BELIEFOPT: RECURSIVE SELF-EVOLVING AGENTS FOR AUTONOMOUS RESEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Research is the disciplined conversion of experience into belief. Autonomous research agents accumulate experience through iterative experimentation, creating an opportunity to continually improve their capacity for discovery. Realizing this potential requires turning experimental outcomes into reliable knowledge and revising that knowledge as new evidence emerges. Yet lessons inferred from past success may reflect contingent advantages or unreliable measurements, allowing plausible but unsupported claims to shape subsequent research. We introduce BELIEFOPT, a framework for autonomous research that organizes experience into falsifiable, revisable beliefs about when and why one choice should outperform an alternative. A research scientist distills experimental evidence into beliefs, while an engineering squad uses them to explore candidate solutions and returns evidence that informs their subsequent selection and revision. This interaction couples the evolution of the belief store with the research process it guides, supported by an evaluator that expands test coverage as experimentation proceeds. We use GPU kernel optimization as a representative setting for studying autonomous research. On 26 GPU kernels from SOL-ExecBench on NVIDIA B200, BELIEFOPT operates from an empty belief store without human intervention, achieving a geometric-mean speedup of $23.9\times$, exceeding OpenEvolve by 24% and CodeEvolve by 6%. Exploratory ablations suggest that organizing relationships among beliefs improves search at larger budgets, while accumulating additional experience or seeding expert advice does not consistently help. These findings support an approach to autonomous research in which accumulated knowledge remains open to testing and revision, allowing experimental evidence to continually reshape both what an agent believes and how it investigates.

1 INTRODUCTION

Large language models increasingly conduct autonomous research through iterative cycles of proposing, executing, and evaluating candidate solutions (Romera-Paredes et al., 2024; Novikov et al., 2025; Lange et al., 2026; Lu et al., 2024). These cycles generate experience that could improve an agent's capacity for discovery. Yet experimental success does not establish why a solution works, and lessons drawn from past outcomes may become unreliable as research progresses. The central challenge is to convert this experience into knowledge that improves subsequent research while remaining accountable to new evidence.

Existing approaches preserve experience through evaluated programs, reflections, reusable skills, and curated advice (Romera-Paredes et al., 2024; Shinn et al., 2023; Wang et al., 2023; Zhao et al., 2024; Ouyang et al., 2026). These mechanisms support increasingly effective reuse and revision. Recent work also makes hypotheses explicit for validation, records their evolution, and updates beliefs using evidence from previous discoveries (Huang et al., 2025; Takahara & Mizoguchi, 2026; Agarwal et al., 2026). A remaining question is how an agent can use revisable claims about comparative performance to guide experiments and assess their utility as optimization progresses. We approach this question through the principle that *research is the disciplined conversion of experience into belief*. A belief is a provisional claim whose predictions guide investigation and whose scope and utility remain subject to revision. This creates a coupled learning problem because beliefs shape the experiments an agent conducts, while those experiments supply the evidence available to revise its beliefs.

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.

We introduce BELIEFOPT, a framework that organizes experimental experience into falsifiable, revisable beliefs about when and why one choice should outperform an alternative. A research scientist forms and revises these claims, while an engineering squad uses them to explore candidate solutions and returns evidence that updates their influence on subsequent research. Relationships among beliefs guide refinement and exploration, and an unguided alternative allows search beyond stored advice. An evolving evaluator expands test coverage and prevents gains rejected by broader validation from reinforcing the beliefs that produced them. Together, these components integrate belief formation, use, and revision into autonomous research, tracking the practical utility of following a belief without treating performance gains as proof of its proposed mechanism.

We use GPU kernel optimization (Lin et al., 2026) as a representative setting for studying autonomous research. It requires agents to formulate hypotheses about computational performance, translate them into executable implementations, and revise their understanding through repeated experiments under explicit correctness constraints. This setting allows us to examine whether accumulated beliefs improve an agent's ability to conduct research as the optimization frontier advances. We evaluate BELIEFOPT on 26 GPU kernels from SOL-ExecBench (Lin et al., 2026) on NVIDIA B200. Starting from an empty belief store without human intervention, BELIEFOPT achieves a geometric-mean speedup of $23.94\times$ under matched backbone models and per-task token budgets, exceeding OpenEvolve (Sharma, 2025) by 24% and CodeEvolve (Assumpção et al., 2025) by 6%. Exploratory ablations suggest that relationships among beliefs become useful at larger search budgets, while additional experience and expert advice do not consistently help. These results demonstrate the performance of the complete system and characterize the opportunities and limitations of using accumulated beliefs to improve subsequent research.

Our contributions are threefold. First, we formulate autonomous research as a coupled process of forming, testing, and revising beliefs from experimental experience. Second, we introduce BELIEFOPT, a framework that integrates this process with experimental exploration and evolving evaluation to improve how agents conduct research over time. Third, we evaluate BELIEFOPT using GPU kernel optimization as a representative setting for autonomous research, demonstrating improved performance under matched budgets and examining the benefits and limitations of accumulated beliefs.

2 BELIEFOPT

BELIEFOPT couples the development of research beliefs with the experiments they guide. A research scientist distills and revises beliefs from measured outcomes, an engineering squad translates selected beliefs into candidate solutions, and an evolving evaluator supplies evidence for both solution improvement and belief revision. Fig. 1 summarizes this process.

Research setting. We study autonomous research in executable environments, where an agent proposes candidate solutions to a task t, evaluates their quality subject to correctness constraints, and seeks the best valid solution within a token budget B_t. The agent processes a set of tasks, searching each within its own budget. Search proceeds in rounds, each of which generates and evaluates a small number of candidate solutions. At round n, the agent maintains an incumbent $p^{\star}_{t,n}$, a belief store $\mathcal{B}_n$, and an evaluation suite $\mathcal{E}_{t,n}$. Experiments update both the incumbent and the knowledge guiding subsequent investigation. The backbone model remains fixed, so adaptation occurs through these external states and the selection of research directions. We instantiate this setting in GPU kernel optimization, where candidate programs are evaluated for numerical correctness and latency. Benchmark details and score aggregation appear in §B.1, with comparison conditions in §D.

2.1 REPRESENTING RESEARCH BELIEFS

A belief expresses a conditional comparison through four fields (C, A, B, M). Under condition C, choice A is predicted to outperform alternative B because of a proposed mechanism M. Each entry also records its scope, experimental evidence and provenance, utility statistics, examination history, and lifecycle status. A claim is admitted only when it specifies a comparison that subsequent experiments could challenge. Fixed correctness and evaluation constraints remain outside the learned store. Task beliefs concern solutions within a particular task, while meta beliefs concern portable research strategies, such as changing direction after a plateau.

