazier, 2019; Zhang et al., 2019) or acquisition rules that account for feasibility and prior beliefs (Gardner et al., 2014; Hvarfner et al., 2022). Prior-guided BO methods reduce the relative influence of prior guidance as evidence accumulates (Souza et al., 2021; Hvarfner et al., 2022); policy reuse similarly reduces reliance on a transferred policy within an episode (Fernández & Veloso, 2006). These methods motivate controlling how strongly guidance shapes search. We use auxiliary-guided BO to investigate this trade-off and adapt guidance attenuation to test its implications for agentic discovery.

3 Background and Problem Setup

Objective and budget. A discovery agent sequentially proposes candidates $x \in \mathcal{X}$ and submits them to an evaluator. We call each evaluation attempt a *submission*. Valid submissions receive an official score $y = f(x)$, with larger values indicating better solutions. Let x_0 denote the evaluated seed program and $x_1, \ldots, x_T$ the valid post-seed candidates evaluated within the search budget. Run-level performance is

$$V_T = \max_{0 \le t \le T} f(x_t). \tag{1}$$

Discovery seeks to advance the known performance frontier, making the best verified solution found within budget its central objective (Yuksekgonul et al., 2026; Romera-Paredes et al., 2024; Chi et al., 2026). Comparing condition maxima requires accounting for both per-run resources and the number of runs. Section 4 specifies the aggregation, and Appendix A documents run counts and budgets.

Score improvement and discovery difficulty. The difficulty of advancing the performance frontier depends on the task and the quality of the current solution. As search progresses, improvements often become rarer and smaller; obtaining even a modest gain over a strong solution can require substantial optimization or a breakthrough (Yuksekgonul et al., 2026; Chi et al., 2026). Absolute and percentage score differences therefore provide no universal measure of discovery difficulty across tasks or stages of search (Yan et al., 2026). Under a fixed search budget, a numerically small shortfall can represent a substantial barrier to further progress.

Feedback-conditioned search. Let $z_t \in \mathcal{Z}$ denote truthful auxiliary diagnostics returned by the evaluator for candidate x_t, with the official objective and validity criteria held fixed. The evaluation histories with and without these diagnostics are

$$H_t^y = \{(x_i, y_i)\}_{i=1}^t, \qquad H_t^{y,z} = \{(x_i, y_i, z_i)\}_{i=1}^t. \tag{2}$$

Task instructions, workspace state, and operational feedback such as validity reports are implicit in both settings. The search policy generates the next candidate as $x_{t+1} \sim \pi(\cdot \mid H_t)$, encompassing

reasoning, editing, and tool use. Full runs develop different histories; shared-history comparisons isolate the immediate effect of revealing diagnostics.

Auxiliary feedback can help exploit useful information about past candidates while concentrating future evaluations on particular modifications, leaving fewer opportunities to explore alternatives. We study how this trade-off affects V_T; Section 5 develops its connection to auxiliary-guided BO.

4 Truthful Auxiliary Feedback Has Heterogeneous Effects

Under matched objectives and per-run budgets, truthful auxiliary feedback improves best-found solutions on some tasks and lowers them on others.

4.1 Cross-task feedback comparison

Comparison protocol. The study covers ten tasks from the algorithmic section of Frontier-CS (Mang et al., 2025) and five tasks from Frontier-Eng (Chi et al., 2026). Within each task, we compare a *baseline* condition with a *baseline + auxiliary* condition. The baseline provides score-only feedback on Frontier-CS and retains the benchmark-provided base feedback on Frontier-Eng. The auxiliary condition adds diagnostics z_t to the respective baseline. The model, OpenCode harness, initial workspace, prompt scaffold, evaluator, official objective, and per-run budget are held fixed across the two conditions. We use GPT-5.5 for Frontier-CS and DeepSeek V4 Pro for Frontier-Eng.

Repeated runs and score aggregation. Agentic search is stochastic: even under the same configuration, separate runs can follow different search trajectories and reach different final scores. We therefore repeat the search in each feedback condition, using three runs per Frontier-CS task, eight for data-center control, and five for each remaining Frontier-Eng task. We follow Frontier-CS's official agent-evaluation time budget of five hours per run. For Frontier-Eng, we adopt the benchmark's 100-iteration search budget (Chi et al., 2026), implemented in our experiments as a cap of 100 submissions. More details are in Appendix A.

For each task and condition $c \in \{0, z\}$ (baseline or baseline + auxiliary), Table 1(a) reports

$$M_c = \max_{1 \le r \le R_c} V_{T_{c,r}}^{(c,r)} = \max_{1 \le r \le R_c} \max_{0 \le t \le T_{c,r}} f\left(x_t^{(c,r)}\right),$$

where R_c is the number of runs, $T_{c,r}$ is the number of valid post-seed candidates evaluated within the budget of run r, and $x_0^{(c,r)}$ is its evaluated seed program. Following Section 3, we report the strongest solution discovered across repeated searches. These best-found scores summarize what each search condition discovers within the available budget. We compare them within each task, since the tasks use different scoring functions and their score differences do not provide a common measure of discovery difficulty. Run-level outcomes in Appendix B document the accompanying variability. Both gains and losses also persist under equal submission budgets on Frontier-CS (Appendix B.2).

To test whether the observed feedback effects generalize across models, we extend the comparison to additional frontier models on representative tasks, Polyomino Packing and Data-Center Control. Table 1(b) reports their best-found scores alongside the auxiliary feedback supplied for each task. Across the four models shown, auxiliary feedback improves the best-found score on packing and lowers it on data-center control under the regular comparison protocol. This consistency across models motivates a system-level question: how does auxiliary feedback change the allocation of search effort over repeated revisions? We investigate this question in data-center control (Section 4.2), where the recurring score decline and inspectable controller code allow us to trace how agents divide their search budget between refining existing rules and exploring new control dependencies.

4.2 Data-center control: tracing the exploration–exploitation trade-off

In data-center control, the agent searches for a program that coordinates workload execution and deferral, cooling, and battery charging and discharging. The objective rewards reductions in carbon emissions and water use while penalizing dropped or overdue work. The program implements decision rules that map available information, such as grid carbon intensity (CI), temperature, and battery

Table 1: Best-found scores with and without auxiliary feedback. Each score is the maximum over runs; higher is better and bold marks the higher score.

(a) Task-suite results

Frontier-CS

Task	Baseline	+ Aux.
Polyomino Packing	77,339.531	**81,392.321**
OgreSort	100,000.000	100,000.000
Rectangle-Free Points	58,066.965	**58,399.281**
Graph 3-Coloring	80,750.652	**81,038.169**
Max-3-SAT	**98,946.608**	98,946.432
Max-Cut	**87,505.564**	87,313.271
Beacon String	**39,912.499**	38,943.926
Cascading Flip Schedule	**77,427.673**	77,363.995
Mobile Relay Layout	55,122.375	55,122.375
Defensive Lineup	**60,102.372**	60,039.658

Frontier-Eng

Task	Baseline	+ Aux.
Data-center control	**31.4948**	30.1842
Temporal smooth control	0.8421	**0.8422**
JobShop SWV	**91.1019**	91.0630
Topology optimization	**−182.5485**	−183.7623
UAV inspection	48.0412	**75.3020**

5 higher / 8 lower / 2 equal

(b) Representative tasks across models

Polyomino Packing

Geometry; rotation/reflection frequencies; contact density and connectivity.

Model	Baseline	+ Aux.
GPT-5.5	77,339.531	**81,392.321**
Gemini 3.8 Flash	80,778.9	**87,958.7**
Grok 4.6	76,956.0	**77,026.4**
GPT-5.6 Sol	82,783.3	**91,695.7**

Data-center control

Workload, cooling, and battery action frequencies under different carbon-intensity conditions.

Model	Baseline	+ Aux.
GPT-5.5	**31.3242**	31.0021
Gemini 3.8 Flash	**30.1042**	23.8371
Grok 4.6	**33.2293**	32.0660
GPT-5.6 Sol	**32.7106**	32.0897

charge, to control actions. For example, a rule may discharge the battery when CI exceeds a threshold. Both feedback conditions provide the same controller inputs; auxiliary feedback additionally summarizes action frequencies under low and high CI. We examine how agents allocate search effort between refining these rules and exploring new dependencies between actions and information, then relate these patterns to their best-found solutions.

We analyze the first 100 completed post-seed submissions of regular baseline and auxiliary runs across five models. Appendix C specifies the cohort and trace analysis.

Measuring exploitation and exploration. We distinguish *feedback-aligned exploitation*—tuning numerical parameters in existing CI-dependent rules—from *structural exploration*—trying previously unexplored dependencies between control actions and information. For example, refining an existing battery threshold exploits a control rule, whereas introducing remaining time into battery decisions explores a new dependency. Table 2(a) defines the measures: R records exploitation's budget share, C its focus within numerical refinement, and E and Q the frequency and breadth of structural exploration. The same definitions apply to baseline runs, providing a reference for how auxiliary feedback changes search allocation.

