COST-EFFECTIVE AGENT TEST-TIME SCALING VIA BUDGET-AWARE THINKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling test-time computation improves performance across different tasks on large language models (LLMs), which has also been extended to tool-augmented agents. For these agents, scaling involves not only "thinking" in tokens but also "acting" via tool calls. Unlike tokens in textual reasoning, the number of tool calls directly bounds the agent's interaction with the external environment. To this end, we study how to scale such agents under explicit tool-call budgets, focusing on web search agents equipped with search and browse tools. We introduce CATS (**C**ost-effective **A**gent **T**est-time **S**caling), a budget-aware framework designed for effective and efficient agent scaling. CATS integrates a lightweight budget tracker that provides a continuous signal of remaining resources to the agent's core modules, encouraging budget-aware adaptations in planning and verification. By constraining the number of tool calls and unifying the costs of both token and tool consumption, we analyze the cost–performance scaling behavior in a more controlled manner. Experiments across search-intensive benchmarks show that CATS produces more favorable scaling curves, attaining higher accuracy with fewer tool calls and lower overall cost. Our work advances a cost-conscious design for agent test-time scaling and offers empirical insights toward a more transparent and principled understanding of scaling in tool-augmented agents.

1 INTRODUCTION

Scaling test-time compute in large language models (LLMs) helps improve the performance across a wide range of tasks including reasoning, coding (Snell et al., 2024; Muennighoff et al., 2025; Wu et al., 2025; Chen et al., 2025b). Mainstream scaling strategies such as sequential (Madaan et al., 2023) and parallel scaling (Wang et al., 2023; Brown et al., 2024) enable models to utilize more effort, elicit deeper reflection, and refine their outputs, often leading to substantial gains in answer quality (Zhang et al., 2025a). These successes motivate recent efforts to extend test-time scaling to tool-augmented agents (Zhu et al., 2025b; Wang et al., 2025a), where LLMs are equipped with various tools to interact with the external environment such as search engines or APIs.

Test-time scaling for tool-augmented agents expands both thinking (tokens) and acting (tool calls). Unlike the token budget in textual reasoning (Han et al., 2025b; Pu et al., 2025), in agent tasks such as web browsing, the number of tool calls directly determines the depth and breadth of exploration, defining the effective boundary of external information access. The challenge is not spending more, but spending wisely: the marginal benefit per tool call is uncertain, so every budget should be spent strategically.

The unique complexity brings up critical research questions in agent test-time scaling: How can tool-augmented agents scale *effectively* by making the best use of given resource budgets? We study this question in a budget-constrained setting, grounding our analysis in search agents equipped with search and browse tools, which are widely used in practice (Google, 2025; OpenAI, 2025) and inherently require extensive tool calls and multi-round interactions to collect external information. In search agents, the major scaling effort is defined by the number of search and browse calls available to the agent. We further evaluate cost with a unified metric that jointly accounts for token consumption and tool-call costs.

Intuitively, keeping the agent updated with the budgets is a simple yet effective step toward better resource use. However, without additional guidance it still tends to perform shallow search (Lu et al.,

2025) and underexplores even when budget remains. To this end, we propose CATS (**C**ost-effective **A**gent **T**est-time **S**caling), a budget-aware framework designed to scale test-time compute for agents effectively and efficiently. CATS maintains a lightweight budget tracker that continuously reports consumed and remaining budgets, and each module adapts its behavior to this signal. The planning module dynamically adjusts stepwise effort to match current budget, while the verification module decides whether to dig deeper along a lead or explore wider with alternative paths accordingly, adapting between sequential and parallel scaling as needed. By steering decisions with explicit budget awareness, CATS enables more deliberate and cost-conscious scaling.

Finally, we empirically study the relationship between overall resource cost and task performance in agent test-time scaling by comparing different scaling frameworks under varying budgets. To ensure fair and transparent comparison, we unify the actual cost of token usage and tool calls into a single cost metric. Our results show that CATS produces more favorable scaling curves: it achieves higher performance while using fewer tool calls and incurring lower overall cost. These findings indicate the clear understanding under the budget constrained setting and demonstrate the potential of budget-aware design for effective tool-augmented agents, highlighting the importance of explicitly accounting for cost in agent test-time scaling.

We summarize our contributions as follows:

- We formalize *budget-constrained* agent test-time scaling with explicit tool-call budgets, and introduce a unified cost that jointly accounts for tokens and tool calls, enabling fair and transparent scaling comparisons.
- We introduce **CATS**, which maintains a lightweight budget tracker and adjusts step-wise effort via budget-aware planning and verification, dynamically switching between deepening a lead and branching to alternatives.
- We conduct systematic experiments under varying budgets and unified costs with search agents, demonstrating that CATS is more cost-effective than comparing methods, yielding more favorable scaling curves and better cost–performance trade-offs.

2 Problem Formulation

2.1 Agent Test-time Scaling

We formulate agent test-time scaling as how an agent's performance scales with its budget for external tool-call interaction, refining the broader concept of test-time interaction scaling as discussed in Shen et al. (2025). To make agent test-time scaling cost-effective, an ideal agent should be able to achieve its best possible performance under an arbitrary budget constraint on the scaling curve. To this end, our target is to design a cost-effective agent framework, π, that maximizes answer accuracy while adhering to a strict tool-call budget.