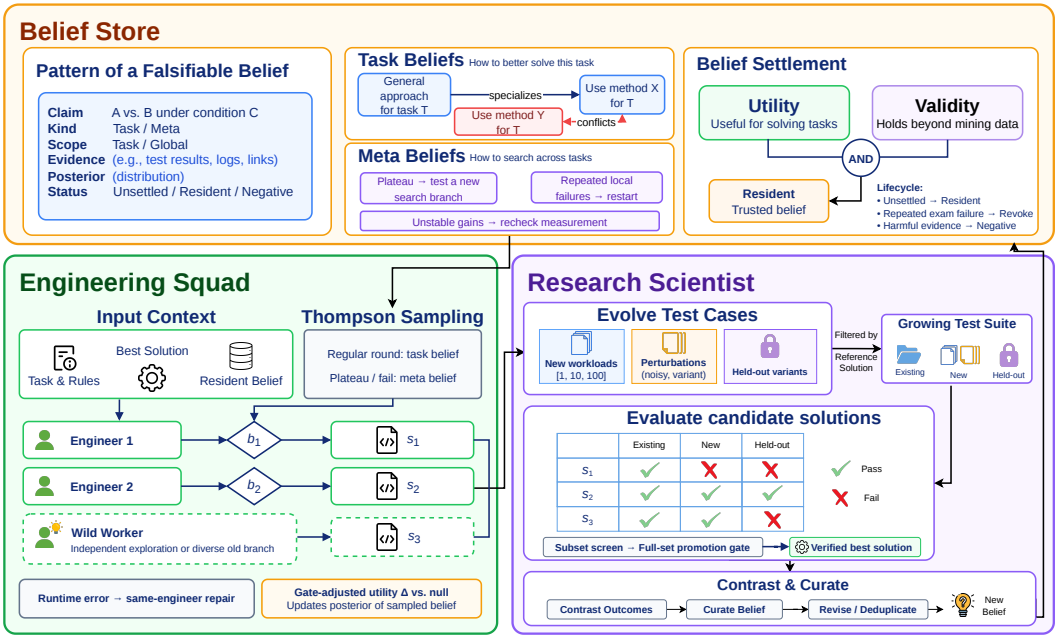

Figure 1: BELIEFOPT couples belief development with autonomous experimentation. The research scientist forms and revises conditional claims, the engineering squad explores implementations guided by selected claims or an unguided alternative, and the evaluator checks candidate outcomes before they update the incumbent and belief utility. Relationships among beliefs guide exploration without contributing statistical evidence.

This representation separates a belief's predicted advantage from its proposed explanation. A useful direction may improve performance without establishing that M is correct. Accordingly, utility statistics estimate the value of following a belief, while the evidence record supports reassessment of its conditions and interpretation. The store also records semantic relationships among claims, enabling the system to distinguish repeated observations from new hypotheses.

2.2 BELIEF-GUIDED EXPERIMENTATION

Each main engineer receives the task specification, incumbent, applicable standing guidance, and one sampled belief as a research direction. The engineer determines how to implement that direction. A permanent null arm supplies no additional direction, allowing stored advice to compete with unguided exploration. The two main engineers draw independently, and an auxiliary slot explores without belief guidance or revisits an archived branch when progress stalls. Auxiliary candidates receive no belief attribution.

For each eligible belief and the null arm, BELIEFOPT maintains task-specific and global Normal-Inverse-Gamma utility posteriors. For engineer k, independent Thompson draws determine the selected direction

$$b_{t,n}^{(k)} = \arg\max_{b \in \mathcal{A}_{t,n}} \left[\widetilde{\mu}_b^{(t,k)} + w_{b,t} \widetilde{\mu}_b^{(g,k)} \right], \qquad w_{b,t} = \frac{m_b}{m_b + n_{b,t}}, \quad m_b = \min(n_{b,g}, 3), \tag{1}$$

where $\mathcal{A}_{t,n}$ includes the null arm and $n_{b,t}$ and $n_{b,g}$ count task and global observations. The global term provides a diminishing transfer bias as local evidence accumulates. To accommodate an improving incumbent, older observations are discounted before each update. With probability 0.2, an engineer instead tests a previously untried belief from a graph-ordered queue. Plateau or repeated-failure rounds switch selection to meta beliefs. §B.2 gives posterior updates and scheduling details.

Algorithm 1 One iteration of BELIEFOPT

Require: Task t, reference r_t, incumbent $p^\star_{t,n}$, belief store $\mathcal{B}_n$, evaluation suite $\mathcal{E}_{t,n}$
Ensure: Updated $p^\star_{t,n+1}$, $\mathcal{B}_{n+1}$, and $\mathcal{E}_{t,n+1}$

$(\mathcal{A}, \mathcal{U}, \mathcal{H}) \leftarrow \textsc{PrepareContext}(\mathcal{B}_n, t)$ ▷ Arms, untried queue, standing guidance
for $k \in \{1, 2\}$ **do**
  $u_k \sim \mathrm{Uniform}(0, 1)$
  if $u_k < 0.2$ and $\mathcal{U} \neq \emptyset$ **then**
    $b^{(k)}_{t,n} \leftarrow \textsc{PopFirst}(\mathcal{U})$
  else
    $b^{(k)}_{t,n} \leftarrow \textsc{ThompsonSelect}(\mathcal{A}, \mathcal{B}_n, t)$ ▷ Independent draws using Eq. 1
  end if
  $c_k \leftarrow \textsc{Engineer}(t, p^\star_{t,n}, \mathcal{H}, b^{(k)}_{t,n})$
end for
$c_3 \leftarrow \textsc{ExploreOrRevive}(t, p^\star_{t,n})$ ▷ No belief attribution
$\mathcal{O} \leftarrow \emptyset$; $\mathcal{B}_{n+1} \leftarrow \mathcal{B}_n$
for $k \in \{1, 2, 3\}$ **do**
  $c_k \leftarrow \textsc{Repair}(c_k, t)$
  $o_k \leftarrow \textsc{Evaluate}(c_k, t, \mathcal{E}_{t,n})$ ▷ Screening and full-workload validation
  if $o_k \neq \perp$ **then**
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{(c_k, o_k)\}$
    if $k \leq 2$ **then**
      $\delta_k \leftarrow \textsc{GateAdjustedUtility}(o_k, p^\star_{t,n})$
      $\mathcal{B}_{n+1} \leftarrow \textsc{UpdateUtility}(\mathcal{B}_{n+1}, b^{(k)}_{t,n}, t, \delta_k)$
    end if
  end if
end for
$p^\star_{t,n+1} \leftarrow \textsc{Promote}(p^\star_{t,n}, \mathcal{O})$
$\Delta\mathcal{E} \leftarrow \textsc{ProposeTests}(t, \mathcal{B}_{n+1}, \mathcal{O})$
$\mathcal{E}_{t,n+1} \leftarrow \mathcal{E}_{t,n} \cup \{e \in \Delta\mathcal{E} \mid \textsc{Pass}(r_t, e)\}$
$\mathcal{B}_{n+1} \leftarrow \textsc{Examine}(\mathcal{B}_{n+1}, \mathcal{E}_{t,n+1}, p^\star_{t,n+1})$
$\mathcal{B}_{n+1} \leftarrow \textsc{Curate}(\mathcal{B}_{n+1}, t, \mathcal{O})$ ▷ Form, revise, and merge beliefs

2.3 GROUNDING LEARNING IN EXPERIMENTAL EVIDENCE

The evaluator separates inexpensive screening from the checks required to accept an improvement. In our kernel instantiation, a size-stratified workload subset screens candidates, while the full official workload set determines promotion. A candidate must satisfy correctness requirements and exceed the incumbent by 0.5%. The research scientist can expand the test suite with additional workloads and perturbed inputs. Every proposed test must first pass the reference implementation, and accepted tests are retained. This control checks test admissibility without establishing that the suite is exhaustive.