More exploitation accompanies narrower exploration. Figure 1(a) compares search allocation between the baseline and auxiliary-feedback conditions using all regular runs. The *longest CI refinement streak* measures the longest uninterrupted sequence of CI-linked refinements. The auxiliary condition has higher median refinement allocation (R) and concentration (C), together with longer CI refinement streaks. Median exploration frequency (E) and breadth (Q) are both lower. Figure 1(b) examines variation within the auxiliary condition: greater refinement allocation accompanies narrower coverage ($\rho = -0.82$). The association remains negative after accounting for model-level differences ($\rho = -0.50$), consistent with a trade-off between refining familiar rules and exploring new control dependencies. Model-level summaries and uncertainty estimates appear in Appendix C.

Search behavior behind the best-found results. To connect these behaviors to the reported performance, Table 2(b) compares the highest-scoring regular run in each model and setting. Across all

Table 2: Search measures and the trajectories behind the best-found results. **(a)** Operational definitions. **(b)** The highest-scoring regular run in each model and setting; all measures describe that same trajectory. Bold marks the higher score within each model.

(a) Measuring exploitation and exploration

Search dimension	Measure	Operational meaning
Exploitation	R	CI-linked numerical refinements / all submissions
	C	CI-linked refinements / all numerical refinements
Exploration	E	Submissions introducing new action–information pair / all submissions
	Q	Distinct new action–information pairs explored beyond the seed

(b) Highest-scoring trajectory in each setting

Model	Feedback	R	C	E	Q	Best score
DeepSeek V4 Pro	Baseline	16.0%	94.1%	10.0%	110	**31.4948**
	+ Auxiliary	43.0%	82.7%	6.0%	42	30.1842
Gemini 3.8 Flash	Baseline	21.0%	72.4%	14.0%	69	**30.1042**
	+ Auxiliary	41.0%	100.0%	10.0%	67	23.8371
GPT-5.6 Sol	Baseline	30.0%	44.8%	10.0%	44	**32.7106**
	+ Auxiliary	47.0%	72.3%	10.0%	36	32.0897
Grok 4.6	Baseline	11.0%	84.6%	14.0%	105	**33.2293**
	+ Auxiliary	12.0%	100.0%	8.0%	93	32.0660
GPT-5.5	Baseline	62.0%	86.1%	5.0%	60	**31.3242**
	+ Auxiliary	53.0%	77.9%	8.0%	42	31.0021

five models, the best auxiliary trajectory explores fewer new dependencies (Q) and reaches a lower maximum score than its baseline counterpart; four also devote more submissions to CI-linked refinement (R). Thus, narrower exploration recurs in the trajectories that produce each setting's strongest result, complementing the allocation pattern across all runs.

Strongest-run trajectories show an early advantage and a lower endpoint. Figure 1(c) follows the five strongest regular runs per condition, selected across models by their best score within 100 completed post-seed submissions. Their mean best-so-far score favors auxiliary feedback at submissions 4–43; baseline overtakes at submission 44 and remains ahead through submission 100 (32.45 versus 31.70). During submissions 26–50, the same baseline runs introduce 13.8 new control pairs per run, compared with 3.2 under auxiliary feedback, while auxiliary runs devote 61.6% of submissions to CI-linked refinement versus 15.2% for baseline. The temporal score contrast thus accompanies a marked difference in how these strong runs allocate their search budget.

Repeated exploitation of feedback also appears in other tasks with lower auxiliary best-found scores (Appendix D). Agents repeatedly refine material layouts, permutation rules, and scheduling priorities suggested by the diagnostics, even after further revisions yield little progress. In Cascading Flip Schedule and JobShop, this persistence consumes many submissions without a meaningful improvement in the best-found solution. These traces expose a recurring failure mode: feedback keeps familiar revisions actionable after their marginal value has diminished. Continuing to exploit that guidance spends search opportunities on a narrowing direction, leaving less budget to develop alternatives.

5 When Useful Feedback Constrains Discovery

We use Bayesian optimization (BO) to isolate how guidance strength changes search while holding the auxiliary measurement rule fixed.

Bridging agentic search with BO. Agentic search and Bayesian optimization share the same sequential structure: each uses a history of evaluated candidates and their feedback to choose the next candidate, then incorporates the new evaluation into that history. Auxiliary feedback can therefore affect both the immediate choice and the evidence available for later decisions. Using the history

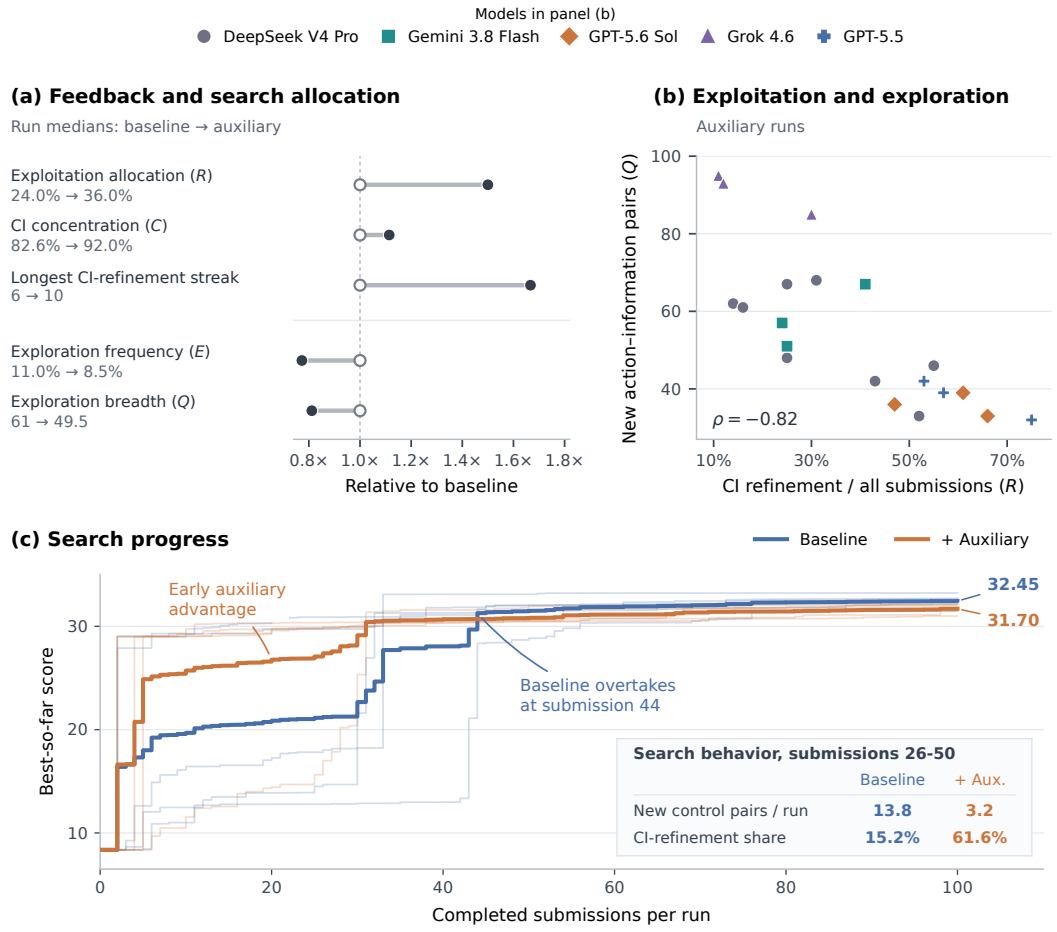


Figure 1: **Feedback concentrates refinement and narrows exploration.** **(a)** Auxiliary/baseline median ratios. **(b)** Refinement (R) versus exploration (Q); each point is an auxiliary run. **(c)** Top five regular runs per condition, ranked across models by best score within 100 post-seed submissions. Thin/thick curves show runs/means. Inset: new control pairs (ΔQ) and CI-refinement share at submissions 26–50.

H_t of candidates, scores, and diagnostics from Section 3, we write the two search loops as

$$\text{Agent:} \quad x^{\mathrm{A}}_{t+1} \sim \pi(\cdot \mid H^{\mathrm{A}}_t), \qquad H^{\mathrm{A}}_{t+1} = H^{\mathrm{A}}_t \oplus (x^{\mathrm{A}}_{t+1}, y^{\mathrm{A}}_{t+1}, z^{\mathrm{A}}_{t+1}), \tag{3}$$

$$\text{BO:} \quad x^{\mathrm{B}}_{t+1} \in \arg\max_x \alpha(x \mid H^{\mathrm{B}}_t), \qquad H^{\mathrm{B}}_{t+1} = H^{\mathrm{B}}_t \oplus (x^{\mathrm{B}}_{t+1}, y^{\mathrm{B}}_{t+1}, z^{\mathrm{B}}_{t+1}), \tag{4}$$

where $\oplus$ appends an evaluation. For the agent, x is a program and π includes reasoning, editing, and tool use; for BO, x is a query point and α is the acquisition rule. This shared structure makes BO a controlled setting for studying how auxiliary guidance changes search allocation. We vary its strength and interrupt it during search while retaining the objective history, tracing how these choices affect the discovery of better solutions. Appendix E develops the connection through a shared view of how feedback reweights candidate preferences.

