PRIMESCIENTIST: STRATEGIC ALLOCATION OF RESEARCH EFFORT IN AUTONOMOUS RESEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous research agents can propose more directions than available resources allow them to pursue, while each complete research attempt can require substantial implementation and experimentation. We study *strategic research effort allocation*: deciding how to distribute effort across research directions as evidence accumulates and resources are consumed. We introduce PRIMESCIENTIST, which formulates plan construction and execution as a sequential decision problem under a shared inference budget. An executable plan tree preserves competing plans and their outcomes, while an adaptive MCTS-based policy uses experimental feedback and remaining resources to balance exploration and refinement. Evaluations on 27 tasks across AI research, systems and code optimization, and machine learning engineering show better or comparable benchmark-level outcomes with fewer attempts on 26 tasks than AutoResearch. Across the 12 FIRE-Bench AI research tasks, PRIMESCIENTIST improves average reward by 10.3% with 50.6% fewer attempts. Planning and execution both count toward the token budget. Ablations support adapting selection to the remaining budget, making effective resource use an explicit capability of autonomous research agents.

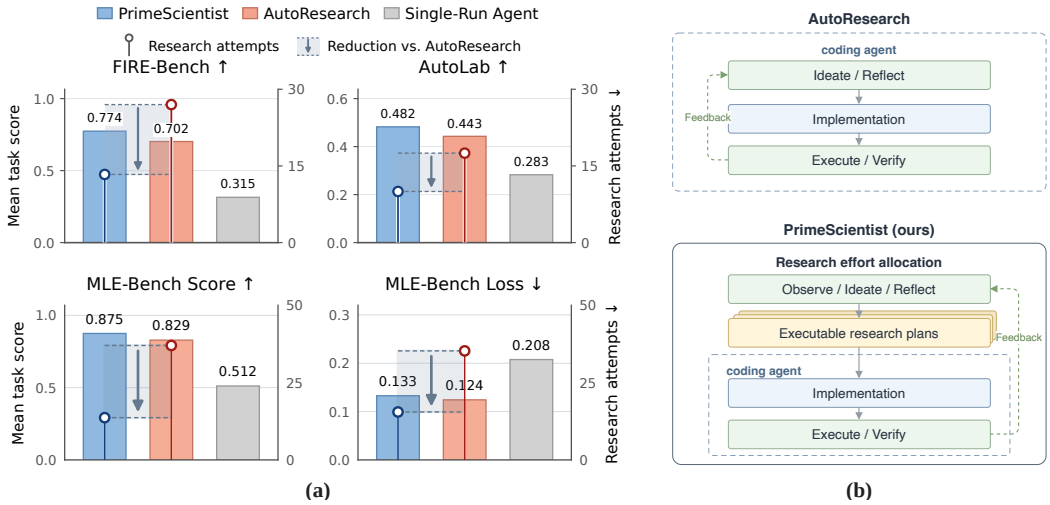

Figure 1: **Strategic research effort allocation.** (a) Benchmark scores and attempts under matched token budgets. Bars show mean scores; stems and shading show mean attempts and reductions. MLE-Bench separates scores from losses. (b) PRIMESCIENTIST uses explicit plans, feedback, and remaining resources to guide research.

1 INTRODUCTION

Autonomous research (using AI agents to propose hypotheses, modify experimental code, run experiments, and decide what to try next) has emerged as a credible regime for both machine-learning engineering and broader scientific work (Karpathy, 2026; Lu et al., 2024; Yamada et al., 2025; Jiang et al., 2025; Yang et al., 2025; Chan et al., 2025). In Karpathy's `autoresearch` (Karpathy, 2026), a single agent reads a Markdown plan, edits a Python training file, runs a fixed-time training experiment, and decides whether to keep or discard the change before the next iteration. Around this core loop,

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 growing line of systems extends the paradigm to ML engineering (Jiang et al., 2025; Yang et al., 2025; Chan et al., 2025), research ideation and experimental workflows (Lu et al., 2024; Yamada et al., 2025; Baek et al., 2025; Schmidgall et al., 2025), biomedicine (Swanson et al., 2025; Gao et al., 2024; 2025; Wang et al., 2026c; Chung et al., 2026), chemistry and materials science (Boiko et al., 2023; Hu et al., 2026c), and industrial R&D (Datar, 2025).

Recent work studies how to use additional computation effectively through inference-time scaling and longer autonomous research loops (Snell et al., 2025; Wang et al., 2025; Xu et al., 2026). Yet agents can generate numerous plausible research ideas and plans, including alternative hypotheses, dataset choices, model families, ablations, debugging strategies, and interpretations of weak evidence (Baek et al., 2025; Schmidgall et al., 2025; Yamada et al., 2025). Every end-to-end research attempt commits substantial agentic effort (e.g., tokens) to one plan and reduces the opportunities available for subsequent research attempts. Progress will depend on how research effort is allocated over time, using both the evidence accumulated so far and the resources that remain (Simon, 1955; Gershman et al., 2015; Lieder & Griffiths, 2020). We therefore study *strategic research effort allocation*, the problem of deciding how autonomous agents should distribute effort across research directions as evidence accumulates. The aim is to make effective resource use a deliberate, adaptable part of the research policy. We view this capability as an essential dimension of intelligence in autonomous research agents.

However, existing autoresearch systems rarely treat strategic effort allocation as an explicit capability to optimize. AutoResearch assigns each new attempt to the incumbent trajectory, so the path already taken largely determines where subsequent effort goes (Karpathy, 2026). Tree- and population-based systems expose multiple alternatives, yet typically navigate them using a fixed search rule or an LLM judgment that does not explicitly represent the remaining research budget (Jiang et al., 2025; Yamada et al., 2025; Chi et al., 2024; Toledo et al., 2025). R&D-Agent selects ideas before implementation and adapts planning to the remaining wall-clock time (Yang et al., 2025). Closer to our setting, AlphaLab guides an LLM Strategist using the remaining experiment count, while FML-Bench's AdaptiveSearch switches once from greedy refinement to multi-branch exploration when progress stalls (Hogan et al., 2026; Zou et al., 2026). The remaining challenge is to coordinate how research plans evolve and how effort is distributed across them. This calls for a unified policy that uses experimental evidence and remaining resources to guide both planning and execution (Appendix A, Table 5).

To address this challenge, we introduce PRIMESCIENTIST to equip autonomous research agents with a policy for directing their research effort. We formulate strategic effort allocation as a sequential decision problem that jointly models plan construction, execution, and their resource costs. To realize this idea, we introduce an executable plan tree that makes competing plans available for selection before execution. A reflector drafts and revises plans from the evidence accumulated in the tree, while a coding-agent executor carries out each selected plan through coding, debugging, experimentation, and result analysis. Building on this representation, we propose an adaptive MCTS-based allocation policy (Kocsis & Szepesvári, 2006; Browne et al., 2012) that uses experimental feedback and remaining resources to balance exploration and exploitation across research directions. Even with the same observed outcomes, different remaining resources can warrant different allocations.

We evaluate PRIMESCIENTIST on 27 tasks across AI research, systems and code optimization, and machine learning engineering (Wang et al., 2026b; Xu et al., 2026; Chan et al., 2025). PRIMESCIENTIST achieves better or comparable benchmark-level research outcomes with fewer research attempts than AutoResearch on 26 of 27 tasks under the same resource budget. Research sample efficiency measures the task quality obtained relative to the number of complete research attempts. Each attempt requires implementation, experimentation, and analysis to test a research direction. Fewer attempts at comparable quality reduce this experimental demand. The resource budget includes planning and execution; Section 4.1 specifies the units and limits. Our ablation studies show that adapting exploration to the remaining budget improves overall reward over *UCT*, *Greedy*, *Random*, and fixed-exponent alternatives. These findings support making strategic effort allocation a core capability of autonomous research agents. Optimizing this capability could enable agents to improve their own research strategies and convert growing computational resources into scientific breakthroughs at scale.

2 RELATED WORK

Autonomous research and structured search. AI Scientist, ResearchAgent, and Agent Laboratory automate ideation, implementation, and reporting through iterative workflows (Lu et al., 2024; Baek

et al., 2025; Schmidgall et al., 2025). Metric-driven systems range from AutoResearch's linear edit-run-keep-or-revert loop to AIDE's search over code solutions (Karpathy, 2026; Jiang et al., 2025). AI Scientist-v2 uses progressive tree search, AIRA compares greedy, MCTS, and evolutionary policies, and GEAR maintains populations through mutation and crossover (Yamada et al., 2025; Toledo et al., 2025; Jeddi et al., 2026). These systems establish feedback and persistent alternatives as building blocks. PRIMESCIENTIST focuses on how experimental evidence and remaining resources jointly determine which plans receive further effort.

Budget-aware planning and allocation. MARS introduces cost-constrained MCTS for automated AI research, backpropagating rewards that combine validation performance and candidate execution time (Chen et al., 2026). AlphaLab's Strategist revises experiment queues and uses the remaining experiment count to shift toward refinement (Hogan et al., 2026). FML-Bench's AdaptiveSearch begins greedily, then makes a one-time switch to multi-branch exploration after stagnation, with branching determined by the remaining step budget (Zou et al., 2026). These latter approaches adapt trial selection without including planning resources in the allocation objective. At finer granularity, BATS tracks remaining tool calls, while BAVT uses the remaining-resource ratio to sharpen value-based selection within a reasoning tree (Liu et al., 2025; Li et al., 2026). Our selection rule draws on this principle, but allocates complete research attempts and plan construction under a shared token budget. Selecting a tree node determines both the direction and whether to construct alternatives or execute a plan.

Resource-rational and self-improving agents. Bounded rationality and rational metareasoning treat computation itself as a resource-allocation decision (Simon, 1955; Gershman et al., 2015; Lieder & Griffiths, 2020; De Sabbata et al., 2024). Reflection and self-evolution methods revise prompts, behaviors, or agent implementations from feedback (Shinn et al., 2023; Agrawal et al., 2026; Zhang et al., 2025; Qu & Lu, 2026). PRIMESCIENTIST instead retains the reflector and executor while adapting effort across task-level plans. Appendix A gives the broader discussion and a comparison of representative research agents (Table 5).

3 PRIMESCIENTIST

3.1 THE STRATEGIC RESEARCH EFFORT ALLOCATION PROBLEM

Consider a research task x, a plan-construction operator $\mathcal{G}$, a coding-agent executor $\mathcal{A}$, a task evaluator h, and a resource budget $B > 0$. The operator $\mathcal{G}$ produces executable plans from the task or revises existing plans using accumulated evidence. The executor carries a selected plan through coding, debugging, experimentation, and analysis, after which h supplies its empirical outcome. Both operations consume resources; their outputs and realized costs may be unknown before completion.

Definition 1 (Strategic research effort allocation) Let $\mathcal{P}_t$ denote the available research plans and $\mathcal{H}_t$ the record of proposals, execution outcomes, and charged costs before decision t. The state comprises these plans, their history, and the remaining resources:

$$X_t = (x, \mathcal{P}_t, \mathcal{H}_t, B - U_t), \qquad U_t = \sum_{i<t} c_i, \tag{1}$$

with $\mathcal{P}_0 = \mathcal{H}_0 = \varnothing$ and $U_0 = 0$. A policy $\pi(\cdot \mid X_t)$ chooses an action a_t: construct or revise plans through $\mathcal{G}$, execute a plan $s \in \mathcal{P}_t$ through $\mathcal{A}$, or stop. A construction action can expand $\mathcal{P}_t$; an execution supplies empirical evidence; either updates $\mathcal{H}_t$ and incurs cost c_t. Given a terminal research-value functional $\mathcal{V}$, the objective is

$$\max_{\pi \in \Pi_B} J_B(\pi), \qquad J_B(\pi) = \mathbb{E}_\pi[\mathcal{V}(\mathcal{H}_{\tau_\pi})], \tag{2}$$

where τ_π is the stopping time and Π_B contains policies that use only observed history and initiate research actions only while $U_t < B$. Actions are charged on completion, and no further action starts once the budget is reached; the final action may therefore cross the threshold. The expectation accounts for variability in plan construction and execution.