Belief utility is updated only after evaluation resolves the candidate's status. For a promoted candidate, the credited gain is the smaller of its screening and full-workload gains. A candidate rejected by the promotion gate receives zero credit, and a correctness failure receives a bounded negative observation. A valid, non-promoted candidate that passes the gate retains its measured gain. Instrument failures yield no observation and are excluded from belief extraction. The null arm follows the same accounting rules. These rules prevent a screening improvement invalidated by broader evaluation from reinforcing the direction that produced it. Exact attribution cases appear in §B.3.

2.4 FORMING AND REVISING BELIEFS

After evaluation, the research scientist contrasts successful and unsuccessful candidates, their measurements, and accumulated failure records to propose new claims or narrow existing ones. A semantic judge compares each proposal with the scope-compatible store. Equivalent claims are merged, while distinct claims retain relationships such as specialization and conflict. Ambiguous beliefs with recorded failures can produce narrower child hypotheses. Supporting observations priori-

Table 1: Performance on 26 SOL-ExecBench kernels on NVIDIA B200. Speedups over the reference implementations are remeasured on the full workload set and aggregated by geometric mean; n gives the number of kernels in each family. OE, CE, SE, and HA denote OpenEvolve, CodeEvolve, ShinkaEvolve, and the human-assisted predecessor. Autonomous systems have matched per-task token budgets, while HA serves as a historical reference. Bold marks the best autonomous speedup. W/T/L counts per-kernel wins, ties, and losses for BELIEFOPT against each baseline, with differences within $\pm 2\%$ counted as ties. Tokens (M) reports total usage.

		Geometric-mean speedup (×)					BELIEFOPT W/T/L		
Kernel family	n	OE	CE	SE	HA	BELIEFOPT	OE	CE	SE
Add + RMSNorm	3	**23.21**	21.34	22.05	23.37	22.33	2/0/1	2/1/0	1/2/0
GEMM	8	0.86	0.98	**1.02**	1.07	0.99	8/0/0	5/2/1	2/4/2
GQA paged decode	2	1,682	1,766	1,744	1,934	**1,874**	2/0/0	2/0/0	1/1/0
GQA paged prefill	2	830	4,269	3,631	4,426	**4,479**	2/0/0	2/0/0	2/0/0
GQA ragged prefill	2	177	194	181	243	**200**	1/1/0	1/1/0	2/0/0
MLA decode/prefill	2	311	337	240	409	**620**	2/0/0	2/0/0	2/0/0
MoE FP8 block-scaled	1	**4.41**	3.68	1.80	1.00	2.24	0/0/1	0/0/1	1/0/0
RMSNorm	6	17.35	16.86	17.03	17.81	**17.45**	2/3/1	3/3/0	3/3/0
Overall	26	19.32	22.65	21.48	23.65	**23.94**	**19/4/3**	**17/7/2**	**14/10/2**
Tokens (M)		33.4	34.1	33.6	36.9	33.0			

tize specializations for exploration, and refutations prioritize competing claims. During stagnation, the system can also revisit a distinct archived lineage.

Graph relationships influence which hypotheses are examined, but do not add observations or directly modify posterior parameters. Consequently, semantic similarity cannot itself count as empirical support. Beliefs remain revisable after repeated use, and beliefs associated with consistently poor utility or measurement exploitation can be retained as negative guidance.

The lifecycle additionally supports resident beliefs that enter standing task context only after satisfying a utility threshold against the null arm and a separate examination on fresh test cases. Residency can be revoked after repeated examination failures. These checks assess operational usefulness and robustness rather than establish the proposed mechanism. No belief reached residency over the course of the 26-task evaluation, so the main results do not demonstrate a benefit from residency. §B.4 specifies these checks, and Alg. 1 summarizes the complete research loop.

3 EXPERIMENTS

We evaluate BELIEFOPT on GPU kernel optimization as a representative setting for autonomous research. Our experiments examine the quality of the solutions discovered under matched budgets and the conditions under which accumulated beliefs improve subsequent search.

3.1 EXPERIMENTAL SETUP

We use 26 kernels from SOL-ExecBench (Lin et al., 2026) on NVIDIA B200 and measure performance by the geometric mean of latency speedups over the reference implementations. All systems use gpt-5.6-sol and are evaluated under identical task-specific token budgets, calibrated to OpenEvolve's token consumption on each kernel. Final solutions are remeasured on the same machine using the full official workload sets. BELIEFOPT starts with an empty belief store and operates without human intervention. We compare against OpenEvolve (Sharma, 2025), ShinkaEvolve (Lange et al., 2026), and CodeEvolve (Assumpção et al., 2025). Component studies use a six-task subset spanning the benchmark families. §D and §B provide the budget accounting, baseline configurations, and measurement procedures.

Table 2: Comparison on a six-task subset using the same backbone model. Full- and small-budget systems are compared within their respective tiers. Bold marks the best result within each tier and metric.

Tier	System	Tokens	Geo-mean	MoE kernel
Full budget	BELIEFOPT (graph steering)	8.0M	**31.3**	**4.7**
	OpenEvolve	8.1M	26.6	4.4
Small budget	ShinkaEvolve	2.4M	**29.5**	**4.8**
	BELIEFOPT (flat store)	2.9M	26.5	1.9
	CodeEvolve	2.6M	24.5	1.9

3.2 MAIN RESULTS

BELIEFOPT achieves a geometric-mean speedup of $23.94\times$, compared with $19.32\times$ for OpenEvolve, $21.48\times$ for ShinkaEvolve, and $22.65\times$ for CodeEvolve (Tab. 1). These results correspond to improvements of 24%, 11%, and 6%, respectively. Against CodeEvolve, the strongest external baseline in this comparison, BELIEFOPT wins 17 tasks, ties 7, and loses 2 under a $\pm 2\%$ tie band. The advantage is most pronounced in attention workloads. BELIEFOPT leads CodeEvolve on every attention family, including a $1.8\times$ advantage on MLA, while the two systems perform similarly on normalization. The MoE kernel remains challenging, with BELIEFOPT reaching $2.24\times$ against $3.68\times$ for CodeEvolve and $4.41\times$ for OpenEvolve.

The reported paired Wilcoxon tests on task log speedups yield $p = 0.0015$, 0.0008, and 0.0009 against OpenEvolve, ShinkaEvolve, and CodeEvolve, respectively. These task-level comparisons support the aggregate advantage of the complete system in the observed evaluations. The component studies below examine which aspects of belief accumulation may contribute to that advantage.

3.3 BUDGET DEPENDENCE AND ABLATIONS

We examine belief accumulation and structure at different search budgets. **??** in **??** reports the complete component studies, including additional sampler variants, expert initialization, and archive expansion.

When does accumulated knowledge help? At 2.4M tokens on six tasks, four runs of the default configuration achieve $24.1 \pm 3.1\times$, compared with $24.5 \pm 1.5\times$ without the belief store, where deviations are sample standard deviations. The difference is unresolved (Welch test, $p \approx 0.8$). Variation is concentrated on MLA decode and MoE, where individual trajectories may discover a substantially better implementation or exhaust the budget before doing so. Thus, the available repeated runs do not establish a benefit from belief accumulation at this budget.