Assume the agent is equipped with a set of K tools as $\mathcal{T} = \{t_1, \ldots, t_K\}$. For a given question $x \in \mathcal{X}$, the agent works under a budget $\mathbf{b} = (b_1, \ldots, b_K)$, where b_i is the maximum number of invocations of tool $t_i \in \mathcal{T}$. Let $\hat{y}_\pi(x)$ denote the agent's predicted answer for question x with ground truth $y(x)$ and let $c_i(x;\pi)$ be the realized number of calls to tool t_i on x. We formulate the cost-effective agent test-time scaling problem as a budget-constrained optimization objective:

$$\begin{aligned} \max_{\pi} \quad & \mathrm{Acc}_{\mathbf{b}}(\pi) \;=\; \mathbb{E}_x\big[\,\mathbf{1}\{\hat{y}_\pi(x) = y(x)\}\,\big] \\ \text{s.t.} \quad & c_i(x;\pi) \;\le\; b_i \quad \text{for all } i = 1, \ldots, K, \text{ and every } x \in \mathcal{X} \end{aligned} \tag{1}$$

Here the objective is the expected accuracy over all questions. The constraint ensures that for any given question, the number of realized calls for each tool never exceeds its allocated budget. By evaluating the agent performance at various budget levels, we can trace the performance-cost curve, which characterizes the agent's test-time scaling behavior, showing how effectively it leverages budget resources to its problem solving capabilities.

Budget vs. Cost. We distinguish between the preset budget constraint, which specifies the maximum number of tool calls available to the agent, and the realized cost, which reflects the resources actually consumed during execution. While the budget imposes a hard upper limit, the final cost

depends on the agent's strategy. To facilitate a more consistent and comprehensive comparison, we introduce in Section 2.2 a unified post-hoc cost metric for analyzing agents' test-time scaling.

Choice of Budget. Among possible constraints, we prioritize a tool-call budget over a token-based budget for its *relevance*, *consistency*, and *practicability*. A tool-call limit offers a more relevant and direct constraint on an agent's ability to acquire external knowledge than the tokens used for internal reasoning. This choice is also consistent with and justified by established practices in Shen et al. (2025). Furthermore, it offers greater practicability, as it is often non-trivial to pre-determine an appropriate token budget for complex, multi-step agentic tasks. While the budget is defined by tool calls, we still incorporate token usage into our unified cost metric (Section 2.2) to ensure a more fair and transparent comparison.

2.2 Problem Instantiation with Search Agent

In our work, we instantiate the test-time scaling problem with search agent, a setting selected for its broad applicability and the presence of established benchmarks.

A search agent is an LLM that answers an information-seeking question x by retrieving external evidence and reasoning over it. The agent follows an iterative ReAct-style loop (Yao et al., 2023), alternating between internal thinking and external actions. The agent has access to two primary tools for interacting with the world:

- **Search.** This tool helps perform a standard search engine query. Given a text query, it returns a list of search results, each including a title, a brief snippet, and a URL.
- **Browse.** Given a specific URL, this tool scrapes the full content of the corresponding webpage, providing detailed information that is often unavailable in a search snippet.

Unified Cost Metric. We model the agent's total cost as the sum of consumed resources along two dimensions: tokens and tool calls. To create a unified metric, we map both to their corresponding economic costs.

- **Token cost.** This represents the agent's internal cognitive effort, including its reasoning, planning, and parametric knowledge processing. Token costs are calculated based on the pricing of the model provider, distinguishing among input, output, and cache hit tokens. In multi-round iterative frameworks, the output of iteration i becomes part of the input for iteration $i + 1$. Any overlap is a cache hit, thereby lowering the token consumption cost.
- **Tool call cost.** This represents the agent's active interaction with the external environment through information-seeking actions. Each invocation of an external service (e.g., a search query or a browsing request) incurs a cost determined by the pricing of the corresponding API or third-party provider.

The total unified cost, $C_{\textit{unified}}(x;\pi)$, for solving a given question x under policy π is the sum of the token cost and the tool call cost. Let $c_i(x;\pi)$ be the number of actual invocations to tool t_i and P_i be the economic cost per invocation of t_i. The unified cost metric is then defined as:

$$C_{\textit{unified}}(x;\pi) = \underbrace{c_{\textit{token}}(x;\pi)}_{\text{Token Cost}} + \underbrace{\sum_{i=1}^{K} c_i(x;\pi) \cdot P_i}_{\text{Total Tool Cost}} \tag{2}$$

Here, $c_{\textit{token}}(x;\pi)$ represents the total cost incurred from token consumption during the agent's reasoning process for question x. This formulation allows us to uniformally analyze the actual incurred costs for each execution. These two dimensions are inherently coupled: additional tool calls generally increase token consumption, as the agent must process and reason over the retrieved external information. Measuring this unified cost under varying budget constraints, $\mathbf{b}$, enables a more comprehensive and consistent analysis across different policies.

3 CATS: Cost-effective Agent Test-time Scaling

We propose **CATS**, a framework for scaling test-time resources for tool-augmented agents under explicit budget. As shown in Figure 1, given an information-seeking question and a tool call budget,