The central coupling is that research alternatives must themselves be produced through research effort. Constructing a plan changes what can be tested, while executing one changes the evidence available for subsequent decisions; both consume the research budget. The allocation policy jointly chooses which research direction to pursue and whether to construct a plan or execute one.

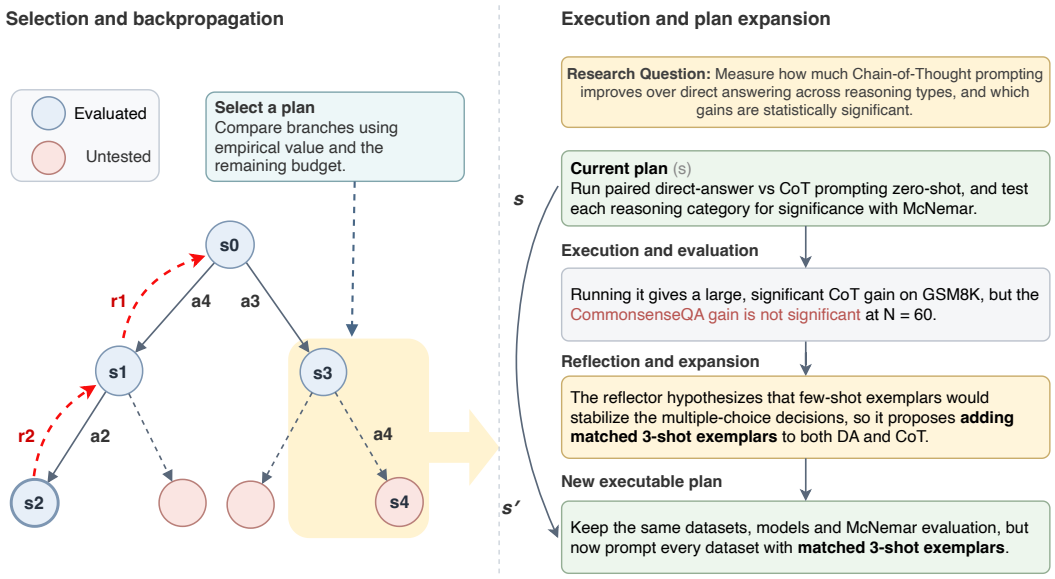

Figure 2: **The PRIMESCIENTIST allocation loop.** Left: selection uses branch values and remaining resources to choose whether to execute a plan or develop alternatives. Right: execution feedback motivates a revised plan. Planning and execution share one token budget.

Research sample efficiency. Research sample efficiency measures the quality of the outcome obtained relative to the number of complete research attempts. We use the best observed task reward as terminal value, $\mathcal{V}(\mathcal{H}_\tau) = \max_{i \in \mathcal{I}_\tau} y_i$, where $\mathcal{I}_\tau$ indexes completed attempts and $y_i \geq 0$ is a reward scaled approximately to $[0, 1]$ (Section 4.1); the value is zero if no attempt has been evaluated. The attempt count is $N_\tau = |\mathcal{I}_\tau|$, counting each executor invocation separately. Comparable value with fewer attempts means that fewer complete cycles of implementation, experimentation, and analysis are needed to obtain the outcome. This matters when testing a research direction requires model training or repeated benchmarking. We assess this tradeoff between outcomes and effort under the same inference budget B, with both planning and execution tokens counted in U_τ. The comparison therefore tests whether resources devoted to choosing research directions translate into more productive research attempts.

3.2 EXECUTABLE PLAN TREE FOR EFFORT ALLOCATION

Research effort is directed by choices of hypothesis, experimental design, and implementation strategy. These choices must remain accessible across attempts so that an agent can revisit an earlier direction or develop a distinct alternative. We introduce an executable plan tree T_t to make these choices explicit and assign effort to complete research plans.

Each node stores a plan, its recorded outcomes, and branch statistics. A reflector constructs these plans; a coding agent executes them. Each plan is a structured Markdown document specifying a research question or proposed change and its implementation strategy (Appendix H). An edge $s \to s'$ records a reflector-generated modification of the parent's research strategy. Each modification is stored as an executable `diff.py` together with a description and prior estimate, allowing the child plan to be reconstructed from its parent. Recording the modification alongside parent and child outcomes makes the development of a research strategy traceable and gives the reflector concrete evidence for its next revision. Because a plan is stored independently of the executor's conversation, a candidate direction can remain available while another branch is evaluated. For each node s, $h(s)$ denotes the reward recorded under §4.1; repeated executions may yield different outcomes. Figure 2 (left) shows evaluated and untested plans as distinct nodes.

The tree exposes two research actions: execute an untested plan or expand an evaluated plan through reflection. Node selection therefore determines both the research direction and whether effort develops an alternative or tests one empirically. The tree instantiates the plans and evidence in the

allocation state X_t, while cumulative token use determines its remaining resources. The executor receives the plan without instructions about the remaining global budget. The reflector can inspect run metadata but receives no additional allocation rule.

The budget covers the input and output tokens of both agents, making plan construction compete with execution for the same inference resources.

$$U_t = C_t^{\text{exec}} + C_t^{\text{refl}}. \tag{3}$$

Here B measures inference tokens, excluding elapsed time and CPU/GPU hours. Costs convertible to a common unit, such as dollars, fit the same budget formulation; allocation under separate resource constraints remains future work.

3.3 Adaptive MCTS-based Effort Allocation

Exploring a new direction produces evidence that can improve later research decisions, while refining a promising plan can improve the current result. The balance between these uses of research effort depends on both observed outcomes and the resources available to act on further evidence. We implement this allocation policy within the Monte Carlo tree search (MCTS) framework (Kocsis & Szepesvári, 2006; Browne et al., 2012).

Selection. An attempt made early can supply evidence for subsequent allocations; as resources are consumed, fewer opportunities remain to act on that evidence. The *Select a plan* step in Figure 2 uses the remaining-budget ratio to adjust how strongly selection favors high-value branches (Li et al., 2026).

Selection traverses the tree to an eligible node. Starting from the root, the policy samples a child v of node u with probability proportional to

$$w_t(v) = \begin{cases} Q(v)^{\alpha_t} & v \text{ has been evaluated,} \\ \left(Q(u)\sqrt{P(v)}\right)^{\alpha_t} & v \text{ is unvisited.} \end{cases} \tag{4}$$

Here $Q(v)$ denotes empirical branch value and $P(v) \in [0, 1]$ the reflector's prior for an unvisited child. Until a child is evaluated, its weight combines this prior with the parent value $Q(u)$. If $Q(u) = 0$, the implementation uses the prior alone, with a positive floor on sampling weights.

Selection concentration is controlled by

$$\alpha_t = \min\left(\frac{1}{r_t}, \alpha_{\max}\right), \qquad r_t = \frac{B - U_t}{B}, \tag{5}$$

with $\alpha_{\max} = 10$ in the reported experiments. With most resources remaining, selection stays comparatively dispersed. As resources are consumed, larger exponents concentrate selection on higher-value branches. Thus, even with the same observed branch values, different remaining budgets induce different allocation distributions. Section 4.3 compares allocation policies with the executor, reflector, tree representation, and evaluation procedure held fixed.

Expansion. The reflector first drafts a root plan from the task instruction. For an evaluated node, it reads the recorded reward, execution evidence, repository state, and relevant tree context, then proposes up to m child plans. The prompt requests substantive alternatives in hypothesis, implementation strategy, or experimental design. Figure 2 (right) shows how the observed outcome motivates a child plan with matched few-shot examples, creating a new direction for subsequent allocation. The child plans enter the tree as separate alternatives for subsequent selection. Appendix I provides the complete prompt, including the required structure of executable plan modifications.

Execution and evaluation. For an untested node s, the executor implements the plan, runs experiments, and analyzes the results. The evaluator supplies $h(s)$ under §4.1. Task execution provides the evaluation signal directly, without a separate simulation rollout.

Backpropagation. After applying the pruning rule, retained outcomes update values along the selected path. The empirical branch value is

$$Q(v) = \frac{1}{|D(v)|} \sum_{s \in D(v)} h(s), \tag{6}$$

where $D(v)$ contains retained evaluated descendants of v, including v itself once evaluated. These updates let subsequent selection use evidence from completed research attempts. Algorithm 1 gives the full loop, including pruning and stopping.

Together, the plan tree and search policy let PRIMESCIENTIST revise both its plans and the allocation of effort across them. Value backpropagation pools evidence across related plans; strategic effort planning adjusts their selection probabilities as resources are consumed. Effective resource use is thus part of the research policy, linking each attempt to the overall research objective.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks and metrics. We evaluate 27 tasks: 12 FIRE-Bench AI research tasks, eight AutoLab systems-optimization and puzzle tasks, and seven MLE-Bench machine-learning competitions (Wang et al., 2026b; Xu et al., 2026; Chan et al., 2025). FIRE-Bench measures rediscovery of established findings using claim-level F1 from a RAGChecker-based evaluator (Ru et al., 2024). AutoLab tests optimization under correctness constraints; MLE-Bench uses task-specific Kaggle metrics. We report the best task score and number of research attempts (*#Attempts*). An attempt is one complete executor invocation, including implementation, experimentation, and analysis; repeating a plan counts again. MLE-Bench counts include only attempts producing valid task scores. Appendix G describes all tasks.

For MLE-Bench search, we orient metrics so higher is better, then linearly map the historical leaderboard's 10th and 90th percentiles to 0 and 1. This places rewards roughly in $[0, 1]$ for a consistent child-selection scale. The mapping is not clipped; valid rewards are floored at 10^{-6} and invalid submissions receive zero. Table 3 reports *original, unnormalized scores* and metric directions.

Implementation and budgets. Both agents use GPT-5 through the Codex CLI with a shared 1.5M-token budget per task, counting all input and output tokens for planning and execution. The primary results use one full search per configuration. Attempt limits are 25 for AutoLab and 30 for FIRE-Bench; MLE-Bench has no additional limit. On FIRE-Bench, each plan is executed twice and the higher score retained, with both executions charged to tokens and attempts. Search uses $m = 3$ child proposals, $\alpha_{\max} = 10$, and pruning threshold $\delta = 0.05$. A child is pruned if its reward is more than δ below its parent's current $Q > 0$; reflection can propose further variants when all children are pruned. Comparisons match token budgets, not elapsed time or CPU/GPU hours. Additional implementation details are in Appendix B.

Baselines. All controlled comparisons use the same coding-agent backbone. *Single-run Agent* executes once on the initial workspace without reflection. *AutoResearch* iteratively edits, runs, and keeps or reverts changes along one trajectory, without reflector-proposed branching or tree pruning (Karpathy, 2026). Its comparison with PRIMESCIENTIST includes the latter's reflection overhead. Policy ablations retain the executor, reflector, tree, pruning, and accounting, changing only plan selection. The AutoLab table also includes the benchmark's published baseline as an external reference.

4.2 MAIN RESULTS

Higher AI research quality with fewer attempts. On FIRE-Bench (Table 2), PRIMESCIENTIST obtains an average reward of 0.7738, compared with 0.7018 for AutoResearch, while reducing research attempts from 27.0 to 13.33. Across the twelve tasks, this corresponds to 10.3% higher average reward with 50.6% fewer attempts. Rewards improve on six tasks and match on two, with fewer attempts on all twelve. The gains extend beyond *MCQ Selection Bias*, the largest individual improvement. Excluding that task leaves mean rewards of 0.7533 versus 0.7266. On *LLM Value Consistency*,

Table 1. **Systems and code optimization on AutoLab.** PRIMESCIENTIST achieves higher overall reward with fewer research attempts than AutoResearch. Each configuration uses one GPT-5 search with a 1.5M-token budget. The AutoLab columns give the benchmark's published baseline.