How does belief structure affect longer searches? Exploratory continuations to 8M tokens reach $26.9\times$ with a flat store, $28.8\times$ with guarded revival, and $31.3\times$ with graph ordering. The graph-ordering run outperforms the approximately 8M-token OpenEvolve reference of $26.6\times$ on all six tasks. These observations suggest that relationships among beliefs become useful as the store develops, although each continuation represents a single trajectory. Budget also changes the observed baseline ranking. At 2.4M to 2.9M tokens, ShinkaEvolve achieves the highest six-task score of $29.5\times$ (Tab. 2). The small-budget and continuation results therefore motivate studying knowledge reuse over the course of an investigation rather than assuming an immediate benefit from adding memory.

Does more information improve research? An expanded experience archive improves one six-task comparison but reaches $23.17\times$ against the default system's $23.94\times$ on all 26 tasks. Likewise, seeding eleven hand-written hints reduces MLA decode performance from $490\times$ to $140\times$ in one matched comparison. These findings show that additional experience or prior advice does not consistently improve the observed search outcomes. They motivate selective use and revision of stored knowledge, while further repetitions are needed to establish the effects of individual initialization choices. **??** reports the remaining sampler and archive comparisons.

Table 3: Division of labor between gpt-5.4-mini engineers and a frontier-model research scientist on six tasks. Geo-mean speedups are remeasured under a common evaluation protocol; MLA decode reports the best design observed during optimization. Attempts counts measured candidates, and planner share is the fraction of tokens spent by the research scientist on plans.

Configuration	Geo-mean	Attempts	Planner share	MLA decode best	Design
Engineers think alone	**13.9**	780	1.3%	**9.0**×	single fused kernel
Research scientist writes the plan	10.8	393	21.5%	3.6×	staged pipeline
Plan + repair until it runs	6.4	282	15.3%	4.7×	repaired pipeline
Research scientist writes the plan, price parity	10.2	**948**	19.0%	46.1×	staged pipeline

4 Analysis of Belief-Guided Research

We analyze the research process behind the aggregate results, focusing on how experience becomes explicit knowledge, how that knowledge guides exploration, and how evaluation shapes the evidence available for revision.

4.1 From Experimental Outcomes to Research Beliefs

The 26-task experiment evaluates 4,340 candidates and produces 92 new beliefs, 5 of them narrower revisions of an existing claim, while the judge merges 24 further proposals into existing entries. The final store holds these 92 beliefs plus one null-control entry per run shard (the evaluation ran as five shards), 97 entries in all, with all but one of the 92 scoped to a particular task and 28 directed relations recorded at minting (20 specialization edges, 5 of them produced by revision, and 8 conflict edges) (Tab. 5). These records show how the scientist consolidates observations into a compact set of comparative claims. For example, a belief about ragged causal prefill predicts that 16-wide KV tiles outperform wider tiles on short contexts because they reduce masked computation. Specifying the condition, alternative, and proposed explanation gives subsequent experiments a concrete claim to revisit.

The store influences search alongside substantial unguided exploration. Of 106 promotions, 30 come from engineers following sampled beliefs and 45 from engineers selecting the null arm; the remainder arise from wild exploration, initial probes, and revival. Among 20 final solutions exceeding the reference, 4 are generated under a sampled belief, 14 under the null arm, and 2 before any belief exists. These counts identify the immediate source of a candidate, but do not isolate its full dependence on prior knowledge because unguided candidates can refine incumbents produced through earlier guided experiments. No belief reaches residency within the evaluation, so the observed results concern the formation, selection, and revision of provisional claims rather than the benefits of resident knowledge.

4.2 How Research Guidance Shapes Exploration

To study the interface between the scientist and engineers, we compare research directions with detailed implementation plans on six tasks using gpt-5.4-mini engineers. The control lets engineers implement sampled directions, while the planning configuration replaces Thompson selection with plans authored by the frontier-model scientist. A third configuration additionally permits up to eight repair attempts. At equal token budgets, planning reduces geometric-mean speedup by 22% and planning with extended repair by 54% (Tab. 3). Plan generation consumes 21.5% of the budget, reducing measured experiments from 780 to 393.

A cost-matched planning run permits 948 experiments but still reaches only $10.2\times$ re-measured, indicating that experiment count alone does not explain the gap. Traces also reveal differences in the explored designs. On MLA decode, a scientist-prescribed staged implementation reaches $3.6\times$, whereas an engineer using the same model independently constructs a fused implementation reaching $9.0\times$ (in-run scores). These observations support retaining implementation freedom in the tested configuration. Since planning also changes selection and context, the comparison concerns the complete guidance interface. §C.3 provides the detailed configurations and repetition diagnostics.

4.3 How Evaluation Shapes Belief Updates

An apparent improvement can influence later research even when it results from incomplete evaluation. In a development run, an MLA decode candidate passes a screen containing the eight smallest workloads but fails an official large-shape check. Size-stratified screening and full-workload promotion subsequently produce a candidate passing all 47 workloads at $442\times$. This case illustrates why the validity of an observation must be resolved before its gain reinforces a belief.

In the main experiment, the evaluator rejects 60 of 166 full-workload evaluations, and these rejections receive zero utility credit. The attribution rule therefore blocks this specific route by which invalid screening gains could influence future belief selection. Additional audits identify omitted tail shapes and infrastructure failures, showing that both test coverage and the distinction between candidate failure and instrument failure affect the learning signal. These cases motivate the evaluation design, while their separate contribution to final performance remains to be established through controlled ablations. §C records the diagnostic evidence.

5 Related Work

Memory and Self-Evolving Agents. Agents learn from experience through reflections, reusable skills, and distilled lessons (Shinn et al., 2023; Wang et al., 2023; Zhao et al., 2024). Recent systems extend this approach with autonomous curation, contrastive revision, and verification before retaining skills (Zhang et al., 2026a; Ouyang et al., 2026; Hu et al., 2026; Zhang et al., 2026b). Structured memories further associate experience with provenance, usage statistics, and supporting or contradicting evidence (Volkov et al., 2026; Qian & Ren, 2026; Huang et al., 2026). However, the benefits of distillation depend on task repetition and evaluation distribution, and stored skills do not consistently outperform raw experience or memory-free baselines (Lei et al., 2026; Deng et al., 2026; Zheng et al., 2026). This motivates mechanisms for revocation, evidence accounting, and selective retention (Zhou et al., 2026; Qi et al., 2026; Zhang et al., 2026c), alongside explicit learning from failures (Wang, 2026). Our work shares this emphasis on revisable knowledge and focuses on its role in autonomous research. BeliefOpt represents lessons as conditional, comparative beliefs with proposed mechanisms, then uses experimental evidence to revise both their scope and their influence on subsequent exploration.