5.1 Useful guidance in a simple search problem

We design two promising regions: a broad local basin that is easy to find but has a lower peak, and a narrow global basin containing the best solution. This separates refining a familiar solution from discovering a better alternative. Each evaluation returns the objective score and an auxiliary measurement of proximity to either basin. The feedback helps locate promising regions without revealing which has the higher peak.

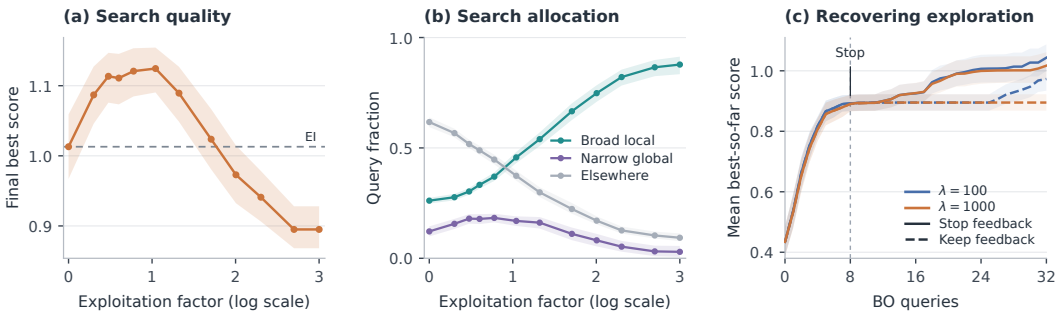


Figure 2: **Strong guidance concentrates search; stopping it restores progress.** **(a)** Final best score; dashed line: EI. **(b)** Query allocation across basins. **(c)** Best-so-far scores with continued guidance (dashed) or a switch to EI after query 8 (solid). Search queries follow eight initial evaluations. Means over 48 paired runs; bands: pointwise 95% bootstrap intervals.

Without auxiliary guidance, BO uses *expected improvement* (EI): it selects candidates based on their estimated potential to improve the best score so far, considering both predicted quality and uncertainty. With auxiliary guidance, candidates predicted to receive higher auxiliary measurements gain additional priority. An *exploitation factor* λ controls this preference: $\lambda = 0$ gives ordinary EI, while larger values make auxiliary guidance more influential. This lets us vary how strongly search follows the same feedback while keeping the objective and measurement rule unchanged.

We compare 48 paired runs per condition, each with eight initial evaluations and 32 search queries. Conditions share starting points and candidate pools, but their histories diverge as they select different candidates. Appendix E gives the objective, EI and guidance-weighting formulas, and full experimental settings.

5.2 From productive focus to missed alternatives

Search becomes more focused before it becomes less successful. Increasing exploitation produces two different patterns. Search becomes progressively more concentrated, while mean final best score first rises from 1.013 under EI to 1.124 at $\lambda = 10$, then falls to .895 at $\lambda = 1000$ (Figure 2a). Both the improvement and the decline recur in a separate set of 48 paired runs. Thus, the same feedback can support better solutions at one strength and restrict discovery at another.

The moderate regime shows why focusing can help. Compared with EI, it allocates fewer queries outside the two promising regions and more queries to *both* basins (Figure 2b). At this moderate strength, focused search accompanies better solutions; the cost emerges as concentration increases further.

Further concentration leaves better alternatives undiscovered. At stronger exploitation, the broad local basin takes an increasing share of the budget, reaching 88% of queries at $\lambda = 1000$, while the narrow global basin receives only 3%. Performance now declines as search concentrates further. The discovery histories make the cost concrete: among 41 runs whose initial design misses the narrow basin, EI and moderate exploitation each reach it in 36 runs; the strongest exploitation reaches it in none. The feedback still marks both promising regions, yet its strong influence on sequential choices leaves search concentrated in the inferior one.

Interrupting guidance restores exploration and progress. We next stop auxiliary guidance after eight search queries at $\lambda = 100$ or 1000 and use objective-only EI for the remaining budget. Each interrupted run shares its initialization and first eight queries with its continuing-guidance counterpart, retaining all evaluated points and objective observations. Stopping guidance raises mean final best score from .973 to 1.043 at $\lambda = 100$ and from .895 to 1.017 at $\lambda = 1000$ (Figure 2c), compared with 1.013 for EI throughout. Interrupting strong guidance restores discovery of the better basin: at $\lambda = 1000$, subsequent search reaches it in 36 of the 41 runs that had not reached it when guidance stopped, compared with none under continued guidance. Thus, the improvement accompanies renewed discovery of the better region within the same evaluation budget.

These results show that the exploration cost of strong guidance can be reduced without discarding the search history it produced. Feedback can identify promising regions, yet its continued influence can keep search concentrated after local gains become limited. Section 6 examines whether restoring exploration can likewise improve agentic discovery.

6 Combining Early Feedback with Renewed Exploration

Renewing exploration. We probe the proposed mechanism through a staged use of auxiliary feedback. The agent first exploits the diagnostics to refine candidates. When repeated refinements yield limited further gains, we stop supplying auxiliary diagnostics and continue search with objective feedback, retaining the candidate programs and search history. This preserves experience acquired with feedback while opening room to explore other directions. Table 3 reports renewed-exploration runs that exceed the strongest baseline references across four tasks. Appendix F details their search trajectories.

Table 3: Renewed exploration yields solutions exceeding baseline best scores.

Task	Search change	Baseline reference	Renewed exploration run
Data-center control	Time- and temperature-dependent control	33.2293	**33.7400**
Max-Cut	Gain-bucket tabu implementation	87,505.564	**87,563.723**
Cascading Flip Schedule	Revised vertex-selection rule	77,427.673	**77,467.416**
Topology optimization	Direct random initialization	−182.5485	**−180.6064**

Exploration compounds the value of earlier exploitation. The data-center trajectory shows how renewed exploration makes earlier CI-tuning experience productive again. Once nearby threshold changes yield little progress, search introduces episode-time-dependent scheduling and temperature-conditioned decisions. A new scheduling rule reaches 31.6518, and refining its activation window raises the score to 32.0105. Further structural changes and refinement reach 33.7400, exceeding the best baseline across models, 33.2293. Crucially, the final improvement reuses CI-tuning experience to refine a discharge threshold within a newly discovered temperature-conditioned branch. Exploration thus creates a richer strategy on which earlier exploitation can build: new control dependencies give familiar refinements renewed value. The resulting controller also exceeds the best data-center score reported in Frontier-Eng (30.1873) by 3.5527 points, a notable benchmark advance documented in Appendix G.

The same pattern across tasks. The other tasks likewise combined changes in search direction with subsequent refinement, through alternative initialization, a more efficient implementation, or a revised decision rule. Together, these cases motivate a complementary role for the two stages: early feedback develops useful rules and experience; later exploration creates new opportunities to apply and refine them.

7 Conclusion

Truthful feedback can improve familiar solutions while narrowing the search for better ones. Through failure-case analysis, agent traces, and controlled BO experiments, we identify repeated exploitation of useful guidance as a mechanism behind this exploration cost. Renewed exploration reveals a complementary possibility: discovering new strategies can make earlier refinement experience productive again. These findings highlight a central challenge for feedback-guided discovery: exploiting what feedback reveals while preserving opportunities to discover what it leaves unexplored. Understanding when to refine, when to change direction, and how these stages reinforce each other opens a path toward more effective agentic discovery.

AI USE STATEMENT

We used generative AI to assist with writing code, polishing manuscript prose, analyzing agent trace data in batches, and searching for relevant literature. AI tools also assisted with developing the conceptual connection between agentic search and Bayesian optimization, discussing and refining experimental designs, and interpreting results. The authors take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Key code implementations and agent traces are available at `https://anonymous.4open.science/r/feedback-discovery-E31B/`. The supplementary material is organized into experimental protocols (Appendix A), complete results (Appendix B), data-center analysis (Appendix C), cross-task feedback patterns (Appendix D), controlled BO experiments (Appendix E), renewed exploration in agentic search (Appendix F), and additional benchmark results and reproducibility materials (Appendix G).

REFERENCES

Raul Astudillo and Peter Frazier. Bayesian optimization of composite functions. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 354–363. PMLR, 2019. URL https://proceedings.mlr.press/v97/astudillo19a.html.