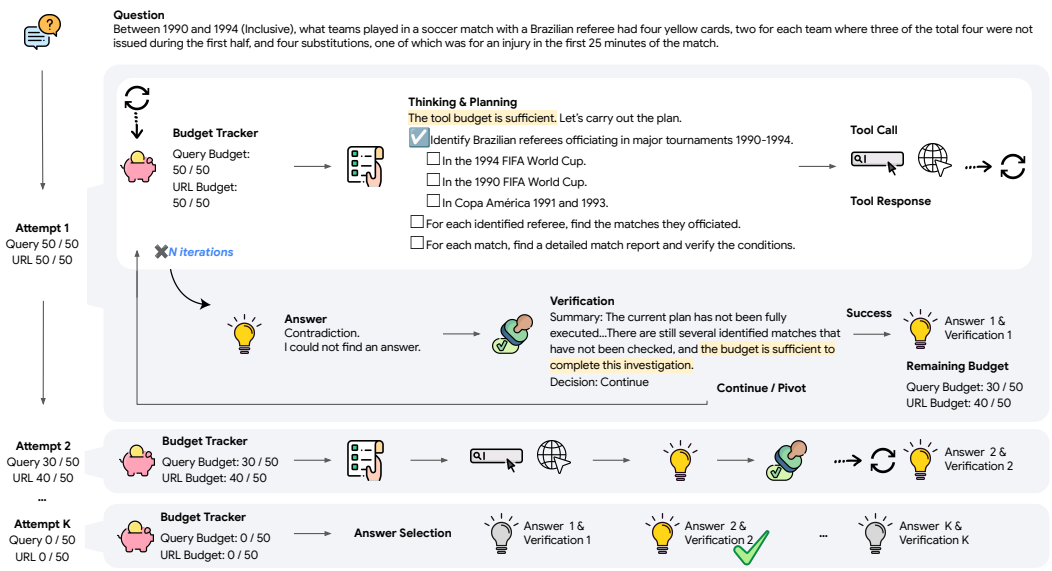


Figure 1: Overview of the CATS framework. Given a question and per-tool budgets, the agent begins with budget-aware thinking and planning, structured as a checklist. Each tool call consumes part of the assigned budget, and the agent keeps iterating by reasoning over new information and updated budgets. When an answer is proposed, CATS performs verification and decides to either continue, pivot, or proceed to a new attempt with the remaining budget. CATS terminates when any of the budgets are exhausted.

CATS first engages in internal thinking to decompose the constraints specified in the question and formulate a structured plan of actions. Then the agent invokes tools by generating specific tokens that trigger external APIs for search and browse actions. For the search tool, the agent issues queries and receives candidate results. For the browse tool, an LLM processes the scraped webpage content and produces a tailored web content summary and relevant findings. These outputs are appended to the working sequence as tool responses, expanding the context with newly retrieved evidence. A budget tracker is updated after each tool call, recording the resources used and the remaining budget.

Whenever the agent proposes a candidate answer, the verification module checks its validity and decides whether to continue with the current sequence or start a new attempt with the remaining budget. CATS terminates when any budgeted resource is exhausted. Finally, an LLM-as-a-judge selects the best answer by evaluating all verified answers, along with their corresponding verification details, from each attempt, to produce the final output.

The central design principle of CATS is *budget awareness*: the agent is continually updated on its remaining budget in every iteration of execution. This persistent awareness not only adapts its thinking, planning, and verification behaviors to the budget but also encourages the agent to make full and effective use of its available resources.

3.1 Budget Tracker

An explicit budget block records both usage and remaining resources throughout execution, updating after each iteration. Beyond guiding decision-making, explicit budget tracking enforces effective resource utilization. By making budget status transparent, the agent is encouraged to balance the usage of different types of tools, fully leveraging the resources provided.

3.2 Budget-Aware Planning

Planning in CATS incorporates both *constraint decomposition* and *structured dynamic planning*. Information-seeking tasks require strategic planning to decompose the problem and allocate the

budget to navigate the large search space efficiently. Selecting an appropriate starting point is critical: a well-chosen entry narrows the search space and conserves budget, while a poor choice can quickly exhaust the budget. To address this, we prompt the agent to first perform **constraint decomposition** and to categorize the clues implied in the question into two types: *exploration* and *verification*. Exploration clues open new directions or candidate sets, whereas verification clues help confirm details or filter among candidates. While a verification clue can sometimes provide a direct shortcut to the answer, relying on it prematurely is risky, as it may consume resources without guaranteeing progress.

The agent is further instructed to generate and maintain an explicit plan throughout execution. This **tree-structured plan** acts as a dynamic checklist, recording step status, resource usage, and allocation, while guiding future actions. As shown in the planning block from Figure 1, a single step in the plan represents a subtask that may require multiple tool calls to complete, for instance, several searches followed by browsing and filtering to get a candidate list. The planning module then adapts its exploration strategy to the budget: with more budget available, it can afford to explore multiple complementary sources in parallel, whereas with limited budget, it must strategically prioritize the most promising candidates. During execution, the agent continually revises the plan to reflect newly acquired information, prune unproductive paths, and redirect progress toward promising leads.

By integrating constraint decomposition with budget-aware dynamic planning, the agent is able to balance exploration and verification within a controllable search space, ensuring both efficiency and reliability in information-seeking tasks.

3.3 Budget-Aware Self-verification

Once the agent proposes an answer, the verification module revisits the trajectory and budget usage to check its correctness. This process begins with a backward verification of the reasoning and tool calling trajectory, evaluating each constraint specified in the question. For each of them, the module determines whether it has been satisfied, contradicted, or remains unverifiable. This retrospective reasoning systematically grounds the answer against the original requirements and clues.

Based on this analysis, the module produces a verification decision. If all constraints are satisfied, the answer is marked as a *success*. If several constraints remain unverifiable but the trajectory appears promising, provided the budget is sufficient for deeper exploration, the outcome is to *continue* exploration. In contrast, if contradictions are identified or the remaining budget cannot support further investigation towards this lead, the agent is expected to terminate expensive or low-yield directions early and *pivot* toward a different direction to avoid wasting tool call resources while resources are still sufficient to pursue alternatives.

When the decision is to *continue* or *pivot*, the module also generates a concise summary of the trajectory so far. This includes the reasoning history, intermediate conclusions, and suggestions for optimization to avoid redundant exploration. Moreover, this summary serves as a condensed context, replacing the previous lengthy trajectory to reduce token usage. By compressing the reasoning state into a shorter form, the module ensures that subsequent exploration is not only better informed but also more cost-effective in terms of context length. Together, budget-aware verification and trajectory summarization allow the agent to balance correctness, resource efficiency, and context management, ensuring reliable progress within budget constraints.