Agents for Autonomous Research. Evolutionary research agents discover improved programs through repeated generation and evaluation (Romera-Paredes et al., 2024; Novikov et al., 2025). OpenEvolve, ShinkaEvolve, and CodeEvolve develop this paradigm through open implementations, adaptive search, and population-based exploration (Sharma, 2025; Lange et al., 2026; Assumpção et al., 2025), while specialized systems apply it to GPU kernel optimization (Liao et al., 2026; Dong et al., 2026; Huang et al., 2026). Adaptive strategy selection provides a complementary direction, exemplified by MetaScale's selection among evolving meta-thoughts (Liu et al., 2026) and grounded more broadly in Thompson sampling, hierarchical transfer, and adaptation to changing rewards (Daniel et al., 2018; Kveton et al., 2021; Raj & Kalyani, 2017). Reliable evaluation is equally necessary, as coding benchmarks remain vulnerable to exploitation and incomplete validation (Wang et al., 2026; Zhao et al., 2026), including in executable kernel settings (Ouyang et al., 2025). BeliefOpt brings these concerns together around the evolution of explicit research beliefs, coupling their selection and revision with an evaluator that expands test coverage. We study GPU kernel optimization as a representative experimental setting for this broader problem of learning to conduct research from accumulated evidence.

6 Conclusion

We set out from an observation about waste, an evolutionary optimizer evaluates thousands of expensive candidates and learns almost nothing that outlives the round, and from an observation about danger, memory systems that write lessons without testing them canonize their own mistakes. BeliefOpt occupies the position between the two. Its only knowledge object is a falsifiable belief; use is the first test and an interventional exam the second; structure steers the search but never touches the statistics; and the evaluator evolves alongside the agent it evaluates. Under a protocol that fixes the model, the per-task budget, the cold start, and the evaluation, the autonomous system

beat the OpenEvolve baseline by 24% across 26 production kernels and beat its own human-assisted predecessor. The controlled experiments along the way were as informative as the headline. Structure multiplies the value of beliefs, untested expertise hurts, blueprints written for a weaker engineering squad make it worse, and an idea that helps on six tasks can fail on 26. Research is the disciplined conversion of experience into belief, and an agent that practices that discipline improves its own research as it works.

AI USE STATEMENT

Generative AI models are integral components of the research agents studied in this work. As described in the paper, these models propose and revise optimization hypotheses, generate and modify candidate implementations, and synthesize experimental feedback into reusable beliefs that guide subsequent experiments. These uses constitute part of the evaluated research methodology. Additionally, we used generative AI tools to assist with reviewing and revising the manuscript, improving clarity and organization, and preparing illustrative figures.

The authors reviewed and edited the AI-assisted text and figures for accuracy and consistency with the proposed method and experimental findings. Candidate implementations generated by the research agents were evaluated through the correctness checks and performance measurements described in the paper. We distinguish model-generated hypotheses and beliefs from experimentally supported findings, and base our empirical claims on the reported evaluations. The authors take responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies research agents that accumulate and revise beliefs from experimental feedback, with evaluation focused on GPU kernel optimization. Such systems may improve research efficiency, but they can also propagate incorrect hypotheses, exploit incomplete evaluation criteria, or produce code that behaves unexpectedly. We therefore treat agent-generated beliefs as provisional and assess candidate implementations through the correctness checks and performance measurements described in the paper. These checks provide evidence within the evaluated setting, rather than guarantees of general correctness or safe deployment. Broader applications should incorporate execution safeguards, resource limits, and human oversight appropriate to the domain, particularly when experiments involve sensitive data or consequential decisions. Improvements in automated optimization may also increase downstream computational use, so efficiency gains should not be interpreted as evidence of reduced overall environmental impact.

REPRODUCIBILITY STATEMENT

The main text describes the agent architecture, belief representation and revision mechanisms, and experiment allocation procedure. The experimental section specifies the benchmark, hardware setting, baselines, evaluation metrics, and token budgets used for comparison. Additional implementation and evaluation details are provided in the appendix. Together, these descriptions support reproduction of the research workflow and assessment of the reported results. Exact trajectories may vary because of stochastic model outputs and hardware-dependent execution times.

REFERENCES

Dhruv Agarwal, Reece Adamson, Andrew McCallum, Peter Clark, Ashish Sabharwal, and Bodhisattwa Prasad Majumder. Evidence-informed llm beliefs for continual scientific discovery. *arXiv preprint arXiv:2606.29182*, 2026.

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

J Russo Daniel, Van Roy Benjamin, Kazerouni Abbas, Osband Ian, and Wen Zheng. A tutorial on thompson sampling. *Foundations and trends® in machine learning*, 11(1):1–99, 2018.

Zihao Deng, Yining Zhu, Leiming Wang, Jingfei Lu, Junbo Wang, Chuncheng Ran, Yu Yang, Dixuan Yang, and Jikun Shen. Finevolvebench: A benchmark for self-evolving agents on low-repetition tasks with implicit rewards. 2026. URL https://arxiv.org/abs/2606.06960.

Kris Shengjun Dong, Sahil Modi, Dima Nikiforov, Sana Damani, Edward Lin, Siva Kumar Sastry Hari, and Christos Kozyrakis. Kernelblaster: Continual cross-task cuda optimization via memory-augmented in-context reinforcement learning. *arXiv preprint arXiv:2602.14293*, 2026.

Wentao Hu, Zhendong Chu, Yiming Zhang, Junda Wu, Ming Jin, Xiangyu Zhao, Yilei Shao, Yanfeng Wang, and Qingsong Wen. Skillbrew: Multi-objective curation of skill banks for llm agents. *arXiv preprint arXiv:2605.29440*, 2026.

Kexin Huang, Ying Jin, Ryan Li, Michael Y Li, Emmanuel Candès, and Jure Leskovec. Automated hypothesis validation with agentic sequential falsifications. *arXiv preprint arXiv:2502.09858*, 2025.

Zixuan Huang, Da Chen, Kecheng Huang, Lihao Yin, Xing Li, Huiling Zhen, Mingxuan Yuan, and Zili Shao. Kernel foundry: A diagnosis-driven evolutionary kernel optimizer with multi-experts. *arXiv preprint arXiv:2605.30359*, 2026.

Branislav Kveton, Mikhail Konobeev, Manzil Zaheer, Chih-wei Hsu, Martin Mladenov, Craig Boutilier, and Csaba Szepesvari. Meta-thompson sampling. In *International Conference on Machine Learning*, pp. 5884–5893. PMLR, 2021.

Robert Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. In *International Conference on Learning Representations*, volume 2026, pp. 74026–74078, 2026.

Yingtie Lei, Zhongwei Wan, Jiankun Zhang, Samiul Alam, Zixuan Zhong, Peizhou Huang, Xin Wang, Jingxuan Zhang, Donghao Zhou, Yunta Hsieh, et al. Skillevolbench: Benchmarking the evolution from episodic experience to procedural skills. *arXiv preprint arXiv:2605.24117*, 2026.

Gang Liao, Hongsen Qin, Ying Wang, Alicia Golden, Michael Kuchnik, Yavuz Yetim, Ruichao Xiao, Jia Jiunn Ang, Chunli Fu, Yihan He, et al. Kernelevolve: Scaling agentic kernel coding for heterogeneous ai accelerators at meta. In *2026 ACM/IEEE 53rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 733–747. IEEE, 2026.

Edward Lin, Sahil Modi, Siva Kumar Sastry Hari, Qijing Huang, Zhifan Ye, Nestor Qin, Fengzhe Zhou, Yuan Zhang, Jingquan Wang, Sana Damani, et al. Sol-execbench: Speed-of-light benchmarking for real-world gpu kernels against hardware limits. *arXiv preprint arXiv:2603.19173*, 2026.