Terry Chen, Zhifan Ye, Bing Xu, Zihao Ye, Timmy Liu, Ali Hassani, Tianqi Chen, Andrew Kerr, Haicheng Wu, Yang Xu, Yu-Jung Chen, Hanfeng Chen, Aditya Kane, Ronny Krashinsky, Ming-Yu Liu, Vinod Grover, Luis Ceze, Roger Bringmann, John Tran, Wei Liu, Fung Xie, Michael Lightstone, and Humphrey Shi. AVO: Agentic variation operators for autonomous evolutionary search, 2026a. URL https://arxiv.org/abs/2603.24517.

Zhenhao Chen, Yongqiang Chen, Chenxi Liu, Junchi Yu, Xiangchen Song, Zijian Li, Jialin Li, Philip Torr, Bo Han, and Kun Zhang. CausalGame: Benchmarking causal thinking of LLM agents in games, 2026b. URL https://arxiv.org/abs/2607.04293.

Yizhe Chi, Deyao Hong, Dapeng Jiang, Tianwei Luo, Kaisen Yang, Boshi Zhang, Zhe Cao, Xiaoyan Fan, Bingxiang He, Han Hao, Weiyang Jin, Dianqiao Lei, Qingle Liu, Houde Qian, Bowen Wang, Situ Wang, Youjie Zheng, Yifan Zhou, Calvin Xiao, Eren Cai, and Qinhuai Na. Frontier-eng: Benchmarking self-evolving agents on real-world engineering tasks with generative optimization. *arXiv preprint arXiv:2604.12290*, 2026. URL https://arxiv.org/abs/2604.12290.

Yee Hin Chong, Jiaming Wu, Youhui Zhang, and Peng Qu. Towards feedback-to-plan decisions for self-evolving LLM agents in CUDA kernel generation, 2026. URL https://arxiv.org/abs/2605.26720.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006. URL https://www.cs.cmu.edu/afs/cs/user/mmv/www/papers/06aamas-policy-reuse.pdf.

Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pp. 937–945. PMLR, 2014. URL https://proceedings.mlr.press/v32/gardner14.html.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Carl Hvarfner, Danny Stoll, Artur Souza, Marius Lindauer, Frank Hutter, and Luigi Nardi. πBO: Augmenting acquisition functions with user beliefs for bayesian optimization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=MMAeCXIa89.

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. PMLR, 2024. URL https://proceedings.mlr.press/v235/liu24bs.html.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.12931.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.17651.

Qiuyang Mang, Wenhao Chai, Zhifei Li, Huanzhi Mao, Shang Zhou, Alexander Du, Hanchen Li, Shu Liu, Edwin Chen, Yichuan Wang, et al. FrontierCS: Evolving challenges for evolving intelligence. *arXiv preprint arXiv:2512.15699*, 2025. URL https://arxiv.org/abs/2512.15699.

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

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*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Evgeny S. Saveliev, Samuel Holt, Nabeel Seedat, David L. Bentley, Jim Weatherall, and Mihaela van der Schaar. Influence-guided symbolic regression: Scientific discovery via LLM-driven equation search with granular feedback, 2026. URL https://arxiv.org/abs/2605.29184.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

Artur Souza, Luigi Nardi, Leonardo B. Oliveira, Kunle Olukotun, Marius Lindauer, and Frank Hutter. Bayesian optimization with a prior for the optimum. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, 2021. URL https://arxiv.org/abs/2006.14608.

George Tsoukalas, Anton Kovsharov, Sergey Shirobokov, Anja Surina, Moritz Firsching, Gergely Bérczi, Francisco J. R. Ruiz, Arun Suggala, Adam Zsolt Wagner, Eric Wieser, Lei Yu, Aja Huang, Miklós Z. Horváth, Andrew Ferraiuolo, Henryk Michalewski, Edward Lockhart, Codrut Grosu, Thomas Hubert, Matej Balog, Pushmeet Kohli, and Swarat Chaudhuri. Advancing mathematics research with AI-driven formal proof search, 2026. URL https://arxiv.org/abs/2605.22763.

Minghao Yan, Bo Peng, Benjamin Coleman, Ziqi Chen, Zhouhang Xie, Shuo Chen, Zhankui He, Noveen Sachdeva, Isabella Ye, Weili Wang, Chi Wang, Ed H. Chi, Fernando Pereira, Wang-Cheng Kang, Derek Zhiyuan Cheng, and Beidou Wang. PACEvolve: Enabling progress-aware consistent evolution, 2026. URL https://arxiv.org/abs/2601.10657v3.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time, 2026. URL https://arxiv.org/abs/2601.16175.

Yehong Zhang, Zhongxiang Dai, and Kian Hsiang Low. Bayesian optimization with binary auxiliary information, 2019. URL https://arxiv.org/abs/1906.07277.

Zijian Zhang, Rong Wang, Shiyang Li, Yuebo Luo, Mingyi Hong, and Caiwen Ding. CudaForge: An agent framework with hardware feedback for CUDA kernel optimization, 2025. URL https://arxiv.org/abs/2511.01884.

APPENDIX CONTENTS

A Tasks and Experimental Protocol **14**
A.1 Task suite and comparison design . . . 14
A.2 Evaluation feedback . . . 14
A.3 Agent workflow and information access . . . 15
A.4 Initial prompt template . . . 15
A.5 Agent permissions . . . 16

B Complete Experimental Results **16**
B.1 Frontier-CS run-level results . . . 16
B.2 Equal-submission comparison on Frontier-CS . . . 16
B.3 Frontier-Eng run-level outcomes . . . 17

C Data-Center Search Analysis **17**
C.1 Run cohort and outcomes . . . 17
C.2 Numerical refinement of CI-dependent rules . . . 18
C.3 Exploration of action–information dependencies . . . 18
C.4 Search allocation across models . . . 18
C.5 Trajectories behind the strongest outcomes . . . 19

D Feedback-Use Patterns Across Tasks **19**
D.1 Topology optimization: refinement becomes restrictive . . . 19
D.2 Defensive Lineup: local refinement reaches diminishing returns . . . 19
D.3 Cascading Flip Schedule: diagnostics sustain low-return revisions . . . 19
D.4 JobShop: useful guidance leads into a persistent plateau . . . 20

E Controlled BO Experiments **20**
E.1 A unified search-allocation view of agentic discovery and BO . . . 20
E.2 Objective and auxiliary measurement . . . 21
E.3 Expected improvement and auxiliary guidance . . . 22
E.4 Paired design and measures of discovery . . . 22
E.5 Useful guidance and the cost of excessive focus . . . 23
E.6 Recovering exploration by interrupting guidance . . . 23

F Renewed Exploration in Agentic Search **24**
F.1 Protocol and comparison setting . . . 24
F.2 Data-center control: expanding control dependencies . . . 24
F.3 Topology optimization: changing initialization . . . 24
F.4 Max-Cut: enabling deeper search . . . 24
F.5 Cascading Flip Schedule: revising the decision rule . . . 24

G A High-Scoring Data-Center Controller **25**
G.1 Exceeding the published benchmark result . . . 25
G.2 Controller design . . . 25
G.3 Scenario outcomes . . . 25

A Tasks and Experimental Protocol

A.1 Task suite and comparison design

We study ten algorithmic tasks from Frontier-CS (Mang et al., 2025) and five engineering tasks from Frontier-Eng (Chi et al., 2026). Agents search for programs evaluated under each task's official objective and validity criteria. All reported scores are higher-is-better; topology optimization uses negative compliance.

Within each task and model, both conditions share the OpenCode harness, initial workspace and program, prompt scaffold, tools, evaluator, and resource limit. The auxiliary condition adds diagnostics to the baseline evaluation response.

A.2 Evaluation feedback

The baseline includes the official score and operational or failure reports, together with any original benchmark-provided feedback. Auxiliary diagnostics are computed from the candidate and its evaluation output and returned in the same response. They do not change the official score.

Table 4 lists the diagnostics for all 15 tasks. Frontier-CS summarizes structural descriptors across test instances using means and standard deviations; Frontier-Eng reports task-specific behavior or geometry.

Table 4: Auxiliary feedback exposed in the 15 task comparisons.

Benchmark	Task	Truthful auxiliary feedback summary
Frontier-CS	Polyomino Packing	Rectangle aspect ratio; reflection and rotation use; contact density and contact-graph structure.
	OgreSort	Move count and direction; normalized source/destination positions; move-distance and sequence statistics.
	Rectangle-Free Points	Occupied-row/column fractions; occupancy imbalance and maxima; singleton-row/column fractions.
	Graph 3-Coloring	Color-class count, balance, and entropy; label-transition rates and constant-run structure.
	Max-3-SAT	True-variable balance; assignment transitions, runs, and alternation; true-index centroid and dispersion.
	Max-Cut	Partition and degree-mass imbalance; same-side components and isolated vertices; label-transition rates.
	Beacon String	Letter entropy; bigram diversity and transitions; constant runs, alternation, and letter-position statistics.
	Cascading Flip Schedule	Displacement from identity; adjacent index jumps and order trend; consecutive tree-edge frequency.
	Mobile Relay Layout	Hub use; assignment balance and transitions; normalized hub-coordinate spans; singleton-hub frequency.
	Defensive Lineup	Displacement from identity; adjacent index jumps and order trend; adjacent name-length transitions.
Frontier-Eng	Data-center control	Carbon-conditioned load-shifting, cooling, and battery action frequencies.
	Temporal smooth control	Trajectory command magnitude, control slew/smoothness, and limit-use statistics.
	JobShop SWV	Family-level gaps and validity patterns; machine utilization, idle time, bottlenecks, and repair diagnostics.
	Topology optimization	Grayness and binarization; checkerboard and smoothness; connected components and small islands.
	UAV inspection	Per-scene coverage and energy; safety clearances; speed, acceleration, and control-slew statistics.