	PRIMESCIENTIST		AutoResearch		Single-run Agent		AutoLab	
Task	**Max reward**	**#Attempts**	**Max reward**	**#Attempts**	**Max reward**	**#Attempts**	**Max reward**	**#Attempts**
Gaussian Blur	0.5122	10	0.5165	20	0.0000	1	0.1376	1
Hash Join	0.6306	12	0.6376	16	0.6019	1	0.5484	1
Concurrent KV WAL	0.6335	10	0.5946	18	0.4844	1	0.4824	1
Flash Attention	0.3572	11	0.3781	16	0.2029	1	0.2240	1
FFT (Rust)	0.5192	14	0.5317	25	0.5002	1	0.4444	1
VLIW Scheduler	0.5350	9	0.5350	25	0.4620	1	0.5965	1
Smallest Game Player	0.3224	8	0.0000	7	0.0000	1	0.0000	1
Sha256 Throughput	0.3490	6	0.3491	13	0.0106	1	0.0000	1
Average	0.4824	10.0	0.4428	17.5	0.2828	1	0.3042	1

Table 2. **AI research on FIRE-Bench.** PRIMESCIENTIST achieves higher research quality with about half as many attempts as AutoResearch. Results report best F1 from one GPT-5 search per configuration (1.5M tokens). *#Attempts* includes both executions per PRIMESCIENTIST plan.

	PRIMESCIENTIST		AutoResearch		Single-run Agent	
Task	**Max reward**	**#Attempts**	**Max reward**	**#Attempts**	**Max reward**	**#Attempts**
Activation Control	0.3810	10	0.4400	24	0.2220	1
SECA Hallucination	0.6980	16	0.6320	30	0.6320	1
QuestBench	0.6820	16	0.7500	25	0.4000	1
Learning Order Agreement	0.8570	12	0.8570	30	0.8000	1
LLM Value Consistency	0.8890	14	0.7270	30	0.3330	1
To CoT or not to CoT	0.8570	16	0.7740	27	0.6000	1
Grokking or Not	0.4900	14	0.5000	23	0.0000	1
Max Suppression	0.7740	16	0.7060	25	0.0000	1
Neural Collapse Losses	1.0000	4	1.0000	25	0.0000	1
CoT Faithfulness Gaps	0.8890	16	0.7500	29	0.0000	1
Lost in The Middle	0.7690	14	0.8570	30	0.6450	1
MCQ Selection Bias	1.0000	12	0.4290	26	0.1430	1
Average	0.7738	13.33	0.7018	27.0	0.3146	1

for instance, reward rises from 0.727 to 0.889 with 14 versus 30 attempts. These results support strategic allocation as a means to improve research quality and sample efficiency together.

Comparable optimization quality with fewer research attempts. On AutoLab, PRIMESCIENTIST achieves an average reward of 0.4824, compared with 0.4428 for AutoResearch, using 10.0 versus 17.5 research attempts (Table 1). It obtains a nonzero reward on the accuracy-gated *Smallest Game Player* task. On the remaining seven tasks, average rewards are comparable (0.5052 versus 0.5061), with fewer attempts on every task. The benefit therefore includes preserving optimization quality while reducing experimentation, as well as finding stronger solutions. On *Concurrent KV WAL*, reward improves from 0.5946 to 0.6335 with 10 versus 18 attempts. These results extend strategic allocation to implementation choices tested under correctness constraints.

Sample efficiency extends to machine learning engineering. On MLE-Bench (Table 3), PRIMESCIENTIST leads on four of the seven competitions: *APTOS 2019 Blindness*, *NOMAD 2018*, *Plant Pathology 2020*, and *Denoising Dirty Documents*. AutoResearch leads on *Spooky Author ID*, *Leaf Classification*, and *Histopathologic Cancer*. Across all seven, PRIMESCIENTIST uses fewer research attempts with valid task scores, using 10 to 19 per task, compared with 27 to 44. These results extend the sample-efficiency benefit to research attempts involving data processing, training, and validation.

Strategic allocation makes research attempts more productive. Across the three benchmarks, PRIMESCIENTIST uses fewer research attempts on 26 of 27 tasks and achieves higher or comparable benchmark-level performance. Planning and execution both count toward the token budget, so the comparison includes the inference spent on choosing research directions. Taken together, the results show that allocating part of the research effort to planning can preserve or improve outcomes while reducing the number of attempts needed to test research directions. Section 4.4 examines whether

Table 3. **Machine learning engineering on MLE-Bench.** PRIMESCIENTIST achieves competitive scores with fewer research attempts across seven competitions. Best scores come from one GPT-5 search per configuration (1.5M tokens). Scores use the original competition metrics and directions; search uses higher-is-better rewards scaled approximately to $[0, 1]$ (Section 4.1). *#Attempts* counts research attempts that return a valid task score.

	PRIMESCIENTIST		AutoResearch		Single-run Agent	
Task	**Best score**	**#Attempts**	**Best score**	**#Attempts**	**Best score**	**#Attempts**
APTOS 2019 Blindness (↑)	0.81712	11	0.69838	35	0.00000	1
NOMAD 2018 (↓)	0.05856	19	0.06090	40	0.06087	1
Plant Pathology 2020 (↑)	0.88741	16	0.81934	35	0.66423	1
Spooky Author ID (↓)	0.35347	19	0.33788	30	0.57038	1
Leaf Classification (↓)	0.06244	10	0.04086	27	0.12444	1
Histopathologic Cancer (↑)	0.92028	14	0.96816	41	0.87044	1
Denoising Dirty Documents (↓)	0.05652	14	0.05761	44	0.07499	1

Table 4. **Allocation-policy summaries from two separate experiments.** Entries are mean best rewards across tasks. Models, budgets, task sets, and repetitions differ between panels; means should not be compared across panels. Full results are in Tables 6 and 7.

GPT-5: 8 tasks, 1.5M tokens, one search per policy/task				
Policy	Adaptive	UCT	Greedy	Random
Mean reward	**0.598**	0.551	0.563	0.546
GPT-5-mini: 6 tasks, 1M tokens, 3 adaptive / 2 fixed searches				
Policy	Adaptive	Fixed $\alpha = 10$	Fixed $\alpha = 3$	Random $\alpha = 0$
Mean reward	**0.537**	0.507	0.452	0.502

this benefit persists across independent searches. Appendices F.1, F.2, and F.4 provide the token breakdown, reward comparisons at matched resources, and elapsed search times.

4.3 EFFECT OF BUDGET ADAPTIVITY

Table 4 summarizes two separate policy experiments. The first tests plan selection while holding the executor, reflector, tree representation, pruning, and accounting fixed. The second isolates budget adaptivity by retaining the sampling rule but replacing its changing exponent with a constant. With GPT-5 on eight tasks (four FIRE-Bench, four AutoLab), adaptive allocation outperforms UCT (Kocsis & Szepesvári, 2006) with $c = \sqrt{2}$, deterministic Greedy, and uniform Random in mean reward, exceeding UCT on seven tasks and matching it on one. Random leads on *SECA Hallucination* and *Gaussian Blur*, showing that the best policy can differ across tasks.

With GPT-5-mini on six tasks (three per benchmark), we compare adaptive selection with fixed exponents using a 1M-token budget and three adaptive versus two fixed-policy searches per task. Adaptive selection has the highest mean reward and exceeds fixed $\alpha = 3$ on all six tasks. Finite $\alpha = 10$ sampling is distinct from deterministic Greedy. Adaptive selection leads on three tasks and never ranks last; uniform sampling leads on *QuestBench*, while $\alpha = 10$ leads on *LLM Value Consistency*. These results support adjusting exploration without choosing a separate constant for each task. Task-level means, variation, and exceptions appear in Appendix B.2.

4.4 CONSISTENCY AND RESEARCH SAMPLE EFFICIENCY

Across three independent GPT-5-mini searches per method and AutoLab task with 1M-token budgets, mean rewards are comparable (0.4363 versus 0.4359 across six tasks), while mean attempts fall from 28.4 to 21.6 (approximately 24%). Average attempts are lower on every task. On *Concurrent KV WAL*, reward improves from 0.5608 to 0.5830 while attempts fall from 25.7 to 13.0. On *Hash Join*, *Flash Attention*, and *FFT (Rust)*, comparable reward is achieved with fewer attempts.

The accuracy-gated *Smallest Game Player* is evaluated separately: mean maximum accuracy is 0.926 versus 0.918, using 12.3 versus 22.0 attempts. Accuracy exposes progress below the 0.95

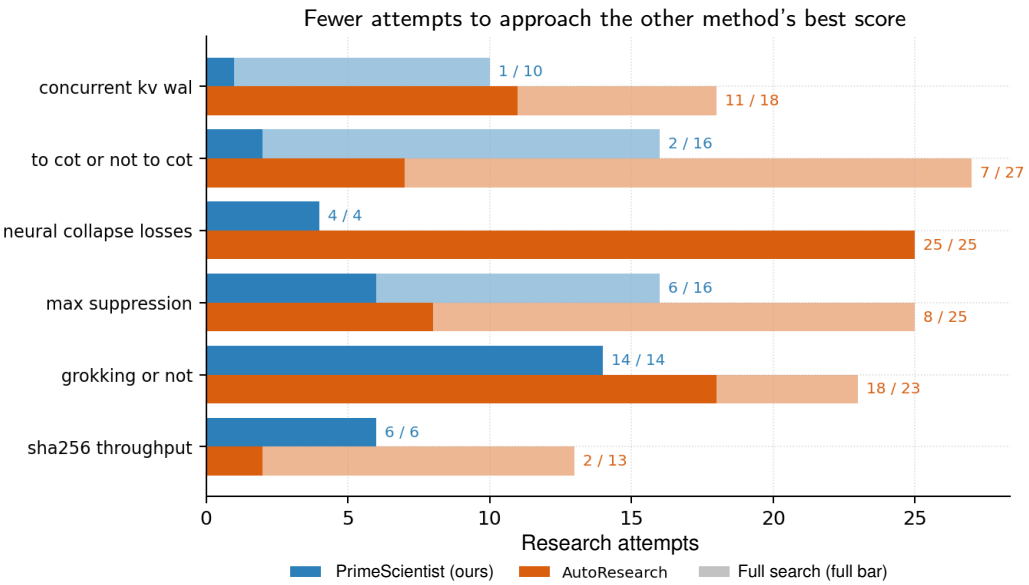

Figure 3: **Fewer attempts to approach the other method's best score.** Inner bars show research attempts needed to reach at least 90% of the other method's best score; full bars show total attempts. PRIMESCIENTIST reaches its reference score with fewer attempts on five of six tasks.

reward threshold, where gated rewards remain zero. Appendix B.3 reports per-task variability and the separately evaluated accuracy-gated task. These are descriptive comparisons, not significance tests.

Figure 3 measures attempts needed to reach 90% of the competitor's best score. PRIMESCIENTIST reaches its corresponding threshold in fewer attempts on five of six tasks. This comparison measures the experimentation needed to approach the other method's final research quality, complementing the total attempt counts reported above. Appendix B.4 traces a single search's quality versus input tokens.

Interpreting allocation decisions. The *Learning Order Agreement* case study illustrates how these choices shape experiments (Appendix B.5). Starting from a CIFAR-10 CNN experiment (F1 0.689), the reflector develops label-noise and architecture branches. A lower-scoring Mixup extension is pruned; patch shuffling and Fourier phase scrambling on the architecture branch reach F1 0.857, while an optimizer and batch-size comparison remains unexecuted. Both methods reach F1 0.857, but PRIMESCIENTIST uses 12 attempts versus 30 for AutoResearch. The tree makes effort allocation visible through alternatives developed, pruned, or left untested.

5 CONCLUSION

In this paper, we study strategic research effort allocation in autonomous research. We introduce PRIMESCIENTIST to make strategic effort allocation an explicit part of how autonomous agents conduct research. To the best of our knowledge, we provide the first formulation that jointly models plan construction and complete research attempts under a shared inference budget. PRIMESCIENTIST's executable plan tree connects research attempts through retained alternatives and empirical outcomes, enabling adaptive MCTS-based allocation across research directions.

PRIMESCIENTIST achieves better or comparable research outcomes with fewer research attempts than AutoResearch under the same resource budget. Our ablation studies show that adapting exploration to the remaining budget improves overall reward over fixed exploration policies. Research-agent evaluation should therefore consider achieved outcomes, research attempts, and total inference costs jointly, including the resources consumed in deciding which research direction to pursue next. A longer-term direction is to learn allocation policies that improve how research agents use their resources, turning advances in AI and computation into scientific discovery at scale.