4 Experiments

4.1 Setup

Datasets. To benchmark web search agents, we employ the challenging information-seeking datasets BrowseComp (Wei et al., 2025) and BrowseComp-zh (Zhou et al., 2025).

Baselines. We compare CATS against a range of models and agentic frameworks, including both general-purpose base models (Hurst et al., 2024; Jaech et al., 2024; Comanici et al., 2025; Anthropic, 2025a;b) and those specifically fine-tuned for agentic search tasks (Li et al., 2025a; Liu et al., 2025a; Gao et al., 2025; Lu et al., 2025). To evaluate the final answer accuracy, we use Gemini-2.5-Flash as the judge model and adopt the evaluation prompt from Phan et al. (2025).

For scaling methods, we evaluate sequential and parallel scaling approaches applied to the ReAct (Yao et al., 2023) baseline. For **sequential scaling**, to encourage the agent to use more tools during its iterative execution, we follow the approach from textual reasoning (Muennighoff et al., 2025) and append the sentence, "Wait, I must use both search and browse tools more before generating the final answer. Let me rethink.", whenever the agent provides an answer. This process is repeated until the agent's tool budget is exhausted, after which it is prompted to produce the final answer. For **parallel scaling**, we use temperature sampling to generate diverse reasoning paths. To aggregate the results, we use Gemini-2.5-Flash to select the most common answer via a majority vote (Wang et al., 2023). To enforce the budget constraint, we sample additional sequences until the budget is exhausted. Thus the number of sampled sequences may vary across different questions.

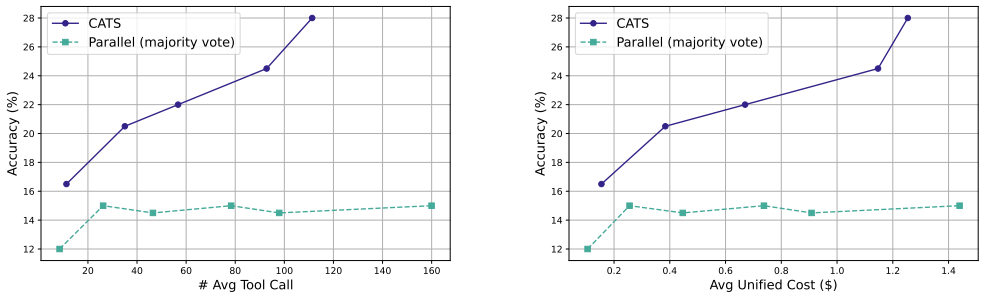
4.2 Implementation Details

We use Gemini-2.5-Flash and Gemini-2.5-Pro (Comanici et al., 2025) as the default backbone models in our framework. By default, we disable the thinking mode by setting the thinking budget as 0 for flash and 1024 for pro models. The maximum number of new tokens for generation was set to 65,536. We use a temperature of 0.7 during agent execution to encourage exploration, and use a deterministic temperature of 0.0 for final answer selection and answer evaluation. We use the Google Custom Search JSON API for search tools, and Jina.ai[1] for web browsing.

5 Results

5.1 Main Results

Table 1 shows the performance comparison across different web search agents. Under the strict budget constraints of 100 tool uses for BrowseComp and 50 for BrowseComp-zh, CATS consistently achieves better results than other baselines, obtaining 21.5% on BrowseComp and 41.9% on BrowseComp-zh using Gemini-2.5-Pro. This indicates the effectiveness of our budget-aware design in maximizing the efficiency of every tool call.



(a) Scaling curve under budget constraints.

(b) Scaling curve against average unified cost.

Figure 2: Scaling behaviors along (a) total number of tool calls and (b) average unified cost, evaluated on a 200-example subset of BrowseComp using Gemini-2.5-Pro.

CATS achieves higher performance under the same budget constraint. To better understand the scaling behavior, we vary the tool-call budget and evaluate performance on a random subset of 200 examples from BrowseComp for a manageable analysis. Figure 2a shows the accuracy against the average number of tool calls, including both search and browse. Across all budget levels, CATS consistently outperforms the parallel majority-vote baseline, demonstrating that budget-aware adaptation leads to more effective use of limited tool calls.

CATS achieves higher performance when accounting for unified costs. Beyond tool-call counts, we measure performance under a unified cost metric that incorporates both token usage and tool-call expenses. As shown in Figure 2b, CATS achieves more favorable scaling curves, delivering higher

[1] https://jina.ai/

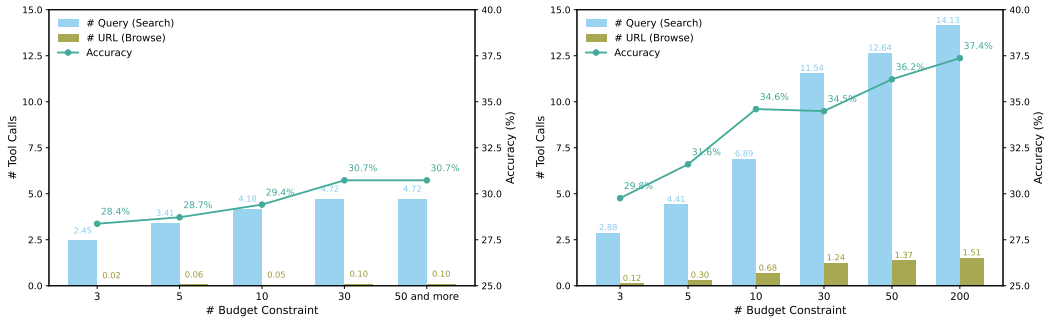
Table 1: Performance comparison across web search agents. We denote results from our own experiments with ⋆; other baseline scores are cited from their respective publications. The "Training" column specifies whether the model has been specifically trained on agentic web search tasks. For our budget-constrained setting, BrowseComp is provided a budget of 100 tool uses per tool, while BrowseComp-zh is limited to 50.