Qin Liu, Wenxuan Zhou, Nan Xu, James Y Huang, Fei Wang, Sheng Zhang, Hoifung Poon, and Muhao Chen. Metascale: Test-time scaling with evolving meta-thoughts. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 11828–11842, 2026.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

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

Anne Ouyang, Simon Guo, Simran Arora, Alex L Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. Kernelbench: Can llms write efficient gpu kernels? *arXiv preprint arXiv:2502.10517*, 2025.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, et al. Skillos: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026.

Yijiashun Qi, Xiang Xu, and Yuxuan Li. When not to write memory: Governing false promotion from correlated agent traces. In *2026 6th International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pp. 228–234. IEEE, 2026.

Yuxi Qian and Yuxiang Ren. Evograph-mem: Failure-aware editable graph memory for long-term language agents. *arXiv preprint arXiv:2608.11248*, 2026.

Vishnu Raj and Sheetal Kalyani. Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*, 2017.

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

Asankhaya Sharma. Openevolve: An open-source implementation of alphaevolve. https://github.com/codelion/openevolve, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. volume 36, pp. 8634–8652, 2023.

Izumi Takahara and Teruyasu Mizoguchi. Toward auditable ai scientists: A hypothesis evolution protocol for llm agents. *arXiv preprint arXiv:2607.09195*, 2026.

Viktor Volkov, Valentin Khrulkov, Andrey V Galichin, Danil Sivtsov, Nikita Glazkov, Olga Volkova, Konstantin Pchelin, Iaroslav Bespalov, Dmitry V Dylov, Petr Anokhin, et al. Evomem: Memory-augmented evolution for code optimization. *arXiv preprint arXiv:2608.10795*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hanchun Wang. Negative knowledge as failure-aware shared memory for autoresearch. *arXiv preprint arXiv:2606.21024*, 2026.

Hao Wang, Hanchen Li, Qiuyang Mang, Alvin Cheung, Koushik Sen, and Dawn Song. Do androids dream of breaking the game? systematically auditing ai agent benchmarks with benchjack. *arXiv preprint arXiv:2605.12673*, 2026.

Genrui Zhang, Erle Zhu, Jinfeng Zhou, Caiyan Jia, and Hongning Wang. Skillevolver: Skill learning as a meta-skill. *arXiv preprint arXiv:2605.10500*, 2026a.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026b.

Xing Zhang, Yanwei Cui, Guanghui Wang, Ziyuan Li, Wei Qiu, Bing Zhu, and Peiyang He. Library drift: Diagnosing and fixing a silent failure mode in self-evolving llm skill libraries. *arXiv preprint arXiv:2605.19576*, 2026c.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Bingchen Zhao, Dhruv Srikanth, Yuxiang Wu, and Zhengyao Jiang. Specbench: Measuring reward hacking in long-horizon coding agents. *arXiv preprint arXiv:2605.21384*, 2026.

Congjie Zheng, Chuanyi Xue, Bin Liang, Jun Yang, and Changshui Zhang. Seagym: An evaluation environment for self-evolving llm agents. *arXiv preprint arXiv:2606.17546*, 2026.

Yan Zhou, Yue Ouyang, Kaiyang Zheng, and Suncheng Xiang. Tepa: Revoking stale memories for conflict-robust language agents. *arXiv preprint arXiv:2608.07429*, 2026.

A PER-TASK RESULTS

Tab. 4 lists every task of the 26-kernel evaluation for all five systems. All values are re-measured on one machine with the full official workload set. Six GEMM tasks for which BELIEFOPT found no candidate clearing the 0.5% full-set margin shipped the reference implementation ($1.00\times$); OpenEvolve shipped subset-screened regressions on the same tasks. Per-task token budgets are the OpenEvolve baseline's ledger (33.4M in total); CodeEvolve, ShinkaEvolve, and BELIEFOPT spent 34.1M, 33.6M, and 33.0M against it.

Table 4: Per-task latency speedup over the reference, 26 SOL-ExecBench kernels, per-task token parity.

#	Task	OpenEvove	CodeEvolve	ShinkaEvolve	Human-assisted	BELIEFOPT
001	fused add RMSNorm h2048	22.17	22.46	22.43	22.64	22.64
002	fused add RMSNorm h4096	26.41	21.26	22.48	26.45	22.51
003	fused add RMSNorm h7168	21.36	20.36	21.27	21.31	21.85
004	GEMM n128 k2048	0.88	1.02	1.30	2.22	1.18
005	GEMM n256 k7168	0.65	0.94	1.01	0.93	0.80
006	GEMM n2048 k4096	0.79	0.97	0.99	1.14	1.00
007	GEMM n4096 k4096	0.86	0.98	0.98	0.90	1.00
008	GEMM n4096 k14336	0.91	0.99	0.99	0.91	1.00
009	GEMM n5120 k2048	0.96	0.97	0.97	0.96	1.00
010	GEMM n6144 k4096	0.93	0.97	0.98	0.93	1.00
011	GEMM n28672 k4096	0.97	1.00	1.00	1.03	1.00
012	GQA paged decode h32 kv4	1869.7	1877.7	2009.9	1981.4	1991.9
013	GQA paged decode h32 kv8	1513.7	1660.2	1512.5	1888.4	1764.0
014	GQA paged prefill causal h32 kv4	4946.9	4918.1	3961.2	5093.1	5252.5
015	GQA paged prefill causal h32 kv8	139.4	3705.1	3328.6	3845.7	3819.0
016	GQA ragged prefill causal h32 kv4	191.7	192.5	168.8	250.1	189.6
017	GQA ragged prefill causal h32 kv8	163.5	196.2	193.8	235.6	210.4
018	MLA paged decode h16	133.7	185.2	198.3	215.0	475.5
019	MLA paged prefill causal h16	722.4	612.1	290.0	779.4	807.1
020	MoE FP8 block-scaled top-8	4.41	3.68	1.80	1.00	2.24
021	RMSNorm h128	19.48	18.20	18.18	20.45	18.67
022	RMSNorm h512	18.21	18.03	18.14	18.51	18.06
023	RMSNorm h1536	18.94	17.91	17.12	18.12	18.91
024	RMSNorm h2048	16.88	15.82	17.26	17.55	17.62
025	RMSNorm h4096	16.91	16.50	16.76	17.25	16.66
026	RMSNorm h7168	14.25	14.95	14.96	15.38	15.09
	Geometric mean	19.32	22.65	21.48	23.65	23.94

B IMPLEMENTATION DETAILS

B.1 EXECUTABLE RESEARCH INSTANTIATION

Each task t has a reference program r_t, an official workload set $\mathcal{W}_t$, a correctness oracle, and a token budget B_t. Let $L(p, \mathcal{W}_t)$ denote the workload latency aggregate returned by the benchmark harness. The task score is

$$s_t(p) = \begin{cases} L(r_t, \mathcal{W}_t)/L(p, \mathcal{W}_t) & \text{if } p \text{ is correct on every workload,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We report the geometric mean of the best score achieved on each task, $[\prod_{t\in\mathcal{T}} \max_p s_t(p)]^{1/|\mathcal{T}|}$. Tasks therefore receive equal weight in log space. The reference initializes the incumbent, and promotion requires a full-workload improvement of at least 0.5%.