AI USE STATEMENT

Generative AI is a component of the research systems evaluated in this work. As described in Sections 3 and 4.1, GPT-5 and GPT-5-mini serve as the reflector and executor to generate and revise research plans and task-level hypotheses, implement and debug experimental code, process benchmark data, conduct experiments, and analyze their outputs. The reported outcomes are assessed using the benchmark-specific evaluation procedures and resource accounting described in Section 4.1.

We also used generative AI assistance to revise and restructure the manuscript, update LaTeX tables and figures from recorded experimental results, and adapt the manuscript to the conference template. Automated checks of these manuscript changes included LaTeX compilation, cross-reference validation, and numerical consistency checks against the supplied results. The authors take responsibility for the final text, claims, results, and artifacts, including content produced with generative AI assistance.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2507.19457v2.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. ResearchAgent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6709–6738, 2025. doi: 10.18653/v1/2025.naacl-long.342. URL https://aclanthology.org/2025.naacl-long.342/.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023. doi: 10.1038/s41586-023-06792-0. URL https://www.nature.com/articles/s41586-023-06792-0.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810. URL https://doi.org/10.1109/TCIAIG.2012.2186810.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.07095v6.

Jiefeng Chen, Bhavana Dalvi Mishra, Jaehyun Nam, Rui Meng, Tomas Pfister, and Jinsung Yoon. MARS: Modular agent with reflective search for automated AI research. *arXiv preprint arXiv:2602.02660*, 2026. URL https://arxiv.org/abs/2602.02660v3.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025. URL https://arxiv.org/abs/2505.05410.

Yizhou Chi, Yizhang Lin, Sirui Hong, Duyi Pan, Yaying Fei, Guanghao Mei, Bangbang Liu, Tianqi Pang, Jacky Kwok, Ceyao Zhang, Bang Liu, and Chenglin Wu. SELA: Tree-search enhanced LLM agents for automated machine learning. *arXiv preprint arXiv:2410.17238*, 2024. URL https://arxiv.org/abs/2410.17238v1.

H. Kay Chung, Cong Liu, Anamika Battu, Alexander N. Jambor, Brandon M. Pratt, Fucong Xie, Brian P. Riesenberg, Eduardo Casillas, Ming Sun, Elisa Landoni, Yanpei Li, Qidang Ye, Daniel Joo, Jarred Green, Zaid Syed, Nolan J. Brown, Matthew Smith, Shixin Ma, Shirong Tan, Brent Chick, Victoria Tripple, Z. Audrey Wang, Jun Wang, Bryan Mcdonald, Peixiang He, Qiyuan Yang, Timothy Chen, Siva Karthik Varanasi, Michael A. LaPorta, Thomas H. Mann, Dan Chen, Filipe Hoffmann, Josephine Ho, Jennifer Modliszewski, April Williams, Yusha Liu, Zhen Wang, Jieyuan Liu, Yiming Gao, Zhiting Hu, Ukrae H. Cho, Longwei Liu, Yingxiao Wang, Diana C. Hargreaves, Gianpietro Dotti, Barbara Savoldo, Jessica E. Thaxton, J. Justin Milner, Susan M. Kaech, and Wei Wang. Atlas-guided discovery of transcription factors for T cell programming.

Nature, 651(8107):1077–1087, 2026. doi: 10.1038/s41586-025-09989-7. URL https://www.nature.com/articles/s41586-025-09989-7.

Aseem Datar. Transforming R&D with agentic AI: Introducing Microsoft Discovery. https://azure.microsoft.com/en-us/blog/transforming-rd-with-agentic-ai-introducing-microsoft-discovery/, 2025. URL https://azure.microsoft.com/en-us/blog/transforming-rd-with-agentic-ai-introducing-microsoft-discovery/. Microsoft Azure Blog, May 19, 2025.

C. Nicolò De Sabbata, Theodore R. Sumers, Badr AlKhamissi, Antoine Bosselut, and Thomas L. Griffiths. Rational metareasoning for large language models. *arXiv preprint arXiv:2410.05563*, 2024. URL https://arxiv.org/abs/2410.05563v3.

Indika Dissanayake, Jie Zhang, Mahmut Yasar, and Sridhar P. Nerur. Strategic effort allocation in online innovation tournaments. *Information & Management*, 55(3):396–406, 2018. doi: 10.1016/j.im.2017.09.006. URL https://doi.org/10.1016/j.im.2017.09.006.

Darshil Doshi, Aritra Das, Tianyu He, and Andrey Gromov. To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/105fdc31cc9eb927cc5a0110f4031287-Abstract-Conference.html.

Shanghua Gao, Ada Fang, Yepeng Huang, Valentina Giunchiglia, Ayush Noori, Jonathan Richard Schwarz, Yasha Ektefaie, Jovana Kondic, and Marinka Zitnik. Empowering biomedical discovery with AI agents. *Cell*, 187(22):6125–6151, 2024. doi: 10.1016/j.cell.2024.09.022. URL https://doi.org/10.1016/j.cell.2024.09.022.

Yiming Gao, Zhen Wang, Jefferson Chen, Mark Antkowiak, Mengzhou Hu, JungHo Kong, Dexter Pratt, Jieyuan Liu, Enze Ma, Zhiting Hu, and Eric P. Xing. scPilot: Large language model reasoning toward automated single-cell analysis and discovery. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/01dde7941bce255c2a061eef4fbb7fad-Abstract-Conference.html.

Samuel J. Gershman, Eric J. Horvitz, and Joshua B. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. *Science*, 349(6245):273–278, 2015. doi: 10.1126/science.aac6076. URL https://doi.org/10.1126/science.aac6076.

Guy Hacohen, Leshem Choshen, and Daphna Weinshall. Let's agree to agree: Neural networks share classification order on real datasets. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3950–3960. PMLR, 2020. URL https://proceedings.mlr.press/v119/hacohen20a.html.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. *arXiv preprint arXiv:2412.18547*, 2024. URL https://arxiv.org/abs/2412.18547v5.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023a. URL https://aclanthology.org/2023.emnlp-main.507/.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8fd1a81c882cd45f64958da6284f4a3f-Abstract-Conference.html.

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, Zhen Wang, and Zhiting Hu. New evaluation, library, and analysis of step-by-step reasoning with large language models. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2404.05221v2.

Brendan R. Hogan, Xiwen Chen, James T. Wilson, Kashif Rasul, Adel Boyarsky, Thomas Kamei, Anderson Schneider, and Yuriy Nevmyvaka. AlphaLab: Autonomous multi-agent research across optimization domains with frontier LLMs. *arXiv preprint arXiv:2604.08590*, 2026. URL https://arxiv.org/abs/2604.08590v1.

Zhengding Hu, Mingge Lu, Zhen Wang, Jixuan Ruan, Chang Chen, Zaifeng Pan, Yue Guan, Ruiyi Wang, Zhongkai Yu, Chao Zhang, and Yufei Ding. FlashEvolve: Accelerating agent self-evolution with asynchronous stage orchestration. *arXiv preprint arXiv:2605.08520*, 2026a. URL https://arxiv.org/abs/2605.08520v1.

Zhengding Hu, Zaifeng Pan, Prabhleen Kaur, Vibha Murthy, Zhongkai Yu, Yue Guan, Zhen Wang, Steven Swanson, and Yufei Ding. Pancake: Hierarchical memory system for multi-agent LLM serving. *arXiv preprint arXiv:2602.21477*, 2026b. URL https://arxiv.org/abs/2602.21477v1.

Zhengding Hu, Kuntal Talit, Zhen Wang, Haseeb Ahmad, Yichen Lin, Prabhleen Kaur, Christopher Lane, Elizabeth A. Peterson, Zhiting Hu, Elizabeth A. Nowadnick, and Yufei Ding. TritonDFT: Automating DFT with a multi-agent framework. *arXiv preprint arXiv:2603.03372*, 2026c. URL https://arxiv.org/abs/2603.03372v2.

Ahmadreza Jeddi, Minh Ngoc Le, Hakki C. Karaimer, Konstantinos G. Derpanis, and Babak Taati. GEAR: Genetic autoresearch for agentic code evolution. *arXiv preprint arXiv:2605.13874*, 2026. URL https://arxiv.org/abs/2605.13874v1.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025. URL https://arxiv.org/abs/2502.13138v1.

Ana Jojic, Zhen Wang, and Nebojsa Jojic. GPT is becoming a Turing machine: Here are some ways to program it. *arXiv preprint arXiv:2303.14310*, 2023. URL https://arxiv.org/abs/2303.14310.

Junmo Kang, Leonid Karlinsky, Hongyin Luo, Zhen Wang, Jacob Hansen, James Glass, David Cox, Rameswar Panda, Rogerio Feris, and Alan Ritter. Self-MoE: Towards compositional large language models with self-specialized experts. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDJUscOjM3.

Andrej Karpathy. autoresearch. https://github.com/karpathy/autoresearch, 2026. URL https://github.com/karpathy/autoresearch. Accessed on September 13, 2026.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning (ECML)*, pp. 282–293. Springer, 2006. doi: 10.1007/11871842_29. URL https://doi.org/10.1007/11871842_29.

Belinda Z. Li, Been Kim, and Zi Wang. QuestBench: Can LLMs ask the right question to acquire information in reasoning tasks? In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/c42c8d51556fabb4b57fc86d3d3d0d09-Abstract-Datasets_and_Benchmarks_Track.html.

Yushu Li, Wenlong Deng, Jiajin Li, and Xiaoxiao Li. Spend less, reason better: Budget-aware value tree search for LLM agents. *arXiv preprint arXiv:2603.12634*, 2026. URL https://arxiv.org/abs/2603.12634v1.

Buyun Liang, Liangzu Peng, Jinqi Luo, Darshan Thaker, Kwan Ho Ryan Chan, and René Vidal. SECA: Semantically equivalent and coherent attacks for eliciting LLM hallucinations. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d077bc9ea82a2998ca6b2d0158b5ac6e-Abstract-Conference.html.

Falk Lieder and Thomas L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1, 2020. doi: 10.1017/S0140525X1900061X. URL https://doi.org/10.1017/S0140525X1900061X.

Jieyuan Liu, Mengzhou Hu, Jefferson Chen, JungHo Kong, Pratibha Jagannatha, Yiming Gao, Dexter Pratt, Hsin-Yuan Lee, Zhiting Hu, Trey Ideker, Wei Wang, Eric P. Xing, and Zhen Wang. HypoEvolve: Genetic algorithms enable multi-agent LLMs to discover scientific hypotheses, 2026.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-aware tool use enables effective agent scaling. *arXiv preprint arXiv:2511.17006*, 2025. URL https://arxiv.org/abs/2511.17006v2.

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. URL https://arxiv.org/abs/2408.06292v3.

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 (NeurIPS)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Batu Ozturkler, Nikolay Malkin, Zhen Wang, and Nebojsa Jojic. ThinkSum: Probabilistic reasoning over sets using large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL https://aclanthology.org/2023.acl-long.68/.

Yaonan Qu and Meng Lu. Bilevel autoresearch: Meta-autoresearching itself. *arXiv preprint arXiv:2603.23420*, 2026. URL https://arxiv.org/abs/2603.23420v2.

Naama Rozen, Liat Bezalel, Gal Elidan, Amir Globerson, and Ella Daniel. Do LLMs have consistent values? In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/68fb4539dabb0e34ea42845776f42953-Abstract-Conference.html.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. RAGChecker: A fine-grained framework for diagnosing retrieval-augmented generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 21999–22027, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/27245589131d17368cccdfa990cbf16e-Abstract-Datasets_and_Benchmarks_Track.html.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, 2025. doi: 10.18653/v1/2025.findings-emnlp.320. URL https://aclanthology.org/2025.findings-emnlp.320/.

Yifei Shao, Kun Zhou, Ziming Xu, Mohammad Atif Quamar, Shibo Hao, Zhen Wang, Zhiting Hu, and Biwei Huang. Learning modal-mixed chain-of-thought reasoning with latent embeddings. *arXiv preprint arXiv:2602.00574*, 2026. URL https://arxiv.org/abs/2602.00574.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Herbert A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, 1955. doi: 10.2307/1884852. URL https://doi.org/10.2307/1884852.