Method	**Training**	**BrowseComp**	**BrowseComp-zh**
Model Only			
GPT-4o	✗	0.6	6.2
Claude-3.7-Sonnet	✗	2.3	11.8
Gemini-2.5-Flash⋆	✗	2.7	23.9
Gemini-2.5-Pro⋆	✗	6.3	27.8
OpenAI o1	✗	9.9	29.1
Agentic Framework			
OpenAI Deep research	✓	51.5	42.9
ASearcher	✓	5.2	15.6
WebSailor	✓	12.0	30.1
DeepDive	✓	14.8	25.6
WebExplorer	✓	15.7	32.0
Budget-constrained			
Flash-Baseline	✗	10.3	29.4
Flash-Sequential	✗	13.5	29.8
Flash-Parallel	✗	14.1	29.8
Flash-CATS	✗	16.5	35.3
Pro-Baseline	✗	8.4	30.7
Pro-Sequential	✗	9.5	30.5
Pro-Parallel	✗	13.1	37.0
Pro-CATS	✗	21.5	41.9

accuracy at comparable or lower costs. This indicates that CATS not only improves effectiveness under budget constraints but also yields better cost–performance trade-offs.

5.2 Early Stopping

In this section, we evaluate how effectively agents perform under various budget constraints without requiring it to exhaust all available resources. We analyze the performance of the agent's first attempt: for CATS, this is the first answer that passes its internal verification, while for the baseline, it corresponds to a single generation pass. For both methods, if any tool budget is exhausted, the agent is prompted to immediately generate a final answer based on its progress.



(a) Baseline on BrowseComp-zh.

(b) CATS on BrowseComp-zh.

Figure 3: Comparison of early stopping on BrowseComp-zh.

Figure 3 compares the performance on the BrowseComp-zh dataset using Gemini-2.5-Pro. The x-axis represents the predefined budget for both search and browse tool calls. The bars indicate the average number of tool calls used, while the line plot shows the resulting first-attempt accuracy.

Budget awareness allows CATS to scale effectively with increased resources. The baseline (Figure 3a) demonstrates poor resource management. It consistently underutilizes the browse tool (fewer than 0.1 calls on average) and shows diminishing returns early, with accuracy stagnating at 30.7% for all budgets of 30 and above. This indicates a lack of budget awareness, preventing it from leveraging increased resources. Conversely, CATS (Figure 3b) shows the advantage of budget-aware framework. It strategically increases its use of both search and browse tools as the budget expands. This balanced approach leads to a sustained improvement in accuracy, rising from 29.8% (budget=3) to 37.4% (budget=200). Notably, CATS's performance with a small budget of 5 already surpasses the baseline's maximum achievable accuracy. This underscores CATS's ability to make more strategic and cost-effective decisions, achieving better results with the same or even fewer resources.

CATS is more cost-effective by enabling early stopping. To provide a direct and transparent comparison of cost-efficiency, Figure 4 shows accuracy against the actual unified cost. CATS demonstrates a much steeper performance curve, indicating that it achieves higher accuracy for a lower cost compared to the parallel majority vote baseline. CATS reaches over 37% accuracy for approximately $0.23, while the parallel baseline requires more than double that cost (over $0.50) to achieve a comparable result. This efficiency benefits from its budget-aware verification module, which enables early termination upon finding a satisfactory answer and minimizes unnecessary spending.

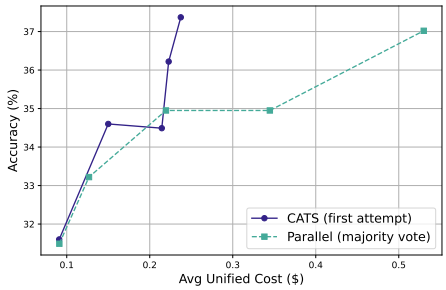


Figure 4: Average unified cost analysis on BrowseComp-zh using Gemini-2.5-Pro.

5.3 Analysis

5.3.1 Planning Module

Table 2: Effect of the planning module. Results are averaged over three runs on a 200-example subset of BrowseComp. With the same budget, the planning module encourages greater tool usage and yields higher average performance.

Method	Accuracy	Avg. # Query	Avg. # URL
ReAct	11.0	7.75	0.35
ReAct + planning	12.8	13.81	0.82

To evaluate the impact of the planning module, we augment the ReAct baseline with our proposed design, which integrates constraint analysis and a dynamically structured checklist plan. The tool-call budget is capped at 200 for both search queries and browse URLs. As shown in Table 2, the addition of planning module alone improves the agent's ability to organize exploration and utilize tool usage more effectively, resulting in a performance gain of 1.8%.

5.4 Budget Tracker

Table 3: Effect of the budget tracker. Results are averaged over three runs on a 200-example subset of BrowseComp.

Method	Accuracy	# search	# browse
ReAct	11.0	7.75	0.35
ReAct + Budget Tracker	12.7	9.31	0.89
ReAct + Sequential Scaling	12.7	16.93	0.78

We next examine the effect of augmenting the ReAct baseline with a budget tracker under a fixed budget of 200 search and 200 browse calls. As shown in Table 3, adding the budget tracker improves

accuracy from 11.0% to 12.7%, while only modestly increasing tool usage. By comparison, sequential scaling also raises accuracy to 12.7%, but does so by nearly doubling the number of search calls (from 7.8 to 16.9). This contrast highlights the efficiency of budget awareness: the budget tracker achieves comparable performance to sequential scaling with significantly fewer tool calls. Rather than relying on sheer volume of actions, the tracker enables more effective utilization of available tools, making it a more cost-effective approach to scaling.