We use 26 production inference kernels from SOL-ExecBench in the FlashInfer-Bench collection. They include eight GEMM tasks, three fused add-and-RMSNorm tasks, six RMSNorm tasks, grouped-query attention with paged and ragged inputs, multi-head latent attention, and an FP8 block-scaled mixture-of-experts task. Candidates are Python programs using Triton or PyTorch, compiled and timed on NVIDIA B200 through the benchmark harness. Official workload sets contain 7 to 48 shapes per task. All final solutions are remeasured on one machine using the complete official workloads.

B.2 Posterior Updates and Exploration

Task and global utility models use Normal-Inverse-Gamma distributions with $\sigma^2 \sim \mathrm{InvGamma}(\alpha, \beta)$ and $\mu \mid \sigma^2 \sim \mathcal{N}(m, \sigma^2/\kappa)$. The initial mean precision is $\kappa = 10^{-3}$. Before incorporating an attributed gain δ, the sufficient statistics are discounted using $\lambda = 0.98$. To distinguish the old and updated mean explicitly, write the discounted parameters as

$$\bar{\kappa} = \lambda\kappa, \qquad \bar{\alpha} = 1 + \lambda(\alpha - 1), \qquad \bar{\beta} = \lambda\beta. \tag{3}$$

The conjugate update is then

$$\begin{aligned} \kappa' &= \bar{\kappa} + 1, \quad m' = \frac{\bar{\kappa}m + \delta}{\bar{\kappa} + 1}, \\ \alpha' &= \bar{\alpha} + \tfrac{1}{2}, \quad \beta' = \bar{\beta} + \frac{\bar{\kappa}}{2(\bar{\kappa} + 1)}(\delta - m)^2. \end{aligned} \tag{4}$$

The prior is $m = 0$, $\kappa = 10^{-3}$, $\alpha = \beta = 1$; an attributed gain is the candidate's relative improvement over the incumbent, $\delta = s_t(p)/s_t(p^\star) - 1$, and discounting precedes every conjugate update. A Thompson draw samples the variance and then the utility mean from this posterior. Task and global draws combine through Eq. 1; this additive rule is a transfer heuristic, not a claim of calibrated hierarchical Bayesian inference.

The arm pool contains at most twelve beliefs and a permanent null arm. Each of two main engineers independently selects a direction, with probability 0.2 reserved for the graph-ordered untried queue. An auxiliary slot generates an unguided candidate or, during a stall, restarts from a distinct archived branch. Revival selects lineage-component representatives by code distance from the incumbent. Tasks run sequentially to their individual token quotas. Candidates that fail to execute can receive up to two repair attempts before final evaluation.

B.3 Evaluation and Utility Attribution

A screening subset of eight workloads is stratified by size. Full-workload validation controls promotion, and the initial reference provides a performance floor. Proposed additional tests include new sizes, perturbed inputs, and variants outside the distribution used to form a belief. The reference must pass each test before admission to the retained suite. Instrument failures are quarantined with backoff and a circuit breaker, without entering utility updates, belief extraction, or failure prompts.

Let δ_{subset} and δ_{full} denote the recorded gains relative to the incumbent under the corresponding measurements. Attribution follows the candidate's final evaluation status.

Outcome	Attributed utility
Promoted candidate	$\min(\delta_{\mathrm{subset}}, \delta_{\mathrm{full}})$
Valid non-promoted candidate passing the gate	Measured gain
Promotion-gate rejection	0
Correctness failure	-0.05
Instrument fault	No observation
Wild, archive, or revival candidate	No belief attribution

The null arm uses the same outcome accounting as sampled beliefs. The utility models describe the outcomes of following a direction within the full agent context, not a causal estimate of the truth of its mechanism.

B.4 Curation and Lifecycle

The scientist receives the task definition, candidates and measured scores, accumulated attempt records, and the scope-compatible belief store. The store listing is size-guarded, with truncation recorded and residents retained first. Candidate beliefs specify conditions, a proposed choice, a named alternative, and a mechanism. A semantic judge identifies equivalence, support, specialization, conflict, or novelty. Equivalent claims are merged without creating new arms. Graph operations order untried claims, select ambiguous claims for revision, and prioritize archived branches for revival;

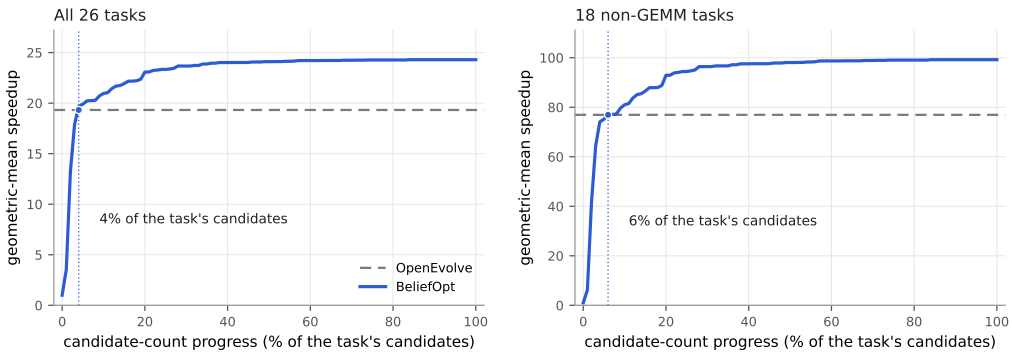

Figure 2: Best-so-far geometric mean speedup of BELIEFOPT against candidate-count progress within each task, with OpenEvolve's final performance as the horizontal line. The left sub-figure shows the result of all 26 tasks, crossed after 4% of each task's candidates. The right sub-figure illustrates the 18 non-GEMM tasks alone, and the crossing moves only to 6%.

edges never contribute utility observations or change posterior parameters directly. Revision targets beliefs with at least four observations, an intermediate posterior estimate, and a recorded failure.

Beliefs begin unsettled. Residency requires $P(\mu_b > \mu_{\text{null}}+0.005) \geq 0.95$ and a separate examination on inputs not used to form the belief. The examination re-measures the incumbent under a fresh evaluation seed as its control and requires a belief-guided candidate generated under that seed to reach at least 0.8 times the incumbent's score on both the public workload set and the held-out measurement, and in every case at least 1.005 times the reference. The check is an absolute performance floor against the incumbent rather than a comparison against an unguided candidate, and it does not validate the proposed mechanism. Two consecutive examination failures revoke residency. Beliefs with utility substantially below the null arm, or identified as measurement exploits, can become negative guidance. No beliefs reached residency during the 26-task evaluation, as discussed in §4.1.

B.5 RESEARCH LOOP

The engineering squad receives selected beliefs and current solutions rather than the raw experiment archive. The research scientist uses the archive to identify repeated outcomes across rounds. This division preserves implementation freedom for engineers while allowing the scientist to consolidate evidence across experiments. §4.2 evaluates the alternative of supplying detailed plans.