Somanshu Singla, Zhen Wang, Tianyang Liu, Abdullah Ashfaq, Zhiting Hu, and Eric P. Xing. Dynamic rewarding with prompt optimization enables tuning-free self-alignment of language models. In *Proceedings of EMNLP*, pp. 21889–21909, 2024. URL https://aclanthology.org/2024.emnlp-main.1220/.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To CoT or not to CoT? chain-of-thought helps mainly on math and symbolic reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ead542f13a38179d1b55b88610f959a1-Abstract-Conference.html.

Yoonchang Sung, Shahaf S. Shperberg, Qi Wang, and Peter Stone. Effort allocation for deadline-aware task and motion planning: A metareasoning approach. *arXiv preprint arXiv:2410.05828*, 2024. doi: 10.48550/arXiv.2410.05828. URL https://arxiv.org/abs/2410.05828v1.

Kyle Swanson, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou. The virtual lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature*, 646(8085):716–723, 2025. doi: 10.1038/s41586-025-09442-9. URL https://www.nature.com/articles/s41586-025-09442-9.

Edan Toledo, Karen Hambardzumyan, Martin Josifoski, Rishi Hazra, Nicolas Baldwin, Alexis Audran-Reiss, Michael Kuchnik, Despoina Magka, Minqi Jiang, Alisia Maria Lupidi, Andrei Lupu, Roberta Raileanu, Kelvin Niu, Tatiana Shavrina, Jean-Christophe Gagnon-Audet, Michael Shvartsman, Shagun Sodhani, Alexander H. Miller, Abhishek Charnalia, Derek Dunfield, Carole-Jean Wu, Pontus Stenetorp, Nicola Cancedda, Jakob Nicolaus Foerster, and Yoram Bachrach. AI research agents for machine learning: Search, exploration, and generalization in MLE-bench. *arXiv preprint arXiv:2507.02554*, 2025. URL https://arxiv.org/abs/2507.02554v2.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arık. DynScaling: Efficient verifier-free inference scaling via dynamic and integrated sampling. *arXiv preprint arXiv:2506.16043*, 2025. URL https://arxiv.org/abs/2506.16043v1.

Peihao Wang, Ruisi Cai, Zhen Wang, Hongyuan Mei, Qiang Liu, Pan Li, and Zhangyang Wang. ∇-Reasoner: LLM reasoning via test-time gradient descent in latent space. In *International Conference on Learning Representations (ICLR)*, 2026a. URL https://arxiv.org/abs/2603.04948v1.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. PromptAgent: Strategic planning with language models enables expert-level prompt optimization. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/686a3f32067838c8dbb68da6e9e3cf69-Abstract-Conference.html.

Zhen Wang, Fan Bai, Zhongyan Luo, Jinyan Su, Kaiser Sun, Xinle Yu, Jieyuan Liu, Kun Zhou, Claire Cardie, Mark Dredze, Zhiting Hu, and Eric P. Xing. FIRE-Bench: Evaluating AI agents on the rediscovery of scientific insights. *arXiv preprint arXiv:2602.02905*, 2026b. URL https://arxiv.org/abs/2602.02905v2.

Zhen Wang, Yiming Gao, Jieyuan Liu, Enze Ma, Jefferson Chen, Mark Antkowiak, Mengzhou Hu, JungHo Kong, Dexter Pratt, Zhiting Hu, Wei Wang, Trey Ideker, and Eric P. Xing. CellMaster: Collaborative cell type annotation in single-cell analysis. *arXiv preprint arXiv:2602.13346*, 2026c. URL https://arxiv.org/abs/2602.13346v1.

Zhangchen Xu, Junda Chen, Yue Huang, Dongfu Jiang, Jiefeng Chen, Hang Hua, Zijian Wu, Zheyuan Liu, Zexue He, Lichi Li, Shizhe Diao, Jiaxin Pei, Jinsung Yoon, Hao Zhang, Mengdi Wang, Radha Poovendran, Misha Sra, Alex Pentland, and Zichen Chen. Autolab: Can frontier models solve long-horizon auto research and engineering tasks? *arXiv preprint arXiv:2606.05080*, 2026. URL https://arxiv.org/abs/2606.05080v1.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025. URL https://arxiv.org/abs/2504.08066v1.

Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuante Li, Haoran Pan, Yuge Zhang, Weiqing Liu, Yelong Shen, Weizhu Chen, and Jiang Bian. R&D-Agent: An LLM-agent framework towards autonomous data science. *arXiv preprint arXiv:2505.14738*, 2025. URL https://arxiv.org/abs/2505.14738v2.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025. URL https://arxiv.org/abs/2505.22954v3.

Zekai Zhao, Qi Liu, Kun Zhou, Zihan Liu, Yifei Shao, Zhiting Hu, and Biwei Huang. Activation control for efficiently eliciting long chain-of-thought ability of language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1951d443ea67f8fbcfe56622734a2edf-Abstract-Conference.html.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=shr9PXz7T0.

Jinxin Zhou, Chong You, Xiao Li, Kangning Liu, Sheng Liu, Qing Qu, and Zhihui Zhu. Are all losses created equal: A neural collapse perspective. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/cdce17de141c9fba3bdf175a0b721941-Abstract-Conference.html.

Yuxuan Zhou, Heng Li, Zhi-Qi Cheng, Xudong Yan, Yifei Dong, Mario Fritz, and Margret Keuper. MaxSup: Overcoming representation collapse in label smoothing. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/ec0707dee9db289906dfc38bd673415f-Abstract-Conference.html.

Dongyao Zhu, Zhen Wang, Xi Xiao, Han Jiang, Saeed Vahidian, Wei-Lun Chao, Tanya Berger-Wolf, Yu Su, Raju Vatsavai, and Jianyang Gu. Leveraging latent visual reasoning in silence. *arXiv preprint arXiv:2605.18641*, 2026. URL https://arxiv.org/abs/2605.18641.

Qiran Zou, Hou Hei Lam, Wenhao Zhao, Tingting Chen, Yiming Tang, Samson Yu, Yingtao Zhu, Srinivas Anumasa, Zufeng Zhang, Tianyi Zhang, Chang Liu, Zhengyao Jiang, Anirudh Goyal, and Dianbo Liu. FML-bench: A controlled study of AI research agent strategies from the perspective of search dynamics. *arXiv preprint arXiv:2605.17373*, 2026. URL https://arxiv.org/abs/2605.17373v2.

Table 5. **Comparison of representative autonomous research agents.** *Branch exploration* retains alternatives; *plan-level search* selects unexecuted plans between attempts. *Value backpropagation* updates ancestors from descendant outcomes. *Strategic effort planning* adjusts exploration and refinement to the remaining budget. A *shared token budget* covers all planning and execution tokens.

Method	Branch exploration	Plan-level search	Value backpropagation	Strategic effort planning	Shared token budget
AutoResearch (Karpathy, 2026)	–	–	–	–	–
AIDE (Jiang et al., 2025)	✓	–	–	–	–
The AI Scientist-v2 (Yamada et al., 2025)	✓	–	–	–	–
R&D-Agent (Yang et al., 2025)	✓	✓	–	✓	–
MARS (Chen et al., 2026)	✓	–	✓	–	–
AlphaLab (Hogan et al., 2026)	✓	✓	–	✓	–
PRIMESCIENTIST	✓	✓	✓	✓	✓

A EXTENDED RELATED WORK

Autonomous Research Agents. Autonomous research agents increasingly automate workflows covering ideation, experimental design, implementation, evaluation, and reporting. Systems such as AI Scientist, ResearchAgent, and Agent Laboratory generate research ideas or artifacts through iterative and multi-stage agentic workflows (Lu et al., 2024; Baek et al., 2025; Schmidgall et al., 2025; Liu et al., 2026). Metric-driven systems use task scores to guide progress. For example, Karpathy's `autoresearch` repeatedly modifies an experimental program and retains changes that improve the target metric, while AIDE formulates machine-learning engineering as tree search over code solutions (Karpathy, 2026; Jiang et al., 2025). More recent systems make search increasingly explicit. AI Scientist-v2 uses progressive agentic tree search to develop and evaluate experiments; AIRA compares greedy, MCTS, and evolutionary search policies for machine-learning research; GEAR maintains a population of research states through mutation and crossover; and FML-Bench studies how search topology and exploration dynamics affect research-agent performance (Yamada et al., 2025; Toledo et al., 2025; Jeddi et al., 2026; Zou et al., 2026). Together, these works establish iterative feedback, persistent research state, and structured search as central components of autonomous research. PRIMESCIENTIST makes research effort allocation an explicit design choice. It studies how empirical evidence and remaining resources should jointly guide the allocation of research attempts.

Agent Planning and Search. Research on language models studies reasoning, planning, and tool use (Hao et al., 2023a; 2024; Ozturkler et al., 2023; Hao et al., 2023b; Wang et al., 2026a; Jojic et al., 2023; Shao et al., 2026; Zhu et al., 2026). Recent work also makes cost or remaining resources explicit in reasoning and search (Han et al., 2024). MARS introduces cost-constrained MCTS for automated AI research and backpropagates rewards that combine validation performance with candidate execution time (Chen et al., 2026). AlphaLab uses a Strategist to revise experiment queues and a playbook to retain experimental lessons. The Strategist uses the remaining experiment count to shift from exploration toward refinement (Hogan et al., 2026). FML-Bench proposes AdaptiveSearch, which begins with greedy refinement and performs a one-time switch to multi-branch exploration after detecting stagnation, with its branching structure determined by the remaining step budget (Zou et al., 2026). Both approaches adapt trial selection while leaving the resources devoted to planning outside the allocation objective. At a finer granularity, BATS gives tool-using agents continuous awareness of remaining tool calls, while BAVT uses the remaining-resource ratio to sharpen a value-based selection distribution from broader exploration toward exploitation within a reasoning tree (Liu et al., 2025; Li et al., 2026). These search mechanisms inform our solution, yet our unique contribution is to formulate plan construction and execution as decisions under a shared resource budget. In the executable plan tree, selecting an untested plan initiates a research attempt; selecting an evaluated plan develops alternatives. The same policy therefore directs both research planning and experimentation using experimental evidence and remaining resources.

Resource Allocation in Intelligent Systems. Resource allocation is a longstanding question in the study of intelligent decision-making. Simon's account of bounded rationality relates effective choice to the information and computational capabilities available to an agent (Simon, 1955). Computational and resource-rational accounts further evaluate reasoning procedures by the quality of their decisions and the resources they require (Gershman et al., 2015; Lieder & Griffiths, 2020). Rational metar-

easoning treats the choice of what to compute as a decision in its own right. Recent work applies this principle to language models by learning when intermediate reasoning justifies its computational cost (De Sabbata et al., 2024). In adjacent domains, studies of online innovation tournaments show that solvers strategically vary effort in response to feedback and time remaining, and deadline-aware task-and-motion planning has been formulated as a metareasoned effort-allocation problem over alternative options (Dissanayake et al., 2018; Sung et al., 2024). These perspectives make the use of resources part of intelligent decision-making. In autonomous research, agents must construct alternatives and generate the evidence needed to assess them. PRIMESCIENTIST formalizes the effort devoted to both activities within a sequential decision problem. A research attempt can improve the current result and inform subsequent choices, linking its value to the opportunities for further research.

Self-Improving and Self-Evolving Agents. A broader line of work enables language-model systems to improve their behavior or revise their underlying procedures (Shinn et al., 2023; Madaan et al., 2023; Agrawal et al., 2026; Wang et al., 2024; Singla et al., 2024; Kang et al., 2025). Reflexion uses verbal feedback and episodic memory to guide subsequent attempts, while GEPA reflects on execution trajectories to evolve prompts and combine complementary lessons (Shinn et al., 2023; Agrawal et al., 2026). More strongly self-referential systems search over the agent itself. The Darwin Gödel Machine iteratively modifies its own coding-agent implementation and empirically validates the resulting variants, while Bilevel Autoresearch uses an outer autoresearch loop to generate new search procedures for an inner loop (Zhang et al., 2025; Qu & Lu, 2026). PRIMESCIENTISTadapts effort allocation across task-level research plans while retaining the same reflector and executor. Strategic effort allocation could help agents prioritize changes to their own research methods. Optimizing these choices would support recursive self-improvement, with each generation directing its resources toward developing more capable successors.

B ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS

B.1 IMPLEMENTATION DETAILS

The reflector normally reads a compressed execution log and consults the full log when necessary. It can inspect run metadata but receives no additional allocation rule. The executor receives the selected plan without instructions about the remaining global budget. The complete reflector prompt is in Appendix I. Both agents' input and output tokens count toward the shared budget. Costs convertible to a common monetary unit could fit the same formulation; separate resource constraints remain future work.

B.2 EFFECT OF BUDGET ADAPTIVITY

Effective research allocation requires deciding how much effort to devote to testing alternatives and developing promising plans. This balance can change as experiments produce new evidence and resources are consumed. We examine how the selection rule affects research quality, then test whether adapting exploration to the remaining budget improves the outcome.

We isolate the effect of plan selection by varying only the allocation policy, with the executor, reflector, plan tree, pruning rule, and budget accounting held fixed. In the GPT-5 experiments, *Random* samples children uniformly, *Greedy* selects the highest-valued child, and *UCT* (Kocsis & Szepesvári, 2006) uses the conventional exploration bonus with $c = \sqrt{2}$. The eight-task subset contains four tasks from FIRE-Bench and four from AutoLab, listed in Table 6.

Adaptive plan selection improves overall reward. Table 6 reports an average reward of 0.598 for PRIMESCIENTIST's policy, compared with 0.551 for *UCT*, 0.563 for *Greedy*, and 0.546 for *Random*. Only the allocation policy changes in this comparison, linking the higher overall reward to how research effort is distributed across plans. PRIMESCIENTIST exceeds *UCT* on seven tasks and matches it on the eighth. The gain therefore extends across the evaluated subset, even though both policies use the same plan representation and reflector. *Random* leads on *SECA Hallucination* and *Gaussian Blur*, showing that the best policy can differ across tasks.

Adapting exploration to the remaining budget improves reward. The fixed-exponent comparison directly tests the benefit of budget adaptivity. Using GPT-5-mini, we compare PRIMESCIENTIST's allocation rule with three constant exponents: $\alpha = 0$ (uniform sampling), $\alpha = 3$, and $\alpha = 10$

Table 6. **Adaptive allocation improves research outcomes.** The PRIMESCIENTIST policy yields the highest overall reward when only plan selection changes. All variants share the executor, reflector, plan tree, pruning rule, and 1.5M-token budget. Each configuration uses one GPT-5 search.

Benchmark	Task	PRIMESCIENTIST	UCT	Greedy	Random
FIRE-Bench	Activation Control	0.381	0.353	0.279	0.381
	LLM Value Consistency	0.889	0.889	0.889	0.727
	SECA Hallucination	0.698	0.585	0.686	0.750
	QuestBench	0.682	0.649	0.600	0.545
AutoLab	Concurrent KV WAL	0.633	0.618	0.635	0.517
	Flash Attention	0.357	0.304	0.301	0.341
	Hash Join	0.631	0.619	0.625	0.591
	Gaussian Blur	0.512	0.394	0.488	0.516
Average (8 tasks)		0.598	0.551	0.563	0.546

Table 7. **Adapting exploration improves reward over fixed policies.** PRIMESCIENTIST adapts the sampling exponent to the remaining budget; the alternatives hold it constant. Both agents use GPT-5-mini with a shared 1M-token budget. Entries give mean maximum reward (population standard deviation) over three independent searches for PRIMESCIENTIST and two per fixed policy.

Benchmark	Task	PRIMESCIENTIST (n=3)	**Random** (α=0) (n=2)	**Fixed** (α=3) (n=2)	**Greedy** (α=10) (n=2)
FIRE-Bench	Activation Control	0.314 (0.010)	0.245 (0.034)	0.203 (0.016)	0.279 (0.045)
	QuestBench	0.555 (0.085)	0.664 (0.064)	0.467 (0.005)	0.431 (0.031)
	LLM Value Consistency	0.850 (0.108)	0.856 (0.067)	0.656 (0.071)	0.962 (0.038)
AutoLab	Concurrent KV WAL	0.583 (0.020)	0.369 (0.084)	0.499 (0.018)	0.451 (0.196)
	Hash Join	0.626 (0.003)	0.582 (0.029)	0.617 (0.002)	0.624 (0.008)
	Flash Attention	0.294 (0.037)	0.294 (0.030)	0.270 (0.030)	0.297 (0.088)
Average (6 tasks)		0.537	0.502	0.452	0.507
best (highest mean) per task		3 / 6	1 / 6	0 / 6	2 / 6

(strongly concentrated sampling, labeled *Greedy*). The research framework and sampling rule remain fixed; the exponent either adapts to the remaining budget or stays constant. This finite-exponent policy differs from deterministic *Greedy* selection in Table 6.

Adapting the selection exponent achieves the strongest overall reward without choosing a separate constant for each task. Table 7 reports the highest overall mean for PRIMESCIENTIST's policy (0.537), followed by the fixed $\alpha = 10$ policy (0.507), uniform sampling (0.502), and fixed $\alpha = 3$ (0.452). Our policy leads on three of six tasks, exceeds fixed $\alpha = 3$ on all six, and never ranks last. Uniform sampling leads on *QuestBench*, while $\alpha = 10$ leads on *LLM Value Consistency*.

B.3 CONSISTENCY ACROSS RUNS AND TASKS

Sample-efficiency gains should persist across independent searches and different task requirements. We examine this consistency using three searches per method and task with GPT-5-mini on AutoLab (Table 8). Alongside reward and research attempts, we inspect accuracy on the task whose reward threshold can obscure improvements in the underlying solution.

Comparable reward with fewer research attempts. Across the six tasks included in the average, Table 8 reports mean rewards of 0.4363 for PRIMESCIENTIST and 0.4359 for AutoResearch. The mean number of research attempts falls from 28.4 to 21.6, a reduction of approximately 24%. The average attempt count is lower on every task across three independent searches per method and task, supporting a consistent sample-efficiency benefit across this evaluation. On *Concurrent KV WAL*, PRIMESCIENTIST improves reward (0.5830 versus 0.5608) while reducing attempts (13.0 versus 25.7). On *Hash Join*, *Flash Attention*, and *FFT (Rust)*, it achieves comparable reward with fewer attempts. These results show that strategic allocation can preserve task performance while reducing the number of research attempts across distinct optimization problems. Appendix F.3 further examines how these benefits vary with the total token budget.

Table 8. **Comparable reward with fewer research attempts on AutoLab.** Three independent GPT-5-mini searches per configuration (1M tokens each) test consistency. Rewards are mean ± population standard deviation; *#Attempts* is the mean count. The separate *Smallest Game Player* row reports maximum accuracy as mean (range), showing progress below its 0.95 reward threshold.

	PRIMESCIENTIST		AutoResearch	
Task	**Max reward (mean ± std)**	**#Attempts**	**Max reward (mean ± std)**	**#Attempts**
Gaussian Blur	0.4740 ± 0.0487	25.7	0.4859 ± 0.0145	29.3
Hash Join	0.6259 ± 0.0029	27.0	0.6291 ± 0.0050	36.0
Concurrent KV WAL	0.5830 ± 0.0203	13.0	0.5608 ± 0.0377	25.7
Flash Attention	0.2939 ± 0.0372	17.3	0.2960 ± 0.0113	19.3
FFT (Rust)	0.5002 ± 0.0037	30.7	0.5133 ± 0.0151	41.0
Sha256 Throughput	0.1410 ± 0.1523	16.0	0.1304 ± 0.1540	19.3
Average (6 tasks)	0.4363	21.6	0.4359	28.4
Accuracy-gated task (excluded from the average); maximum accuracy: mean (range).				
Smallest Game Player	0.926 (0.916–0.940)	12.3	0.918 (0.872–0.944)	22.0

Higher accuracy with fewer research attempts. On *Smallest Game Player*, PRIMESCIENTIST achieves a higher mean maximum accuracy (0.926 versus 0.918) with fewer attempts (12.3 versus 22.0). Table 8 reports accuracy separately to expose progress below the task's 0.95 reward threshold. Across the three searches, PRIMESCIENTIST's maximum accuracy ranges from 0.916 to 0.940, compared with 0.872 to 0.944 for AutoResearch. The higher minimum accuracy complements the higher average, showing more consistent progress toward the correctness threshold in this setting. Reporting accuracy makes this progress visible even when the gated reward remains zero.

B.4 RESEARCH QUALITY AS INPUT TOKENS ACCUMULATE

Figure 4 traces a single search on *LLM Value Consistency*, which studies whether LLM responses express consistent human values across contexts and question framings (Rozen et al., 2025). PRIMESCIENTIST's running-best score continues to improve after AutoResearch plateaus, reaching approximately 0.89 versus 0.73. The horizontal axis shows input tokens, whereas the search budget includes both input and output tokens. This single trajectory illustrates search behavior rather than variation across independent runs.

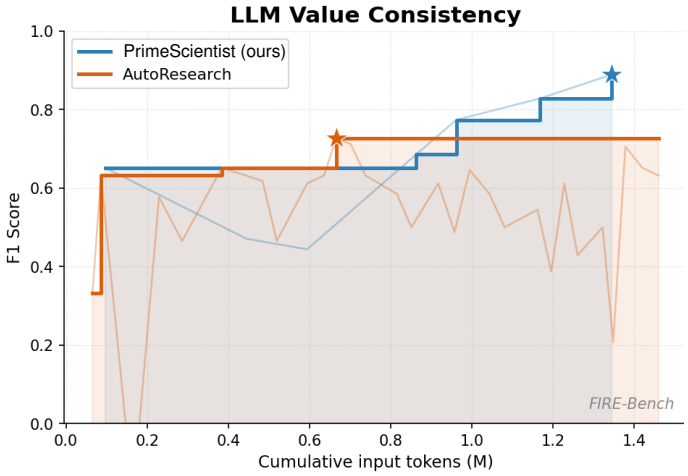

Figure 4: **Research progress as input tokens accumulate.** On *LLM Value Consistency*, PRIMESCIENTIST continues improving after AutoResearch plateaus. Each method contributes one search; thin lines show per-attempt F1 and thick lines track the best score reached.

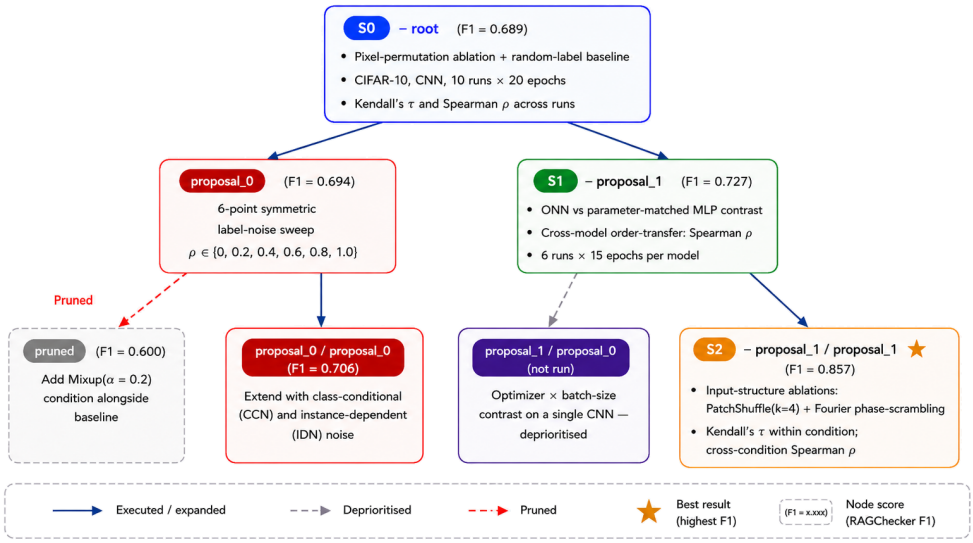

Figure 5: **Selective plan expansion improves research findings.** On *Learning Order Agreement*, PRIMESCIENTIST studies why networks learn images in similar orders (Hacohen et al., 2020). The tree develops label-noise and architecture experiments as alternative explanations. Image-structure tests in the architecture branch raise F1 from 0.689 to 0.857. F1 measures agreement with reference findings.