6 Related Work

6.1 Test-time Scaling

Test-time scaling (TTS) (Snell et al., 2024; Zhang et al., 2025a) strategies typically fall into two categories. The first is sequential scaling, where a model iteratively refines its output based on self feedback or reflection (Madaan et al., 2023; Zhang et al., 2025b; Muennighoff et al., 2025; Liu et al., 2025b). The second category is parallel scaling, where multiple reasoning paths are sampled and an aggregation strategy is used to determine the final answer (Brown et al., 2024; Wang et al., 2023). Further, hybrid scaling attempts to combine their complementary benefits (Chen et al., 2025a;b; Wan et al., 2025; Li et al., 2024). While prior work focuses on text-only reasoning, we extend TTS to tool-augmented agents, where scaling accounts for both tokens and tool calls under budget constraints. As these methods push performance by increasing computation, a complementary line of work examines how to constrain the effort. Typical constraints are defined over tokens, sampled sequences, or FLOPs (Nayab et al., 2024; Welleck et al., 2024; Damani et al., 2025; Pu et al., 2025). Specifically, AgentTTS (Wang et al., 2025a) optimizes LLM size and sampling numbers under a unified FLOPs budget. In contrast, we formalize and constrain the tool-call budget, shifting the focus from token-related limits in text reasoning to cost-effective scaling of tool-augmented agents.

6.2 Web Search Agents

Web search agents use search and browse tools to solve complex, multi-hop queries (Chen et al., 2025c; Wong et al., 2025; Team et al., 2025; Han et al., 2025a; Team, 2025). One research direction builds training data and applies various training methods to specialize the models (Jin et al., 2025; Li et al., 2025a; Liu et al., 2025a; Tao et al., 2025). Another explores inference-time strategies (Li et al., 2025b; Zhu et al., 2025a; Qiao et al., 2025), such as incorporating programmatic execution to perform multiple tool call actions (Pang et al., 2025), finding an optimal, statically efficient configuration (Wang et al., 2025b), or exploring various design choices when scaling test-time compute (Zhu et al., 2025b). Instead, our work focuses on dynamic, cost-effective performance, providing the first analysis of agent scaling behavior under explicit budget constraints.

7 Conclusion

in this paper, we investigate cost-effective test-time scaling of agents under budget constraints. We formulate the problem by introducing explicit tool-call budgets and a unified cost metric that captures overall resource consumption. To address this setting, we propose CATS, a cost-effective agent test-time scaling framework that employs a budget tracker to guide planning and verification, enabling it to make adaptive and strategic decisions. Experiments show that CATS obtains more cost-effective scaling curve under constrained budgets. The budget-awareness design of CATS advances more effective and efficient agent test-time scaling, enabling the development of cost-conscious, adaptive, and practical tool-augmented agents.

The Use of Large Language Models (LLMs)

We acknowledge the use of LLMs (ChatGPT and Gemini) exclusively for editing the text to correct grammatical errors and improve clarity and flow. All core scientific content and research ideas were authored solely by the human authors.

Reproducibility Statement

We conduct evaluations exclusively on publicly available benchmarks and query LLMs through public providers. All experimental settings, including temperature, context length, tool calling, and other configuration parameters, are detailed in Section 4.2.

Ethics Statement

The authors confirm adherence to the ICLR Code of Ethics. This work aims to reduce the computational and economic costs of LLM agents, contributing positively to accessibility and sustainability. At the same time, we recognize that web search agents may inherit web-based biases or propagate misinformation. Addressing these risks requires our continued attention and responsible research practices within the research community.

References

Anthropic. Claude 3.7 sonnet system card. https://assets.anthropic.com/m/785e231869ea8b3b/original/claude-3-7-sonnet-system-card.pdf, 2025a.

Anthropic. Claude 4 system card: Claude opus 4 & claude sonnet 4. https://www-cdn.anthropic.com/6d8a8055020700718b0c49369f60816ba2a7c285.pdf, 2025b.

Bradley C. A. Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *CoRR*, abs/2407.21787, 2024. doi: 10.48550/ARXIV.2407.21787. URL https://doi.org/10.48550/arXiv.2407.21787.

Jianhao Chen, Zishuo Xun, Bocheng Zhou, Han Qi, Qiaosheng Zhang, Yang Chen, Wei Hu, Yuzhong Qu, Wanli Ouyang, and Shuyue Hu. Do we truly need so many samples? multi-llm repeated sampling efficiently scales test-time compute. *CoRR*, abs/2504.00762, 2025a. doi: 10.48550/ARXIV.2504.00762. URL https://doi.org/10.48550/arXiv.2504.00762.

Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, and Sercan Ö. Arik. SETS: leveraging self-verification and self-correction for improved test-time scaling. *CoRR*, abs/2501.19306, 2025b. doi: 10.48550/ARXIV.2501.19306. URL https://doi.org/10.48550/arXiv.2501.19306.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *CoRR*, abs/2508.06600, 2025c. doi: 10.48550/ARXIV.2508.06600. URL https://doi.org/10.48550/arXiv.2508.06600.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit S. Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, Kornraphop Kawintiranon, Tania Bedrax-Weiss, Oliver Wang, Ya Xu, Ollie Purkiss, Uri Mendlovic, Ilaï Deutel, Nam Nguyen, Adam Langley, Flip Korn, Lucia Rossazza, Alexandre Ramé, Sagar Waghmare, Helen Miller, Nathan Byrd, Ashrith Sheshan, Raia Hadsell Sangnie Bhardwaj, Pawel Janus, Tero Rissa, Dan Horgan, Sharon Silver, Ayzaan Wahid, Sergey Brin, Yves Raimond, Klemen Kloboves, Cindy Wang, Nitesh Bharadwaj Gundavarapu, Ilia Shumailov, Bo Wang, Mantas Pajarskas, Joe Heyward, Martin Nikoltchev, Maciej Kula, Hao Zhou, Zachary Garrett, Sushant Kafle, Sercan Arik, Ankita Goel, Mingyao Yang, Jiho Park, Koji Kojima, Parsa Mahmoudieh, Koray Kavukcuoglu, Grace Chen, Doug Fritz, Anton Bulyenov, Sudeshna Roy, Dimitris Paparas, Hadar Shemtov, Bo-Juen Chen, Robin Strudel, David Reitter, Aurko Roy, Andrey Vlasov, Changwan Ryu, Chas Leichner, Haichuan Yang, Zelda Mariet, Denis Vnukov,