A.3 AGENT WORKFLOW AND INFORMATION ACCESS

Each run operates in an isolated workspace. The following schematic shows its main files used in the agent's search workflow, using the Python candidate interface:

```
workspace/
|-- problem.md              Task, objective, and interface
|-- candidate.py            Current editable candidate
|-- submit.py               Submission entry point
|-- analyzer.py             Fixed diagnostics (Used in Auxiliary Condition)
`-- archive/
    |-- attempt_0000/       Evaluated seed program
    |   |-- code.py         Saved program
    |   `-- result.json     Score and evaluation report
    |-- attempt_0001/       First submitted revision
    |   |-- code.py
    |   `-- result.json
    `-- ...                 Subsequent evaluated revisions
```

The agent reads `problem.md`, inspects earlier programs and reports, and writes its next revision to `candidate.py`. Calling `python submit.py candidate.py` evaluates that program and records a code snapshot and result in the archive. The archive preserves the seed, earlier revisions, and the best-scoring candidate, allowing search to build on or revisit previous solutions.

Evaluation reports contain the official score and operational feedback; the auxiliary condition additionally exposes the task's diagnostic summaries. Thus, both conditions support the same read–edit–submit loop, with auxiliary feedback enriching the evidence available for the next revision. Hidden evaluation instances and private evaluator artifacts remain outside the agent-visible interface. When enabled, `analyzer.py` is invoked by the evaluation pipeline to produce `auxiliary_feedback`. Its source is inaccessible to the agent and cannot be edited; the agent receives its diagnostic output in the evaluation response. The tree shows both editable search files and fixed pipeline components; task-specific support files are omitted.

A.4 INITIAL PROMPT TEMPLATE

The initial prompt establishes the search workflow, evaluation interface, and available budget. Below is a condensed template for a new blind run with fixed auxiliary diagnostics. Angle-bracketed fields are filled from the run configuration and evaluated seed; the task specification is supplied in `problem.md`.

```
You are improving candidate programs for a discovery problem.
This is a new run starting from the seeded archive.

Workspace directory (work ONLY here):
<workspace>

Files:
- submit.py: evaluate a candidate and archive it

Archive layout:
  archive/attempt_NNNN/code.py      Candidate source code
  archive/attempt_NNNN/result.json  Score and evaluation feedback

Workflow:
1. Read problem.md and inspect archive/ for past attempts
   (code + results).
2. Write the next candidate to candidate.py, overwriting it
   each iteration.
3. Submit it: python submit.py candidate.py
4. Read the printed submit output and the archived result.json,
   including processed_feedback.
5. Iterate until you reach the improvement budget.
```

Table 5: Frontier-CS run-level best official scores. Each entry is the best valid score reached during one five-hour trajectory. Bold marks the largest run-level best within each condition; the final column compares these two condition maxima, matching Table 1. Higher is better.

	Score-only			Auxiliary			
Task	R1	R2	R3	R1	R2	R3	Best Δ
Polyomino Packing	73,856.85	**77,339.53**	72,807.09	**81,392.32**	73,534.96	81,024.94	+4,052.79
OgreSort	**100,000.00**	—	—	70,188.43	**100,000.00**	—	0.00
Rectangle-Free Points	57,970.08	**58,066.97**	57,913.49	57,615.18	**58,399.28**	58,278.27	+332.32
Graph 3-Coloring	**80,750.65**	80,543.63	80,665.60	80,954.61	80,838.25	**81,038.17**	+287.52
Max-3-SAT	**98,946.608**	98,939.063	98,943.274	**98,946.432**	98,887.562	98,938.818	−0.176
Max-Cut	**87,505.56**	87,050.79	87,364.35	86,968.05	87,098.07	**87,313.27**	−192.29
Beacon String	**39,912.50**	39,366.09	35,840.71	38,888.24	38,345.34	**38,943.93**	−968.57
Cascading Flip Schedule	74,590.32	**77,427.67**	77,018.64	76,003.72	66,468.86	**77,363.99**	−63.68
Mobile Relay Layout	**55,122.38**	**55,122.38**	**55,122.38**	**55,122.38**	**55,122.38**	**55,122.38**	0.00
Defensive Lineup	**60,102.37**	59,914.94	59,974.07	59,949.46	60,022.50	**60,039.66**	−62.71

```
Rules:
- Do NOT read, search for, or infer evaluator source code
  or scoring implementation.
- Do NOT read or modify analyzer.py; treat processed_feedback
  as fixed benchmark feedback.
- Do NOT change the required function signature or output format.
- Evaluate every candidate through submit.py.
- Do not modify past attempts in archive/; submit new ones instead.

Current best: <seed_attempt_id> with score <seed_score>
Improvement budget: <remaining_submissions> remaining.

Previous attempts:
<seed_evaluation_summary>

Propose and submit improved algorithms until the budget is used up.
```

A.5 Agent permissions

The blind-run configuration separates candidate development from evaluation infrastructure through explicit OpenCode tool permissions. The agent can read the task statement, candidate programs, and archived evaluation reports, and its edit permission is restricted to `candidate.py`. Past code snapshots and results remain read-only. Evaluator and analyzer source, submission-handler source, internal metadata, checkpoints, and raw evaluation artifacts are excluded from the read allowlist. The agent can invoke `submit.py` without reading or modifying its implementation.

Shell access is limited to submission commands and specified workspace inspection commands. External-directory access, web fetching, web search, and delegated-agent tools are disabled. These restrictions keep candidate development within the run's task and visible search history.

B Complete Experimental Results

B.1 Frontier-CS run-level results

Table 5 reports each run's best valid score. Run numbers identify separate searches within each condition. OgreSort has one available baseline run and two auxiliary runs; all other tasks have three per condition.

B.2 Equal-submission comparison on Frontier-CS

For each task, let K be the smallest number of completed post-seed submissions across all available runs in both conditions. We truncate every run to its first K submissions, counting invalid submissions toward the cutoff, and report each condition's highest valid score, including the seed.

Table 6 gives four improvements, four declines, and two ties. Nine tasks retain their five-hour ordering.

Table 6: Frontier-CS results at equal submission counts. Each task uses its own common cutoff K across both conditions and all available runs. Scores are condition maxima within that cutoff; bold marks the higher score.

Task	K	Score-only	+ Auxiliary	Outcome
Polyomino Packing	150	77,182.570	**81,392.321**	Higher
OgreSort	98	100,000.000	100,000.000	Equal
Rectangle-Free Points	254	57,970.081	**58,399.281**	Higher
Graph 3-Coloring	164	80,692.199	**81,038.169**	Higher
Max-3-SAT	218	98,939.410	**98,941.758**	Higher
Max-Cut	192	**87,393.551**	87,313.271	Lower
Beacon String	175	**39,153.791**	38,943.926	Lower
Cascading Flip Schedule	298	**77,378.231**	77,363.995	Lower
Mobile Relay Layout	283	55,122.375	55,122.375	Equal
Defensive Lineup	195	**60,102.372**	60,016.281	Lower

4 higher / 4 lower / 2 equal

B.3 Frontier-Eng run-level outcomes

Table 7 reports five runs per condition for each task below. Data-center results are listed by model and run in Appendix C.

Table 7: Frontier-Eng run-level results with DeepSeek V4 Pro. Scores include the seed; bold marks the best run within each task and condition.

Temporal smooth control		**JobShop SWV**		**Topology optimization**		**UAV inspection**	
Baseline	+ Aux.	Baseline	+ Aux.	Baseline	+ Aux.	Baseline	+ Aux.
0.841970	0.842094	86.9814	89.4293	−183.7136	**−183.7623**	31.3748	**75.3020**
0.841886	0.842092	90.1892	87.9202	−183.8200	−183.9828	38.8342	55.6414
0.842135	0.842077	88.9215	89.5375	**−182.5485**	−184.4971	47.9385	46.4199
0.841885	0.842056	89.5433	**91.0630**	−184.0053	−184.6384	**48.0412**	55.2462
0.842094	**0.842196**	**91.1019**	86.7514	−183.4298	−184.0547	43.7468	52.5364

C Data-Center Search Analysis

C.1 Run cohort and outcomes

Table 8 reports 18 baseline and 20 auxiliary runs across five models. Scores include the seed and first 100 completed post-seed submissions.