C SUPPLEMENTARY EXPERIMENTS

C.1 INTERPRETING SEARCH TRAJECTORIES

Fig. 2 plots the best-so-far geometric mean of BELIEFOPT's solutions against candidate-count progress within each task, with OpenEvolve's final re-measured level as a horizontal line. The horizontal coordinate is the fraction of a task's candidates evaluated so far rather than measured token expenditure, and the vertical values are in-run full-workload gate scores, which run about 1.5% above re-measured values. The level is crossed after about 4% of each task's candidates on all 26 tasks and after 6% on the 18 non-GEMM tasks, so the early crossing is not produced by the GEMM tasks that the full-workload gate holds at the reference from the first round. Both crossings describe the plotted coordinate and are not token-efficiency estimates; the earlier 15% token-axis result was measured for a previous system version and does not quantify the current system. A current-version efficiency comparison requires token-aligned trajectories and a common measurement procedure.

C.2 BELIEF AND PROMOTION RECORDS

Tab. 5 summarizes the store and the immediate generation context of successful candidates. The 92 newly formed beliefs, five of which are revisions, and one null-control entry per run shard account for the 97 final entries; duplicate proposals are merged into existing entries. Of 80 beliefs with utility

Table 5: Belief lifecycle for the 26-task run. Every mechanism above the rule fired continuously.

Event	Count
Candidates measured	4,340
Full-set gate evaluations / promotions	166 / 106
promotions by slot (main / wild / phase-0 / revival)	75 / 20 / 8 / 3
main-slot promotions by draw (belief / null control)	30 / 45
final solutions by draw (belief / null control / before any belief)	4 / 14 / 2
Beliefs minted (novel / specialization / conflict)	92 (64 / 20 / 8)
Duplicates merged by the judge	24
Revisions minted / declined	5 / 1
Revival-pool activations	53
Instrument faults quarantined	4
Utility observations per observed belief (median / mean, 80 beliefs)	4.0 / 5.4

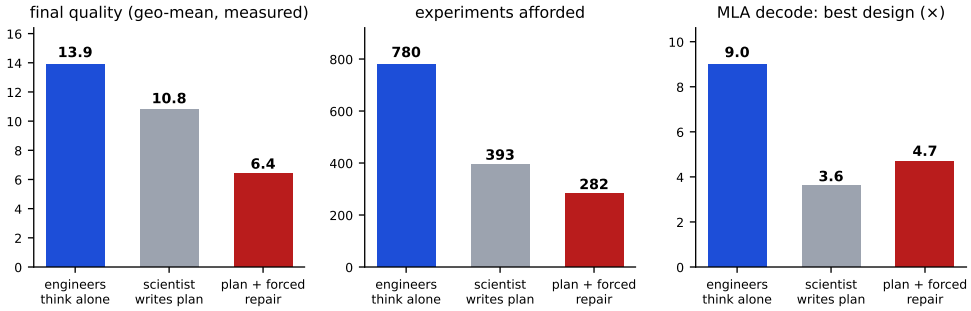

Figure 3: Planning diagnostics with a small model as the engineers. The panels compare final quality, measured experiment counts, and MLA decode designs.

observations, 40 receive at least one positive gain and 3 finish with a positive posterior mean. None reaches residency. These statistics describe utilization and observed utility rather than the correctness of the proposed mechanisms.

C.3 Planning and Repetition Diagnostics

The planning comparison uses gpt-5.4-mini engineers in all arms. The control retains ordinary direction selection. The planned arm replaces this selection with a scientist-generated idea and implementation plan for each main engineer, while the repair variant additionally permits up to eight repair attempts. All configurations retain the wild slot. Tab. 3 reports the token-matched comparison, and Fig. 3 shows the original diagnostics.

Plan generation consumes 21.5% of the token budget, and the approximately 2,500-character plan enlarges engineer prompts by 60%. A subsequent cost-matched evaluation assigns each small-model token one-fifth the cost of a scientist-model token, using the same pricing ratio as the main evaluation. It consumes 6.5M raw tokens, evaluates 948 candidates, and reaches $10.2\times$ re-measured ($10.4\times$ in-run). On MLA decode, the scientist's staged implementation reaches $3.6\times$, while a self-directed engineer finds a fused design reaching $9.0\times$. These observations suggest that both resource allocation and the explored designs contribute to the differences.

Candidate repetition. Under a code-similarity threshold of 0.90, 61% of control attempts resemble an earlier candidate, compared with 25% under planning. Among these repeats, 38% and 35%, respectively, improve on the earlier candidate. Thus, repeated code does not uniformly represent unproductive work.

Near-duplicate filtering. A separate variant rejects proposals with similarity at least 0.92 to a known failed or inferior attempt, exempts incumbent refinements, and permits one informed retry. It reaches $23.1\times$ at 2.4M tokens and $25.1\times$ in an 8M continuation. Across the continuation it requests

143 retries and rejects 16 candidates. The largest performance differences occur on MLA decode and MoE, where interventions are infrequent. The same run also suffers an endpoint-related attribution fault described below. These factors prevent a clean causal interpretation of the single-run difference, and this filter is excluded from the default configuration.

D COMPARISON PROTOCOL

Comparisons between open-ended optimization agents are easy to bias by accident, a stronger model in one role, a larger budget on the hard tasks, a different timing harness. We fixed four axes and report anything we could not fix.

One model. Every system, ours and every baseline, runs the same frontier model (gpt-5.6-sol through Azure OpenAI) in every role. ShinkaEvolve's LLM ensemble and CodeEvolve's model options were collapsed to this single model; our research scientist and engineering squad are the same model. The division-of-labor experiments of §4.2 deliberately break this rule for the engineers and say so.

Budget parity per task. We first ran the OpenEvolve baseline (150 iterations per task) and recorded its token ledger per task, 33.4M tokens in total, ranging from 0.6M on a small RMSNorm to 2.5M on a large prefill kernel. Every other system then received, per task, exactly the tokens the baseline spent on that task, enforced by an isolated ledger that counts every API call in and out and kills the task when the quota is crossed. Budgets are therefore matched task by task rather than only in aggregate, so no system may shift tokens to where it is strong.

Cold start and no human intervention. BELIEFOPT begins each run with an empty store, no seed beliefs, and no hand-written hints. No prompt was edited, no run was nudged, and no solution was touched by a human between launch and the final measurement. The one earlier system of ours that did receive human help during its run is reported as the *human-assisted predecessor*; §B describes it and the interventions it received, and we include it because an autonomous system beating a supervised one on the same tasks is a claim readers can weigh only if the supervision is described.

Baselines. The AlphaEvolve family is represented by OpenEvolve on all 26 tasks. Two further open-source agents were selected as general (not kernel-specific) systems published within the last year. ShinkaEvolve (Lange et al., 2026) was adapted by pointing its LLM clients at our single model and its evaluator at our harness; its embedding-based novelty gate requires an embedding endpoint we did not provide and was therefore inactive, so the numbers we report are for ShinkaEvolve without novelty filtering, a handicap we note wherever it matters. CodeEvolve (Assumpção et al., 2025) ran with four islands and a lock that serializes its evaluations on one GPU. Both were run on the six-task fast lane at 2.4M and 2.6M tokens respectively and on all 26 tasks at the same per-task quotas as BELIEFOPT.

Fast lane. Method development used a six-task subset spanning the families (a fused add-RMSNorm, a GEMM, a GQA decode, an MLA decode, the MoE kernel, and an RMSNorm) at 2.4M to 2.9M tokens per run, with continuations to 8M where a full-budget comparison was needed. Every mechanism in the shipped configuration was admitted or rejected through a controlled A/B on this lane before the 26-task run, and §3.3 reports the ones that lost.