Tim Sohn, Amy Stuart, Wei Liang, Minmin Chen, Praynaa Rawlani, Christy Koh, JD Co-Reyes, Guangda Lai, Praseem Banzal, Dimitrios Vytiniotis, Jieru Mei, and Mu Cai. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025. doi: 10.48550/ARXIV.2507.06261. URL https://doi.org/10.48550/arXiv.2507.06261.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=6qUUgw9bAZ.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *CoRR*, abs/2508.07976, 2025. doi: 10.48550/ARXIV.2508.07976. URL https://doi.org/10.48550/arXiv.2508.07976.

Google. Gemini deep research — your personal research assistant. https://gemini.google/overview/deep-research/, 2025.

Rujun Han, Yanfei Chen, Zoey CuiZhu, Lesly Miculicich, Guan Sun, Yuanjun Bi, Weiming Wen, Hui Wan, Chunfeng Wen, Solène Maître, George Lee, Vishy Tirumalashetty, Emily Xue, Zizhao Zhang, Salem Haykal, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. Deep researcher with test-time diffusion. *CoRR*, abs/2507.16075, 2025a. doi: 10.48550/ARXIV.2507.16075. URL https://doi.org/10.48550/arXiv.2507.16075.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 24842–24855. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.findings-acl.1274/.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll L. Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, and Dane Sherburn. Gpt-4o system card. *CoRR*, abs/2410.21276, 2024. doi: 10.48550/ARXIV.2410.21276. URL https://doi.org/10.48550/arXiv.2410.21276.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan

Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, and Ilge Akkaya. Openai o1 system card. *CoRR*, abs/2412.16720, 2024. doi: 10.48550/ARXIV.2412.16720. URL https://doi.org/10.48550/arXiv.2412.16720.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor: Navigating superhuman reasoning for web agent. *CoRR*, abs/2507.02592, 2025a. doi: 10.48550/ARXIV.2507.02592. URL https://doi.org/10.48550/arXiv.2507.02592.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *CoRR*, abs/2501.05366, 2025b. doi: 10.48550/ARXIV.2501.05366. URL https://doi.org/10.48550/arXiv.2501.05366.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=ndR8Ytrzhh.

Junteng Liu, Yunji Li, Chi Zhang, Jingyang Li, Aili Chen, Ke Ji, Weiyu Cheng, Zijia Wu, Chengyu Du, Qidi Xu, Jiayuan Song, Zhengmao Zhu, Wenhu Chen, Pengyu Zhao, and Junxian He. Webexplorer: Explore and evolve for training long-horizon web agents. 2025a. URL https://api.semanticscholar.org/CorpusID:281204359.

Licheng Liu, Zihan Wang, Linjie Li, Chenwei Xu, Yiping Lu, Han Liu, Avirup Sil, and Manling Li. A simple "try again" can elicit multi-turn LLM reasoning. *CoRR*, abs/2507.14295, 2025b. doi: 10.48550/ARXIV.2507.14295. URL https://doi.org/10.48550/arXiv.2507.14295.

Rui Lu, Zhenyu Hou, Zihan Wang, Hanchen Zhang, Xiao Liu, Yujiang Li, Shi Feng, Jie Tang, and Yuxiao Dong. Deepdive: Advancing deep search agents with knowledge graphs and multi-turn rl, 2025. URL https://arxiv.org/abs/2509.10446.

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 Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *CoRR*, abs/2501.19393, 2025. doi: 10.48550/ARXIV.2501.19393. URL https://doi.org/10.48550/arXiv.2501.19393.

Sania Nayab, Giulio Rossolini, Giorgio C. Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. Concise thoughts: Impact of output length on LLM reasoning and cost. *CoRR*, abs/2407.19825, 2024. doi: 10.48550/ARXIV.2407.19825. URL https://doi.org/10.48550/arXiv.2407.19825.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025.