Table 8: Data-center best-found scores by model and run. Every displayed run contributes 100 completed post-seed submissions. Bold marks each model's highest score within a condition.

DeepSeek V4 Pro		**GPT-5.5**		**Gemini 3.8 Flash**		**Grok 4.6**		**GPT-5.6 Sol**	
Baseline	+ Aux.	Baseline	+ Aux.	Baseline	+ Aux.	Baseline	+ Aux.	Baseline	+ Aux.
31.4948	12.2067	**31.3242**	**31.0021**	**30.1042**	**23.8371**	10.4845	30.3582	**32.7106**	31.4620
28.7869	14.6902	—	16.6909	29.3736	13.9382	**33.2293**	**32.0660**	32.1780	**32.0897**
9.8307	23.2464	—	30.8107	30.0401	16.7491	30.6743	15.5532	32.6125	31.8718
23.6944	13.3147	—	—	—	—	—	—	—	—
28.0519	29.9820	—	—	—	—	—	—	—	—
21.8759	**30.1842**	—	—	—	—	—	—	—	—
9.5880	30.0748	—	—	—	—	—	—	—	—
24.7026	25.8613	—	—	—	—	—	—	—	—

C.2 Numerical refinement of CI-dependent rules

We compare consecutive candidates through Python abstract syntax trees (ASTs), ignoring comments and leading docstrings. Replacing numeric literals with placeholders yields a canonical structure that preserves names, operators, calls, booleans, and control flow. A numerical refinement preserves this structure and changes only parameters of existing decision predicates, directly or through scoped assignments. Action identifiers, observation indices, slices, and loop bounds are excluded.

A refinement is *CI-linked* if a changed parameter belongs to a predicate that depends on carbon intensity. This includes other thresholds in a joint rule, such as a battery-charge limit in a condition involving both charge and CI.

For n_{num} numerical refinements and n_{CI} CI-linked refinements, we compute

$$R = \frac{n_{\mathrm{CI}}}{100}, \qquad C = \frac{n_{\mathrm{CI}}}{n_{\mathrm{num}}}.$$

C is undefined when $n_{\mathrm{num}} = 0$. The longest CI refinement streak counts consecutive CI-linked refinements. Transitions with an unparseable candidate at either endpoint remain unclassified. The cohort contains 1,609 numerical refinements, including 1,259 CI-linked refinements.

C.3 Exploration of action–information dependencies

Each dependency pairs a component-specific action with an information family, such as battery discharge with state of charge. Families cover CI and its forecasts, time, queue state, workload, temperature, battery charge, persistent memory, episode steps, and cross-component state. An availability tag also records each returned action, so introducing a new action expands coverage.

Static tracing follows observations, assignments, aliases, helper arguments, output slots, and branch guards to returned actions. Unused observations and uncalled helpers contribute no dependencies; variable renaming and threshold-only changes preserve the pair set.

Let F_t be candidate t's pair set and F_0 the seed's. Newly encountered pairs and the exploration measures are

$$D_t = F_t \setminus \bigcup_{s=0}^{t-1} F_s, \qquad E = \frac{1}{100}\sum_{t=1}^{100} \mathbf{1}\{|D_t| > 0\}, \qquad Q = \left| \left(\bigcup_{t=0}^{100} F_t \right) \setminus F_0 \right|.$$

Repeated pairs do not count as new discoveries; one submission can introduce several pairs.

C.4 Search allocation across models

Table 9 reports model-level and pooled medians for the measures in Figure 1(a).

Table 9: Median search measures by model and condition over all regular runs. Streak is the longest consecutive sequence of CI-linked refinements.

Model	Feedback	R (%)	C (%)	E (%)	Q	Streak
DeepSeek V4 Pro	Baseline	19.5	97.1	8.5	62.0	6.0
	+ Auxiliary	28.0	93.3	8.0	54.5	8.5
Gemini 3.8 Flash	Baseline	24.0	72.4	13.0	50.0	9.0
	+ Auxiliary	25.0	100.0	13.0	57.0	4.0
GPT-5.5	Baseline	62.0	86.1	5.0	60.0	24.0
	+ Auxiliary	57.0	79.2	5.0	39.0	22.0
GPT-5.6 Sol	Baseline	28.5	44.5	10.5	49.5	5.5
	+ Auxiliary	61.0	89.2	10.0	36.0	13.0
Grok 4.6	Baseline	19.0	84.6	13.0	93.0	5.0
	+ Auxiliary	12.0	96.8	9.0	93.0	4.0
All runs	Baseline	24.0	82.6	11.0	61.0	6.0
	+ Auxiliary	36.0	92.0	8.5	49.5	10.0

C.5 Trajectories behind the strongest outcomes

Table 2(b) selects the highest-scoring run within each model and condition. Figure 1(c) instead pools models and selects the five highest-scoring runs per condition at submission 100. This selection remains fixed across the full trajectory. Dark curves are unsmoothed, equally weighted means of the five best-so-far curves, including the seed.

Both selected groups contain the three displayed GPT-5.6 Sol runs. The baseline group additionally contains the strongest Grok 4.6 and DeepSeek V4 Pro baseline runs; the auxiliary group contains the strongest Grok 4.6 and GPT-5.5 auxiliary runs (Table 8).

The submissions 26–50 annotation uses these same runs. New-dependency counts total 69 for baseline and 16 for auxiliary, divided by five to obtain the per-run means. CI-linked refinement counts are 19 and 77, respectively, each divided by 125 submissions to obtain the reported budget shares.

D Feedback-Use Patterns Across Tasks

The following traces examine four tasks with lower auxiliary best-found scores. They show how useful feedback can sustain repeated refinement after its returns diminish.

D.1 Topology optimization: refinement becomes restrictive

The agent optimizes a beam's material layout under a fixed material budget. Diagnostics describe intermediate densities, checkerboarding, disconnected islands, and smoothness.

Guided by the intermediate-density report, the agent introduces a projection that makes the layout more binary. The intermediate-density fraction falls from 15.4% to 4.4%, while the score improves from -192.06 to -187.26. Further refinement brings additional gains, but later attempts to remove checkerboarding and islands damage load-bearing structure.

The separate recovery case changes the initialization after refinement plateaus, improving the score from approximately -180.86 to -180.69 before further tuning (Appendix F.3). Changing the starting layout opens opportunities beyond continued refinement of the existing design.

D.2 Defensive Lineup: local refinement reaches diminishing returns

The agent orders names to minimize query-dependent costs between adjacent positions. Permutation diagnostics and score changes guide revisions to the ordering procedure.

Increasing adjacent-swap passes initially helps, but successive gains shrink from tens of score units to fractions of a unit and then to zero. The agent continues adding passes despite recognizing convergence.

Progress resumes when construction accounts for neighboring-position costs instead of assigning names from a fixed ranking. With local refinement reduced from nine passes to five, the score rises from $59{,}629.55$ to $59{,}769.50$. Further tuning improves the new construction: changing the procedure makes refinement productive again.

D.3 Cascading Flip Schedule: diagnostics sustain low-return revisions

The agent orders tree vertices whose removal changes neighboring states. Diagnostics summarize displacement, ascending adjacency, and consecutive tree edges.

When fewer swap passes lower both a favored structural statistic and the score, the agent seeks cheaper ways to preserve the previous behavior. It repeatedly adjusts pass counts, the portion of the ordering processed, and local swap windows.

Over 38 further submissions, the best score improves by less than 0.001. Small diagnostic differences nevertheless motivate more nearby adjustments, sustaining investment in the same ordering procedure after its objective gains have nearly vanished.

D.4 JOBSHOP: USEFUL GUIDANCE LEADS INTO A PERSISTENT PLATEAU

The agent schedules operations under precedence and machine-capacity constraints, using utilization, idle-time, and instance-family diagnostics.

Changes to remaining-work weights improve both the score and reported idle time, motivating further priority-weight variants. The best score then remains unchanged for 31 submissions. Alternative procedures, including operation swaps and simulated annealing, fail to improve the incumbent or exceed the evaluation limit. The agent returns to the familiar parameter grid despite recognizing saturation.

Together, these cases show how productive feedback-guided refinement can become persistent low-return search. Renewed progress requires developing alternatives, not merely recognizing that the current direction has plateaued.

E CONTROLLED BO EXPERIMENTS

The controlled experiments isolate how strongly using a fixed auxiliary signal changes search. The objective, feedback, initialization, and evaluation budget remain the same across guidance strengths. This lets us connect three observations: useful guidance improves discovery, excessive guidance concentrates search around an accessible local optimum, and interrupting that guidance restores exploration.