B.5 CASE STUDY OF STRATEGIC EFFORT ALLOCATION

To understand how strategic allocation shapes the experiments an agent performs, we examine *Learning Order Agreement*. Independently trained networks often learn to classify the same images earlier than others (Hacohen et al., 2020). The task asks what explains this shared learning order. Figure 5 shows how PRIMESCIENTIST tests competing explanations and chooses which plans to develop further.

Executable plans test alternative explanations. The reflector begins with repeated CNN training on CIFAR-10, including pixel-permutation and random-label comparisons. The coding agent executes this plan and reports findings, scoring F1 0.689. The reflector then proposes label-noise and architecture experiments to test alternative explanations for shared learning order.

Selective expansion yields stronger research findings. Both branches develop further experiments. The label-noise branch examines how different patterns of incorrect labels affect learning order, while a lower-scoring Mixup extension is pruned. The architecture branch alters image structure through patch shuffling and Fourier phase scrambling, reaching F1 0.857. A separate optimizer and batch-size comparison remains unexecuted. The search thus explores different explanations while selectively committing effort to their experimental tests.

In this case, PRIMESCIENTIST matches AutoResearch's research quality with fewer attempts. Both reach F1 0.857, using 12 and 30 attempts, respectively (Table 2). The tree makes this allocation visible through the alternatives developed, pruned, or left unexecuted.

C LIMITATIONS

Planning under limited resources. The reflector is a full LLM agent, and a reflection call can consume tokens on the same order as a coding-agent run. Sharing budget B makes the balance between constructing alternatives and executing them explicit. The budget analysis in Appendix F.3 identifies task-dependent benefits at different budgets; characterizing the point at which further planning is preferable to immediate execution remains an open research direction.

Scope of resource accounting. We measure inference cost in input and output tokens across both agents, and experimental effort in complete executor invocations. Each invocation constitutes

one research attempt; its training workload, runtime, and monetary cost depend on the task and execution environment. A shared token budget constrains inference expenditure, while realized consumption depends on the completed actions and stopping point. Runtime is measured separately in Appendix F.4. Extending this accounting to monetary cost or energy requires the corresponding execution measurements, hardware characteristics, and inference pricing.

Variability across independent searches. Complete research attempts provide task-level evidence for branch selection, with the amount of evidence determined by the available budget. On FIRE-Bench, each plan is executed twice and the higher score is retained; both invocations count toward inference consumption and experimental effort. The two or three independent full-search repetitions in the GPT-5-mini setting characterize variability through descriptive means and standard deviations. Larger repeated-search panels would enable more precise estimates of small reward differences. Statistical significance and backbone effects require separate analysis. The primary GPT-5 setting uses a different budget and task coverage.

Plan-level granularity. PRIMESCIENTIST allocates complete research attempts and receives their empirical outcomes after execution. This granularity connects strategic decisions to task-level evidence, while leaving decisions within an attempt to the coding agent. Even a narrowly specified plan modification requires another complete attempt for evaluation. The current method therefore cannot reallocate effort among intermediate implementation decisions during an ongoing execution.

D BROADER IMPACT

Accessible autonomous research. Explicit allocation provides a framework for studying how limited inference resources should be distributed across research directions. Achieving competitive outcomes with fewer complete research attempts can make autonomous research more accessible when empirical trial and error requires substantial implementation and experimentation. The framework includes planning in the inference budget and exposes the corresponding demand for complete research attempts, supporting resource-aware choices about research effort. Application beyond the evaluated task classes also requires suitable execution environments and reliable feedback.

Strategic allocation for recursive self-improvement. A longer-term opportunity is *recursive self-improvement*, in which research agents develop improved research procedures and use them to guide subsequent improvements (Zhang et al., 2025; Qu & Lu, 2026). Our formulation offers a way to study how such systems divide limited resources between proposing changes to the researcher and empirically testing those changes. Strategic allocation could help sustain this improvement loop by directing effort toward promising changes while retaining alternatives for future evaluation.

Scientific validation and reproducibility. Stored parent plans and executable modifications make research decisions traceable and support reconstruction of the plan tree. Reliable scientific use requires independent validation of conclusions beyond the scalar reward used to guide search. Retaining execution records, documenting software environments, and using available held-out tests help verify reproducible improvements and detect evaluator exploitation.

Resource transparency. The reported inference budgets and numbers of research attempts document distinct uses of research resources. End-to-end energy use depends on inference, experimental execution, and hardware utilization, requiring measurements beyond token and attempt counts.

E ALGORITHM

Algorithm 1 PRIMESCIENTIST Plan Search and Effort Allocation

Require: Task instruction $\mathcal{T}$, evaluator h, reflector f, selection policy π (defined in §3.3), branching factor m, total agentic budget B, exponent cap $\alpha_{\max}$, prune threshold δ

Initialise tree $T \leftarrow \{s_0\}$ where $s_0 \leftarrow f(\emptyset;\, \mathcal{T})$ {root executable plan drafted from instruction}
$U \leftarrow$ tokens consumed in drafting s_0
while $U < B$ **do**
  $\alpha_t \leftarrow \min\big(1/r_t,\ \alpha_{\max}\big),\ r_t \leftarrow (B-U)/B$
  $s \leftarrow \pi(T;\, \alpha_t)$ {research effort allocation; pruned nodes are skipped, so a node whose children are all pruned is returned as a leaf}
  if s unevaluated **then**
    $\rho \leftarrow h(s)$; record the observed outcome at s
    if $s \neq s_0$ and $\delta > 0$ and $Q(\mathrm{parent}(s)) > 0$ and $\rho < Q(\mathrm{parent}(s)) - \delta$ **then**
      exclude s from subsequent selection {prune}
    else
      backpropagate ρ to s and its ancestors
    end if
  else
    $\{s'_1, \ldots, s'_m\} \leftarrow f\big(s;\, \Sigma(T)\big)$ {reflect; propose m children}
    $T \leftarrow T \cup \{s'_1, \ldots, s'_m\}$
  end if
  update U with tokens consumed in this iteration
end while
return $\arg\max_{s \in T:\, s \text{ evaluated}} h(s)$

In Algorithm 1, $\Sigma(T)$ denotes the tree context supplied to the reflector during plan expansion.

F RESOURCE USE AND BUDGET SENSITIVITY

The main results show that PRIMESCIENTIST can achieve comparable or better research outcomes with fewer attempts. The AutoLab analyses below examine how this benefit relates to resource use. We first account for the inference spent on planning and execution, then compare research quality at matched token budgets and attempt counts. We next examine how outcomes change with the total budget and measure elapsed search time. These analyses provide complementary views of research efficiency, with the comparison conditions specified separately for each study.

F.1 INFERENCE ALLOCATION AND EXPERIMENTAL EFFORT

Planning consumes inference resources before a research direction is tested. To understand this expenditure, we group token use into coding-agent execution and planning, including reflection, tree context, and child-plan generation (Table 9). This accounting captures the resources used to choose research directions alongside those used to execute them.

Table 9. **Inference spent on planning and execution.** Planning accounts for a substantial share of token use, including reflection, tree context, and child-plan generation. Both components contribute to the total inference expenditure.

Component	Token share
Coding-agent tokens	62%
Reflection, tree context, and child-plan generation	38%

Coding-agent execution accounts for 62% of the token use in Table 9, while planning accounts for 38%. The reflector alone uses 21% to 37% of tokens per run, and a plan expansion consumes 0.1 to 0.3 million tokens. Planning is therefore a substantial part of the shared budget. The next comparison examines the number of research attempts performed after including this planning expenditure.

For the six-task comparison in Table 10, each method receives a total budget of 1M tokens per task. PRIMESCIENTIST performs 112 research attempts across the six tasks, compared with 158 for AutoResearch, a reduction of approximately 29%. It uses fewer attempts on every task; on *Hash Join*, for example, the count falls from 34 to 21. Section 4.4 examines consistency across repeated searches, while the following subsection examines the quality of the resulting solutions.

Table 10. **Fewer research attempts under a fixed inference budget.** PRIMESCIENTIST uses approximately 29% fewer attempts across six AutoLab tasks (112 versus 158). Each method has 1M tokens per task for planning and execution.

Task	PRIMESCIENTIST	AutoResearch
Gaussian Blur	18	22
Hash Join	21	34
FFT (Rust)	30	42
Concurrent KV WAL	11	24
Flash Attention	18	19
Sha256 Throughput	14	17
Total	**112**	**158**

F.2 REWARD UNDER MATCHED RESOURCES

To assess the quality obtained from the available resources, we compare PRIMESCIENTIST and AutoResearch under two matching criteria. The first fixes total inference, including both planning and execution. The second fixes the number of research attempts, giving both methods the same number of empirical tests within each task.

At the shared checkpoint of 1M tokens, Table 11 summarizes rewards over three searches per method and task. The rewards are comparable across the six tasks, with PRIMESCIENTIST reaching 0.583 versus 0.561 on *Concurrent KV WAL* and 0.141 versus 0.130 on *Sha256 Throughput*. These results show that PRIMESCIENTIST preserves competitive solution quality when its planning and execution must share the same total inference budget.

Table 11. **Research quality at matched token budgets.** AutoLab rewards remain comparable when each method has 1M tokens for planning and execution. Values are mean ± standard deviation across three searches per method and task.

Task	PRIMESCIENTIST	AutoResearch
Gaussian Blur	0.460 ± 0.03	0.485 ± 0.02
Hash Join	0.626 ± 0.00	0.629 ± 0.01
FFT (Rust)	0.499 ± 0.01	0.513 ± 0.02
Concurrent KV WAL	0.583 ± 0.02	0.561 ± 0.04
Flash Attention	0.294 ± 0.04	0.296 ± 0.01
Sha256 Throughput	0.141 ± 0.15	0.130 ± 0.15

We next match the number of research attempts within each task to examine the quality achieved with the same amount of experimentation. Table 12 presents the three improved cases from a six-task comparison. PRIMESCIENTIST achieves higher rewards on *Hash Join*, *FFT (Rust)*, and *Concurrent KV WAL*; on the latter, reward increases from 0.551 to 0.583. These cases show that strategic allocation can produce stronger outcomes from the same number of research attempts.

Table 12. **Higher reward at matched research attempts.** The table shows the three improved cases from a six-task AutoLab comparison with the number of research attempts matched within each task.

Task	PRIMESCIENTIST	AutoResearch
Hash Join	0.625	0.617
FFT (Rust)	0.498	0.493
Concurrent KV WAL	0.583	0.551

F.3 SENSITIVITY TO THE TOKEN BUDGET

A larger budget creates room both to construct alternative plans and to execute them. The benefit depends on whether the additional research effort leads to stronger solutions. We compare PRIMESCIENTIST and AutoResearch on three AutoLab tasks at four total token budgets, from 250k to 1M, with the same budget assigned to both methods in each comparison (Table 13). This sweep examines when additional resources improve outcomes and how the benefit varies across tasks.

Concurrent KV WAL shows the clearest benefit from additional resources for PRIMESCIENTIST. Its reward rises from 0.42 to 0.57 as the budget grows from 250k to 1M tokens, and it leads at 750k and 1M tokens. On *Sha256 Throughput*, both methods improve substantially between 500k and 750k tokens, reaching 0.19 at 750k. On *Flash Attention*, rewards range from 0.26 to 0.33 across the two methods and four budgets. The sweep thus distinguishes gains shared by both methods from settings where strategic allocation produces stronger outcomes.

Table 13. **Task-dependent gains from additional research resources.** The sweep compares rewards at four total token budgets; entries list PRIMESCIENTIST / AutoResearch. At 750k and 1M tokens, PRIMESCIENTIST achieves higher reward on *Concurrent KV WAL*.

Task	250k	500k	750k	1M
Concurrent KV WAL	$0.42/0.52$	$0.47/0.51$	$0.52/0.49$	$0.57/0.54$
Flash Attention	$0.30/0.30$	$0.26/0.33$	$0.30/0.32$	$0.27/0.30$
Sha256 Throughput	$0.02/0.02$	$0.01/0.03$	$0.19/0.19$	$0.20/0.18$

F.4 ELAPSED SEARCH TIME

Elapsed time measures how long a user waits for the research outcome. Work on agent infrastructure shows that inference serving and workflow scheduling also shape this duration (Hu et al., 2026b;a). We therefore measure search time directly, complementing inference tokens and research attempts with the time required to deliver a result. Table 14 reports durations across complete searches.