Xianghe Pang, Shuo Tang, Rui Ye, Yuwen Du, Yaxin Du, and Siheng Chen. Browsemaster: Towards scalable web browsing via tool-augmented programmatic agent pair. *CoRR*, abs/2508.09129, 2025. doi: 10.48550/ARXIV.2508.09129. URL https://doi.org/10.48550/arXiv.2508.09129.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, Daron Anderson, Tung Nguyen, Mobeen Mahmood, Fiona Feng, Steven Y. Feng, Haoran Zhao, Michael Yu, Varun Gangal, Chelsea Zou, Zihan Wang, Jessica P. Wang, Pawan Kumar, Oleksandr Pokutnyi, Robert Gerbicz, Serguei Popov, John-Clark Levin, Mstyslav Kazakov, Johannes Schmitt, Geoff Galgon, Alvaro Sanchez, Yongki Lee, Will Yeadon, Scott Sauers, Marc Roth, Chidozie Agu, Søren Riis, Fabian Giska, Saiteja Utpala, Zachary Giboney, Gashaw M. Goshu, Joan of Arc Xavier, Sarah-Jane Crowson, Mohinder Maheshbhai Naiya, Noah Burns, Lennart Finke, Zerui Cheng, Hyunwoo Park, Francesco Fournier-Facio, John Wydallis, Mark Nandor, Ankit Singh, Tim Gehrunger, Jiaqi Cai, Ben McCarty, Darling Duclosel, Jungbae Nam, Jennifer Zampese, Ryan G. Hoerr, Aras Bacho, Gautier Abou Loume, Abdallah Galal, Hangrui Cao, Alexis C. Garretson, Damien Sileo, Qiuyu Ren, Doru Cojoc, Pavel Arkhipov, Usman Qazi, Lianghui Li, Sumeet Motwani, Christian Schröder de Witt, Edwin Taylor, Johannes Veith, Eric Singer, Taylor D. Hartman, Paolo Rissone, Jaehyeok Jin, Jack Wei Lun Shi, Chris G. Willcocks, Joshua Robinson, Aleksandar Mikov, Ameya Prabhu, Longke Tang, Xavier Alapont, Justine Leon Uro, Kevin Zhou, Emily de Oliveira Santos, Andrey Pupasov Maksimov, Edward Vendrow, Kengo Zenitani, Julien Guillod, Yuqi Li, Joshua Vendrow, Vladyslav Kuchkin, and Ng Ze-An. Humanity's last exam. *CoRR*, abs/2501.14249, 2025. doi: 10.48550/ARXIV.2501.14249. URL https://doi.org/10.48550/arXiv.2501.14249.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. THOUGHTTERMINATOR: benchmarking, calibrating, and mitigating overthinking in reasoning models. *CoRR*, abs/2504.13367, 2025. doi: 10.48550/ARXIV.2504.13367. URL https://doi.org/10.48550/arXiv.2504.13367.

Zile Qiao, Guoxin Chen, Xuanzhong Chen, Donglei Yu, Wenbiao Yin, Xinyu Wang, Zhen Zhang, Baixuan Li, Huifeng Yin, Kuan Li, Rui Min, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webresearcher: Unleashing unbounded reasoning capability in long-horizon agents. 2025. URL https://api.semanticscholar.org/CorpusID:281325175.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, and Aviral Kumar. Thinking vs. doing: Agents that reason by scaling test-time interaction. *CoRR*, abs/2506.07976, 2025. doi: 10.48550/ARXIV.2506.07976. URL https://doi.org/10.48550/arXiv.2506.07976.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/ARXIV.2408.03314. URL https://doi.org/10.48550/arXiv.2408.03314.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webshaper: Agentically data synthesizing via information-seeking formalization. *CoRR*, abs/2507.15061, 2025. doi: 10.48550/ARXIV.2507.15061. URL https://doi.org/10.48550/arXiv.2507.15061.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Tongyi DeepResearch Team. Tongyi-deepresearch. https://github.com/Alibaba-NLP/DeepResearch, 2025.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 3613–3635. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.184. URL https://doi.org/10.18653/v1/2025.naacl-long.184.

Fali Wang, Hui Liu, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Zongyu Wu, Chen Luo, Zheng Li, Xianfeng Tang, Qi He, and Suhang Wang. Agenttts: Large language model agent for test-time compute-optimal scaling strategy in complex tasks. *CoRR*, abs/2508.00890, 2025a. doi: 10.48550/ARXIV.2508.00890. URL https://doi.org/10.48550/arXiv.2508.00890.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Efficient agents: Building effective agents while reducing cost. *CoRR*, abs/2508.02694, 2025b. doi: 10.48550/ARXIV.2508.02694. URL https://doi.org/10.48550/arXiv.2508.02694.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *CoRR*, abs/2504.12516, 2025. doi: 10.48550/ARXIV.2504.12516. URL https://doi.org/10.48550/arXiv.2504.12516.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaïd Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=eskQMcIbMS.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. Widesearch: Benchmarking agentic broad info-seeking. *CoRR*, abs/2508.07999, 2025. doi: 10.48550/ARXIV.2508.07999. URL https://doi.org/10.48550/arXiv.2508.07999.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=VNckp7JEHn.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. What, how, where, and how well? A survey on test-time scaling in large language models. *CoRR*, abs/2503.24235, 2025a. doi: 10.48550/ARXIV.2503.24235. URL https://doi.org/10.48550/arXiv.2503.24235.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew C. Yao. Cumulative reasoning with large language models. *Trans. Mach. Learn. Res.*, 2025, 2025b. URL https://openreview.net/forum?id=grW15p4eq2.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *CoRR*, abs/2504.19314, 2025. doi: 10.48550/ARXIV.2504.19314. URL https://doi.org/10.48550/arXiv.2504.19314.

He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang, Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, Jian Yang, Minghao Liu, Xitong Gao, Jiaheng Liu, and Wangchunshu Zhou. Oagents: An empirical study of building effective agents. *CoRR*, abs/2506.15741, 2025a. doi: 10.48550/ARXIV.2506.15741. URL https://doi.org/10.48550/arXiv.2506.15741.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for LLM agents. *CoRR*, abs/2506.12928, 2025b. doi: 10.48550/ARXIV.2506.12928. URL https://doi.org/10.48550/arXiv.2506.12928.

A Resource Details

The cost of tool calls is determined by the pricing of the providers. To standardize billing, we've established a unified rate of $0.001 USD per invocation for both search API calls and web browsing actions. The consumption of tokens is billed separately, adhering to the official pricing models of the API provider pricing[2].

[2] https://cloud.google.com/vertex-ai/generative-ai/pricing