E.1 A UNIFIED SEARCH-ALLOCATION VIEW OF AGENTIC DISCOVERY AND BO

We develop a mechanistic bridge between agentic discovery and Bayesian optimization: **auxiliary feedback reweights candidate preferences, thereby changing both the next evaluation and the evidence available for future search.** This view connects the code-level allocation patterns in Section 4.2 to a controlled system in which the strength of feedback use can be varied independently of the measurement itself.

A common representation of feedback use. At a fixed evaluation history H_t, let $\pi_0(x \mid H_t)$ denote the agent's candidate distribution with baseline feedback and $\pi_z(x \mid H_t)$ its distribution when the corresponding auxiliary measurements are also exposed. For distributions with the same positive support, their log-density ratio gives an exact representation:

$$\pi_z(x \mid H_t) = \frac{\pi_0(x \mid H_t)\exp(r_t(x))}{\sum_{x'} \pi_0(x' \mid H_t)\exp(r_t(x'))}, \qquad r_t(x) = \log \frac{\pi_z(x \mid H_t)}{\pi_0(x \mid H_t)}. \tag{5}$$

Here H_t fixes the evaluated candidates and outcomes; the two policies differ in access to the auxiliary measurements. The representation makes feedback's behavioral effect explicit: it changes the relative preference for candidates. Parameterizing a shift as $r_t(x) = \lambda g_t(x) + \text{const}$ separates the direction of guidance, g_t, from its strength, λ. In the controlled model, these become independently specified quantities.

An explicit counterpart in BO. Multiplicative acquisition weighting provides an established way to incorporate guidance into BO (Hvarfner et al., 2022). We use it to instantiate the same preference shift: $\alpha_t^{\lambda}(x) = \mathrm{EI}_t(x)\exp(\lambda g_t(x))$. To express the connection at the policy level, consider the normalized weights

$$q_{\lambda,\tau}(x \mid H_t) = \frac{[\mathrm{EI}_t(x)\exp(\lambda g_t(x))]^{1/\tau}}{\sum_{x'}[\mathrm{EI}_t(x')\exp(\lambda g_t(x'))]^{1/\tau}} \;\propto\; q_{0,\tau}(x \mid H_t)\exp\left(\frac{\lambda g_t(x)}{\tau}\right), \tag{6}$$

for $\tau > 0$ on a finite candidate pool with positive EI. This has the same exponential reweighting form as Equation 5. As $\tau \to 0$, its mass concentrates on acquisition maximizers; our experiments select a maximizer deterministically. The temperature is used here to express the correspondence, while λ is the experimental control.

For any two candidates with positive weights,

$$\log \frac{q_{\lambda,\tau}(x \mid H_t)}{q_{\lambda,\tau}(x' \mid H_t)} = \log \frac{q_{0,\tau}(x \mid H_t)}{q_{0,\tau}(x' \mid H_t)} + \frac{\lambda}{\tau}\big(g_t(x) - g_t(x')\big). \tag{7}$$

Thus, stronger guidance increases the relative preference for feedback-favored candidates even when the feedback itself is unchanged.

From preference shifts to discovery costs. The bridge identifies a sequential mechanism. Reweighting changes which candidates are evaluated; those evaluations determine the next history, which shapes later candidate preferences. Repeatedly favoring one productive region can therefore reduce opportunities to acquire evidence about better alternatives. Truthfulness concerns the measurement, whereas discovery depends on the trajectory induced by using it.

Our contribution is to connect this mechanism across observed agentic search and controlled BO interventions. The agent traces reveal repeated feedback-aligned refinement; the BO experiments isolate how guidance strength changes allocation, discovery, and recovery after interruption. The correspondence concerns candidate preferences and history formation. It turns the empirical failure pattern into a testable search-allocation account and motivates restoring exploration while preserving the useful experience acquired with feedback.

E.2 Objective and auxiliary measurement

We maximize a deterministic objective on $x = (x_1, x_2) \in [0,1]^2$. Define two Gaussian components,

$$b_j(x) = \exp\left[-\frac{1}{2}\sum_{k=1}^{2}\left(\frac{x_k - c_{j,k}}{s_{j,k}}\right)^2\right], \qquad j \in \{b, n\}, \tag{8}$$

with centers $c_b = (0.30, 0.34)$ and $c_n = (0.77, 0.72)$, and scales $s_b = (0.145, 0.115)$ and $s_n = (0.052, 0.060)$. The objective is

$$\begin{aligned} f(x) = {} & 0.82\, b_b(x) + 1.18\, b_n(x) \\ & + 0.16 \exp\left[-\frac{1}{2}\left(\frac{x_2 - (0.72 - 0.42x_1)}{0.105}\right)^2\right] + 0.035 \sin(8\pi x_1)\cos(6\pi x_2). \end{aligned} \tag{9}$$

Figure 3 visualizes the objective. The broad component creates an accessible local optimum; the narrow component contains the better solution and requires more precise exploration. A diagonal ridge and small oscillations add structure between and around these regions. This design separates finding a productive region from discovering the best region.

At each evaluated point, auxiliary feedback reports

$$p(x) = \frac{b_b(x) + b_n(x)}{\sqrt{2}}. \tag{10}$$

This measurement identifies proximity to both productive regions, weighting their components equally, while the objective assigns them different heights. It therefore supplies useful directional information while leaving the optimizer to discover their relative value. Both $f(x)$ and $p(x)$ are observed only at evaluated points; separate surrogate models predict their values elsewhere. The auxiliary measurement is fixed across all guidance strengths and interruption experiments.

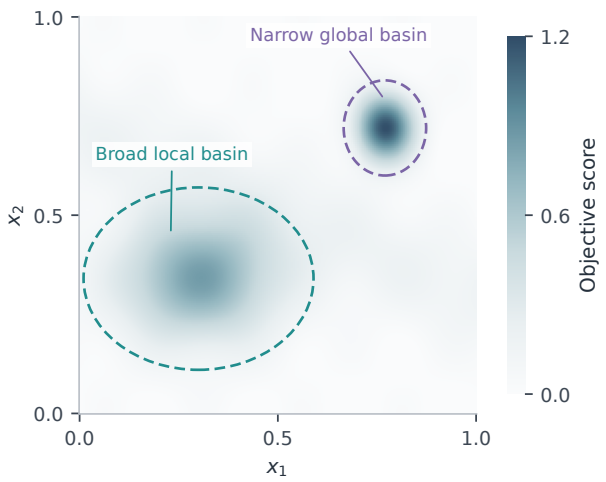


Figure 3: Objective landscape with a broad, lower local optimum and a narrow, higher global optimum. Dashed ellipses mark the basin boundaries used to measure search allocation. The auxiliary signal weights the two Gaussian components equally, indicating proximity without encoding their different objective heights.

E.3 Expected improvement and auxiliary guidance

At each step, we fit separate Gaussian processes (GPs) to the observed objective and auxiliary values. Both use a unit-amplitude radial basis function kernel, with fixed length scales 0.11 for the objective and 0.15 for the auxiliary measurement. Targets are centered and scaled at each fit, and 10^{-6} is added to the kernel diagonal. Objective predictions are transformed back to the original score scale.

Let $\mu_t(x)$ and $\sigma_t(x)$ be the objective posterior mean and standard deviation, and let f_t^+ be the best observed score. Expected improvement is

$$\mathrm{EI}_t(x) = (\mu_t(x) - f_t^+ - \xi)\Phi(u_t(x)) + \sigma_t(x)\phi(u_t(x)), \qquad u_t(x) = \frac{\mu_t(x) - f_t^+ - \xi}{\sigma_t(x)}, \tag{11}$$

where Φ and ϕ are the standard normal CDF and density, and $\xi = 0.01$. EI assigns value to candidates through both their predicted score and their uncertainty.

Let $\mu_t^p(x)$ be the auxiliary posterior mean and $\mathcal{C}_t$ the remaining candidate pool. We standardize this mean over $\mathcal{C}_t$ and clip it to obtain

$$g_t(x) = \mathrm{clip}\left(\frac{\mu_t^p(x) - \mathrm{mean}_{x' \in \mathcal{C}_t}\, \mu_t^p(x')}{\mathrm{std}_{x' \in \mathcal{C}_t}\, \mu_t^p(x')}, -8, 8\right). \tag{12}$$

If the denominator is below 10^{-12}, we set $g_t(x) = 0$. The next query maximizes

$$\alpha_t^\lambda(x) = \mathrm{EI}_t(x)\exp\big(\lambda g_t(x)\big), \qquad x \in \mathcal{C}_t. \tag{13}$$

Thus $\lambda = 0$ gives objective-only EI, while positive λ favors candidates with higher predicted auxiliary measurements. We select deterministically by maximizing the equivalent log acquisition, $\log \mathrm{EI}_t(x) + \lambda g_t(x)$. Stable evaluation of log EI preserves distinctions between very small EI values at large λ. Ties are resolved by candidate index.