The comparison includes twelve AutoLab searches per method. PRIMESCIENTIST completes these searches in 32 minutes on average, compared with 42 minutes for AutoResearch; the median is 34 minutes for both. The lower average completion time provides an additional measure of practical efficiency alongside inference tokens and research attempts.

Table 14. **Reduced elapsed time for AutoLab searches.** In 12 searches per method, mean time falls from 42 to 32 minutes with PRIMESCIENTIST; the median is 34 minutes for both methods.

Method	Runs	Mean (min)	Median (min)
PRIMESCIENTIST	12	32	34
AutoResearch	12	42	34

G BENCHMARK TASKS AND EVALUATION TARGETS

The benchmarks cover distinct research objectives. FIRE-Bench tasks study AI research questions; AutoLab tasks require faster or smaller implementations; MLE-Bench tasks require predictive models for competition datasets. Tables 15, 16, and 17 explain the questions, optimization goals, and evaluation targets behind the task names. The agent develops and tests its plans under the experimental budgets in Section 4.1.

Table 15. **Research questions behind the FIRE-Bench tasks.** Each row explains the phenomenon studied and identifies its source paper. The agent designs and conducts experiments to answer each question; F1 measures agreement between its conclusions and the reference findings.

Task and source	What the agent studies
Activation Control (Zhao et al., 2025)	Intervene on internal LLM activations during inference and examine changes in reasoning length, self-reflection, and answer accuracy.
SECA Hallucination (Liang et al., 2025)	Rephrase questions while preserving their meaning and coherence to test whether plausible wording changes induce hallucinations.
QuestBench (Li et al., 2025)	Present reasoning problems with missing information and test whether an LLM selects the clarification question needed to solve them.
Learning Order Agreement (Hacohen et al., 2020)	Track which examples neural networks learn to classify earlier during training. Compare independently trained models and altered datasets to explain similarities in this order.
LLM Value Consistency (Rozen et al., 2025)	Elicit responses to human-value questionnaires under different contexts and framings. Examine whether the expressed values form a consistent structure, including correlations between values.
To CoT or not to CoT (Sprague et al., 2025)	Compare direct answers with chain-of-thought reasoning across task types to identify when intermediate reasoning improves accuracy.
Grokking or Not (Doshi et al., 2024)	Train networks on modular arithmetic with corrupted labels to study how memorization, generalization, and regularization interact during training.
Max Suppression (Zhou et al., 2025)	Compare label smoothing with maximum-logit suppression in image classification, examining prediction confidence and variation among features from the same class.
Neural Collapse Losses (Zhou et al., 2022)	Train classifiers with different losses and examine whether their learned features converge to similar within-class clusters and between-class geometry.
CoT Faithfulness Gaps (Chen et al., 2025)	Insert answer hints into reasoning questions and examine whether a model acknowledges the hints when they influence its answer.
Lost in The Middle (Liu et al., 2024)	Vary the position of relevant information in long input contexts and measure its effect on question answering and retrieval performance.
MCQ Selection Bias (Zheng et al., 2024)	Reorder multiple-choice options to test whether answer position changes an LLM's selection despite unchanged question content.

The AutoLab tasks (Xu et al., 2026) test concrete implementation decisions. The agent starts from a working program and seeks better performance while satisfying correctness requirements. Table 16 distinguishes runtime optimization from the model-size objective in *Smallest Game Player*.

MLE-Bench (Chan et al., 2025) requires agents to develop prediction pipelines from the supplied competition data. Table 17 explains the input, prediction target, and evaluation metric for each competition. The task names link to the benchmark's descriptions, and score directions match Table 3.

Table 16. **Optimization objectives of the AutoLab tasks.** These tasks span image processing, database operations, numerical computation, and compact model design. Each objective is assessed under the benchmark's correctness requirements.

Task	What the agent optimizes
Gaussian Blur	Reduce the time required to smooth an image with a Gaussian filter.
Hash Join	Accelerate table joins that match rows by key, while preserving the correct joined output.
Concurrent KV WAL	Speed up concurrent operations in a key-value store with write-ahead logging, which records updates for recovery.
Flash Attention	Reduce the runtime of scaled dot-product attention through more efficient computation and memory access.
FFT (Rust)	Compute the discrete Fourier transform of a signal faster in Rust.
VLIW Scheduler	Group instructions into bundles for a very long instruction word processor to reduce execution cycles.
Smallest Game Player	Minimize a Connect-3 network's parameter count while maintaining at least 95% accuracy against perfect-play decisions.
Sha256 Throughput	Increase SHA-256 hashing throughput while preserving correct outputs.

Table 17. **Prediction tasks and metrics in MLE-Bench.** The seven competitions cover medical imaging, materials properties, plant health, authorship and species classification, and document restoration. RMSLE denotes root mean squared logarithmic error; RMSE denotes root mean squared error; ROC AUC measures the area under the receiver operating characteristic curve.

Task	What the model predicts	Evaluation metric
APTOS 2019 Blindness	Diabetic retinopathy severity on a five-level scale from retinal images.	Quadratic weighted kappa ↑
NOMAD 2018	Formation energy and bandgap energy of transparent-conductor materials from their composition and structure.	Mean RMSLE over the two targets ↓
Plant Pathology 2020	Whether an apple leaf is healthy or affected by rust, scab, or multiple diseases, using its photograph.	Mean ROC AUC across categories ↑
Spooky Author ID	Which of three authors wrote a text excerpt: Edgar Allan Poe, Mary Shelley, or H. P. Lovecraft.	Multiclass log loss ↓
Leaf Classification	Which of 99 plant species a leaf belongs to, using its image and shape, margin, and texture features.	Multiclass log loss ↓
Histopathologic Cancer	Whether the central region of a pathology image patch contains metastatic tumor tissue.	ROC AUC ↑
Denoising Dirty Documents	Clean grayscale pixel intensities from noisy scanned document images.	Pixelwise RMSE ↓

H SKILL FORMAT

Each research plan is a structured Markdown skill supplied to the coding agent before a research attempt. For FIRE-Bench, it contains: the research question decomposition, hypotheses to test, implementation plan, dataset handling strategy, and pitfalls to avoid. For AutoLab, it contains: which files to edit, prioritized optimization techniques, correctness constraints, and expected performance targets.

The root skill is written by the reflector from scratch given only the task instruction. Each subsequent skill is produced by running the child's `diff.py`, which applies a focused modification to the parent's skill. The stored root plan and executable modifications allow each descendant plan to be reconstructed. Every modification records a reflector decision and is applied to its parent plan.

I REFLECTOR PROMPT TEMPLATE

During plan expansion, the reflector reads the selected node's execution outputs and proposes child plans. Listing 1 is the full system prompt delivered to the reflector. Runtime placeholders appear in `<angle-bracket>` form: `<task-id>` is the benchmark task name, `<node-path>` is the relative path to the completed node, and `N` is the branching budget (maximum number of child proposals). The `[PRUNED NODES]` line appears only when earlier nodes have been pruned.

```
You are a reflector for task: <task-id>

A new run has just completed. Your working directory is the search
↪ tree
root for this task.  The completed node is at: <node-path>/

If this is not your first call on this task, you are RESUMING a
↪ previous
session -- you already have memory of files and proposals you
↪ inspected
last time.  Do NOT re-read files whose content you already remember;
↪ just
check for NEW files (e.g. the newly-completed node's packet.json,
↪ result
directories) and whatever sibling state has changed since your last
↪ call.

Relevant files (anything not listed here is noise -- skip it):
  [PRUNED NODES]
  <node-path>/packet.json     -- reward score (precision/recall/f1)
↪ and run
                                 metadata.  Reflects correctness of
↪ the
                                 agent's conclusion.
  <node-path>/log_brief.log   -- head+tail excerpt of the agent run.
↪ Primary
                                 diagnostic tool: look for Python
↪ tracebacks,
                                 missing-file errors, or the agent
↪ admitting
                                 it could not finish.  The agent often
↪  claims
                                 success -- trust packet.json, not the
↪  narrative.
  <node-path>/.log.log        -- FULL agent log (very large).  Read
↪ ONLY if
                                 log_brief.log leaves the failure
↪ ambiguous
                                 (e.g. tail ends mid-traceback).
                                 Do NOT read routinely; use only when
↪ the
                                 reward score is very low.
```

```
  <node-path>/skill.md        -- the experimental plan this run
↪ executed.
  <node-path>/changes.md      -- (if present) short prose summary of
↪ what
                                 this node changed relative to its
↪ parent.
  <node-path>/sandbox/        -- the agent's experiment workspace.
↪ Use `ls`
                                 to discover result directories, then
↪ read
                                 specific output files only if needed.
  Other nodes' prior.json / skill.md / packet.json -- to review
↪ previous
                                 trials' approaches and performance.

Your job:
1. Read packet.json for the score.  Read log_brief.log for the run
   trajectory.  Use `ls <node-path>/sandbox/` to find result
↪ directories,
   then read specific files if the log is insufficient.
   Ground your analysis in concrete observations, not assumptions.

2. Check other nodes' prior.json / skill.md / packet.json as needed to
   avoid re-proposing already-tried hypotheses and to learn what has
↪ and
   has not worked.

3. Identify whether failure was due to a setup/runtime error (fixable
↪ by
   changing the experimental plan) or a genuine result (hypothesis
↪ tested
   but scored low).

4. Propose between 1 and N CONTROVERSIALLY DIFFERENT skill variants.
   "Controversially different" means each proposal bets on a
↪ fundamentally
   different hypothesis -- different hyperparameter regime, different
↪ metric
   interpretation, different dataset choices, etc.  Do NOT create
↪ proposals
   that differ only in a minor hyperparameter value or wording; those
↪ are
   not separate bets.

   Decide how many to create:
   - One clearly dominant direction: write 1 proposal.
   - Two to N genuinely competing hypotheses: write that many.
   - Never pad with minor variations just to hit a count.

   Create child directories numbered from where you left off:
     <node-path>/children/proposal_0/
     <node-path>/children/proposal_1/
     ...  (up to proposal_{N-1}/)

   In each child directory write THREE files (and run the diff):

   a) diff.py -- a self-contained Python script that reads the parent'
↪ s
      skill.md and writes a NEW COMPLETE skill.md into this directory.
      The output skill.md must be a FULL self-contained experimental
↪ plan
      (dataset, model, hyperparameters, evaluation metrics, conclusion
      structure) -- NOT a delta.  Template:

        from pathlib import Path
```

```
        parent = Path(__file__).parents[2] / "skill.md"
        output = Path(__file__).parent     / "skill.md"
        text   = parent.read_text()
        # Apply targeted transformations, e.g.:
        # text = text.replace("model = X", "model = Y")
        output.write_text(text)

   b) Run it immediately:  python diff.py
      Verify skill.md was created and is the full updated plan.
      If diff.py errors, fix it and re-run before continuing.

   c) changes.md  (< 1 KB) -- short prose summary written FOR the
↪ child
      agent (3-6 bullet points), e.g.:
        - switched dataset from X to Y
        - moved evaluation metric from accuracy to F1
        - added ablation on hyperparameter Z
      The child agent reads this in its inherited sandbox to know what
      to update without repeating the parent's work.

   d) prior.json -- structured reasoning for the next reflector to
↪ consult:
        {
          "estimate":   <float 0-1>,
          "hypothesis": "<single concrete hypothesis this proposal
↪ tests>",
          "rationale":  "<2-3 sentences: WHY this should improve the
↪ score,
                        citing observations from packet.json /
↪ log_brief.log
                        / sandbox>",
          "changes":    "<1 sentence: which files/sections are
↪ modified and
                        at what abstraction level>",
          "risks":      "<1-2 sentences: most plausible failure mode>"
        }

   Do NOT write skill.md directly -- only via diff.py.

Print a one-line summary for each proposal and each pruned branch.
```

Listing 1: Reflector prompt template with runtime placeholders in <angle-bracket> notation.