E.4 Paired design and measures of discovery

Each replicate uses a 65×65 candidate grid. Interior coordinates receive uniform jitter in $[-0.0025, 0.0025]$, with boundary coordinates fixed. Eight initial evaluations comprise four anchors, $(0.08, 0.08)$, $(0.10, 0.88)$, $(0.50, 0.50)$, and $(0.88, 0.12)$, plus four points sampled uniformly from $[0.05, 0.95]^2$; each is mapped to a distinct nearest candidate. Search then makes 32 additional queries without replacement, for 40 objective evaluations in total.

Table 10: Independent confirmation with 48 new paired replicates. Differences compare final best scores within matched replicates; brackets give 95% bootstrap intervals.

Search condition	Mean final best score	Paired difference from EI
EI ($\lambda = 0$)	0.982	—
Moderate guidance ($\lambda = 10$)	1.071	$+0.089$ [0.038, 0.140]
Strong guidance ($\lambda = 1000$)	0.902	-0.080 [-0.135, -0.026]

The main sweep uses 48 paired replicates and $\lambda \in \{0, 1, 2, 3, 5, 10, 20, 50, 100, 200, 500, 1000\}$. Within a replicate, every condition shares its candidate pool and initial observations. Subsequent queries follow each condition's own accumulated history. The final best score is the maximum over all 40 evaluations; query zero in the time-course plot is the best score after initialization.

We define basin j by $\sum_{k=1}^{2}((x_k - c_{j,k})/s_{j,k})^2 \leq 4$. The broad and narrow basins are disjoint. Search allocation is the fraction of the 32 post-initialization queries falling in each basin or elsewhere. Discovery records whether a run has evaluated any point in the narrow basin. To measure new discovery, we condition on runs that have not yet visited that basin at initialization or at the interruption point, as appropriate. Means and 95% bootstrap intervals use replicates as the sampling unit; paired contrasts resample matched replicate differences. Figure bands are pointwise intervals.

E.5 Useful guidance and the cost of excessive focus

Moderate guidance improves the mean final best score from 1.013 under EI to 1.124 at $\lambda = 10$. It increases the query share in both productive regions: from 26.1% to 45.7% in the broad basin and from 12.2% to 16.9% in the narrow basin. Guidance therefore helps convert the evaluation budget into productive search.

At $\lambda = 1000$, allocation shifts to 87.8% in the broad basin and only 2.9% in the narrow basin, while the mean final best score falls to 0.895. Among the 41 replicates whose initialization misses the narrow basin, EI and $\lambda = 10$ each discover it in 36 runs; $\lambda = 1000$ discovers it in none. Strong guidance repeatedly directs queries toward the accessible region and displaces the queries needed to find the better alternative.

A separate set of 48 paired replicates evaluates EI, $\lambda = 10$, and $\lambda = 1000$ on the same landscape with new candidate pools and initial samples. Table 10 shows that both signs recur: moderate guidance helps, while excessive guidance hurts. These results establish the central contrast under a fixed, useful auxiliary measurement.

E.6 Recovering exploration by interrupting guidance

We interrupt guidance in two strongly guided settings, $\lambda = 100$ and $\lambda = 1000$. The interrupted condition follows the corresponding fixed-guidance condition for the first eight search queries, then uses objective-only EI for queries 9–32:

$$\lambda_t = \begin{cases} \lambda, & 1 \leq t \leq 8, \\ 0, & 9 \leq t \leq 32. \end{cases} \tag{14}$$

Each pair shares exactly the same history through query eight. After interruption, the optimizer retains every objective observation, its incumbent, and its remaining candidate pool; subsequent decisions use the objective GP alone. The total evaluation budget remains unchanged.

Interrupting guidance raises the mean final best score from 0.973 to 1.043 at $\lambda = 100$, and from 0.895 to 1.017 at $\lambda = 1000$. EI throughout attains 1.013 on the same paired replicates. The basin visits explain the recovery: at $\lambda = 1000$, 41 runs have still not reached the narrow basin after eight queries. Continued guidance finds it in none of these runs, whereas interruption finds it in 36.

The improvement follows a change in where search goes next. The optimizer retains what it has learned about the broad region and uses the remaining queries to discover alternatives that sustained guidance leaves unexplored. Together with the moderate-guidance benefit, this result motivates methods that balance the productive use of feedback with the exploration it displaces.

F RENEWED EXPLORATION IN AGENTIC SEARCH

F.1 PROTOCOL AND COMPARISON SETTING

Table 3 compares the renewed-exploration runs with each task's highest baseline score across models. Data-center control and topology optimization use a budget of 100 post-seed submissions; Max-Cut and Cascading Flip Schedule use five-hour searches.

The strategy starts with auxiliary diagnostics and later continues with objective feedback while retaining search history. The trajectories below locate renewed exploration where the observed marginal value of further exploitation has diminished: nearby revisions produce little or no improvement in the incumbent. Submission numbers count completed post-seed evaluations.

F.2 DATA-CENTER CONTROL: EXPANDING CONTROL DEPENDENCIES

Submission 19 reaches 30.8368. CI-threshold refinements at submissions 20–21 add no gain, and the agent explicitly recognizes diminishing returns from battery tuning. It tests load shifting at submission 22, then introduces episode-time-dependent deferral at submission 23, reaching 31.6518. Refining its activation window at submission 24 raises the score to 32.0105.

The pattern recurs: submissions 25–28 leave the incumbent unchanged, before episode-dependent queue-age limits at submission 29 reach 33.1327. Temperature-conditioned scheduling at submission 36 and battery timing at submission 65 open further gains. At submission 97, CI-threshold refinement inside the new temperature-conditioned branch reaches 33.7400. Thus, exploration follows diminishing returns and creates new opportunities to apply earlier tuning experience. Appendix G describes the final controller and evaluation outcomes.

F.3 TOPOLOGY OPTIMIZATION: CHANGING INITIALIZATION

Between submissions 75 and 90, the incumbent improves by only about 10^{-6} from approximately -180.8568. At submission 91, replacing perturbed strain-energy initialization with a directly sampled random density field reaches -180.6899, while retaining the subsequent optimizer. Refining the initialization range and penalty schedule reaches -180.6064 at submission 96. A different starting layout creates opportunities beyond continued refinement of the previous one.

F.4 MAX-CUT: ENABLING DEEPER SEARCH

The best score remains $87{,}441.16$ from submission 113 through 177. The gain-bucket redesign at submission 177 initially matches that score, but enables deeper tabu search to reach $87{,}526.72$ at submission 179. Further depth, restart, and perturbation changes reach $87{,}563.72$ at submission 223. Implementation exploration creates computational room for productive refinement.

F.5 CASCADING FLIP SCHEDULE: REVISING THE DECISION RULE

Local reordering reaches $77{,}226.78$ at submission 18; a larger window times out, and alternative starting orders bring no gain. At submission 22, the agent changes vertex-selection priority to

$$8\,\mathrm{delta} - \mathrm{remaining_degree},$$

where delta is immediate cost reduction and $\mathrm{remaining_degree}$ counts remaining neighbors. This lets degree outweigh a small disadvantage in immediate gain and raises the score to $77{,}453.98$. Refining the rule and reordering window reaches $77{,}467.42$ at submission 39.

Across these tasks, renewed exploration changes the control dependencies, initialization, implementation, or decision rule, creating new opportunities for refinement.

G A HIGH-SCORING DATA-CENTER CONTROLLER

G.1 EXCEEDING THE PUBLISHED BENCHMARK RESULT

As a remarkable by-product of this study, our GPT-5.6 Sol search discovers a controller scoring **33.7400** within 100 post-seed submissions (33.7247 on repeat evaluation). This exceeds the best data-center score in Frontier-Eng's final table, 30.1873 (Qwen), by 3.5527 points (Chi et al., 2026, Table 7, arXiv v2), and our strongest regular baseline, 33.2293. Appendix F.2 details its search trajectory.

G.2 CONTROLLER DESIGN

The policy combines current and forecast CI with queue occupancy, job age, and workload to schedule flexible work. A running temperature average selects scheduling and battery-discharge thresholds, adapting the CI rules to the environmental regime. Separate terminal windows govern workload deferral and the discharge of remaining battery energy; cooling uses a fixed lower-setpoint action. These explicit rules coordinate decisions across time, queue state, temperature, and charge.

G.3 SCENARIO OUTCOMES

All four scenarios score above 32 (Table 11). Across their rollouts, aggregate carbon emissions and grid energy use are 13.29% and 11.85% below the no-op reference, respectively. Water use increases by 2.51%, so the score gains come from carbon savings, with no water-reduction credit.

Table 11: Scenario outcomes of the 33.7400 controller. Carbon reductions are relative to the evaluator's no-op reference. No dropped or overdue tasks are reported in these rollouts.

Scenario	Score	Carbon reduction
Arizona summer	33.7228	13.38%
California spring	35.4342	14.77%
New York winter	32.2432	12.23%
Texas late summer	33.5597	13.25%