ANYSEARCHER: LEARNING TO SEARCH UNDER ARBITRARY INFERENCE BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language model agents increasingly use test-time computation to seek information, interact with tools, and iteratively refine their answers. Yet in real-world settings, an agent rarely operates under a fixed computational endpoint: user patience, latency requirements, tool availability, resource constraints, and task urgency may change while an agent is running. A desirable agent should therefore exhibit an *anytime property*, remaining reliable whenever its computation is interrupted while continuing to benefit from additional computation when it remains available. Existing approaches typically optimize agents for predefined inference budgets or terminal performance, making them less suited to such dynamic conditions. We introduce *AnySearcher*, a single search policy trained with an *AnyBudget* objective that optimizes expected task performance over possible interruption points along the search process. This encourages the agent to be correct at earlier interruption points and remain correct as additional search is performed. Across four challenging search benchmarks, AnySearcher achieves higher accuracy across inference budgets, reaches high performance with fewer tool interactions, and better preserves correctness as additional computation becomes available, demonstrating stronger anytime reliability throughout the process.

1 INTRODUCTION

Large language model agents have rapidly emerged as a powerful paradigm for solving complex real-world tasks, including information seeking, web navigation, scientific reasoning, coding, and decision-making (Yao et al., 2022; Zhou et al., 2024; Boiko et al., 2023; Yang et al., 2024). A key source of their capability is the ability to interact with external environments through tools such as search engines, browsers, code interpreters, and APIs (Nakano et al., 2021; Schick et al., 2023). Recent advances in test-time scaling have shown that allocating additional inference computation can substantially improve model reasoning (Snell et al., 2024; Muennighoff et al., 2025). For tool-augmented agents, this additional computation may be leveraged by repeated interaction with the environment, i.e. additional tool calls that gather more evidence, explore alternative hypotheses, and iteratively refine the agent's solutions (Shen et al., 2025).

How much compute agents should use, however, is situation-dependent. Computing an exact or optimal solution to a user query may be computationally infeasible, economically undesirable, or incompatible with the time available for making a decision (Horvitz, 2013; Dean et al., 1988; Zilberstein, 1996). This issue becomes particularly important in interactive environments, where task states, resource constraints, and the urgency of a decision may change during execution. A model deployed as decision support agent may deliberate extensively while a patient's condition is stable, but may be prompted to act immediately on its best current assessment when conditions deteriorates. An embodied robotics agent should continue to refine a motion plan while computation is available, yet should remain capable of producing a feasible action at any time (Mishani et al., 2024). Coding and web agents similarly operate under changing execution resources, tool availability, costs, and environmental constraints (Kim et al., 2026; Ma et al., 2026; Lin et al., 2026; Liu et al., 2026), the most pressing of which may simply be user patience. These settings motivate an *anytime* view of agentic computation: ideally deployed agents should remain useful when their computation is interrupted, while additional computation should preserve or improve a solution already available (Dean et al., 1988; Zilberstein, 1996).

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.

Modern agents do not yet reliably exhibit these anytime properties. Existing agentic test-time scaling approaches are primarily optimized for performance at predefined computational endpoints, while budget-aware methods adapt inference to specified resource constraints (Qin et al., 2026; Ou et al., 2026; Aggarwal & Welleck, 2025; Liu et al., 2025a). Although these approaches improve the use of inference computation, they do not directly address the value of computation before the prescribed endpoint is reached. *Anytime inference* would make these choices flexible by allowing computation to be interrupted without diminishing the value of searches already performed. Yet, achieving this property remains challenging for current search agents. Correct answers may emerge only after extensive tool interactions, while continued search may overturn previously correct solutions through misleading evidence or unnecessary revisions (Jazbec et al., 2023; Yan et al., 2021; Zhou et al., 2026). Consequently, strong final performance alone does not ensure that computation maximizes useful and reliable answers all throughout the search process.

To address these challenges, we introduce **AnySearcher**, a single search policy designed for settings where the available inference budget is unknown in advance. As illustrated in fig. 1(a), we formulate an *AnyBudget* objective that optimizes expected task performance over possible interruption budgets along the same search trajectory. This objective can be decomposed into first-success utility and correctness regression (section 3.2), providing the basis for our training procedure. As shown in fig. 1(b), we optimize this objective by evaluating sampled trajectories at multiple interruption checkpoints and constructing trajectory-level rewards from the resulting correctness signals. We further use capability-adaptive training to balance terminal task success and AnyBudget supervision according to query solvability (section 3.3). Across multiple search benchmarks, AnySearcher improves performance across inference budgets, reaches strong accuracy with fewer tool interactions, and reduces correctness regressions during continued search.

Our contributions can be summarized as follows:

- We introduce *AnySearcher*, a single search policy for settings where the available inference budget is unknown in advance, together with an *AnyBudget* objective that optimizes answer quality across possible interruption budgets.
- We characterize the AnyBudget objective through an exact decomposition into first-success utility and correctness regression, and develop an interruption-aware training procedure with capability-adaptive reinforcement learning to optimize this objective in practice.
- Across multiple search benchmarks, we show that AnySearcher consistently improves anytime performance across inference budgets, achieves strong accuracy with fewer tool interactions, and better preserves correctness during continued search.

2 Related Work

Web Search Agents. Web search agents augment language models with search and browsing tools to solve complex, multi-hop information-seeking tasks. Early work such as WebGPT (Nakano et al., 2021) and ReAct (Yao et al., 2022) established browser-assisted question answering and interleaved reasoning with tool use. More recent methods specialize language models for search through supervised fine-tuning and reinforcement learning, including Search O1 (Li et al., 2025), Search R1 (Jin et al., 2025), ReSearch (Chen et al., 2025), DeepResearcher (Zheng et al., 2025), and ZeroSearch (Sun et al., 2025). Together with broader advances in reasoning reinforcement learning (Guo et al., 2025), these studies show that search behavior can be substantially improved through post-training.

Test Time Scaling. Test time scaling improves model performance by allocating additional computation during inference (Snell et al., 2024). For reasoning models, additional computation is typically expressed through longer reasoning traces, repeated sampling, or iterative refinement (Madaan et al., 2023; Muennighoff et al., 2025). Recent work extends this paradigm to tool-augmented agents, where additional computation can also take the form of interactions with external environments. Scaling Test Time Compute for LLM Agents (Zhu et al., 2025) studies sequential and parallel scaling strategies, while Thinking vs. Doing (Shen et al., 2025) distinguishes internal reasoning from environment interaction. AgentTTS and related approaches (Wang et al., 2026; Han et al., 2026) further investigate how inference resources can be allocated across agentic trajectories. However, while these approaches demonstrate the benefit of additional computation, they focus on examining performance at predetermined compute levels only.

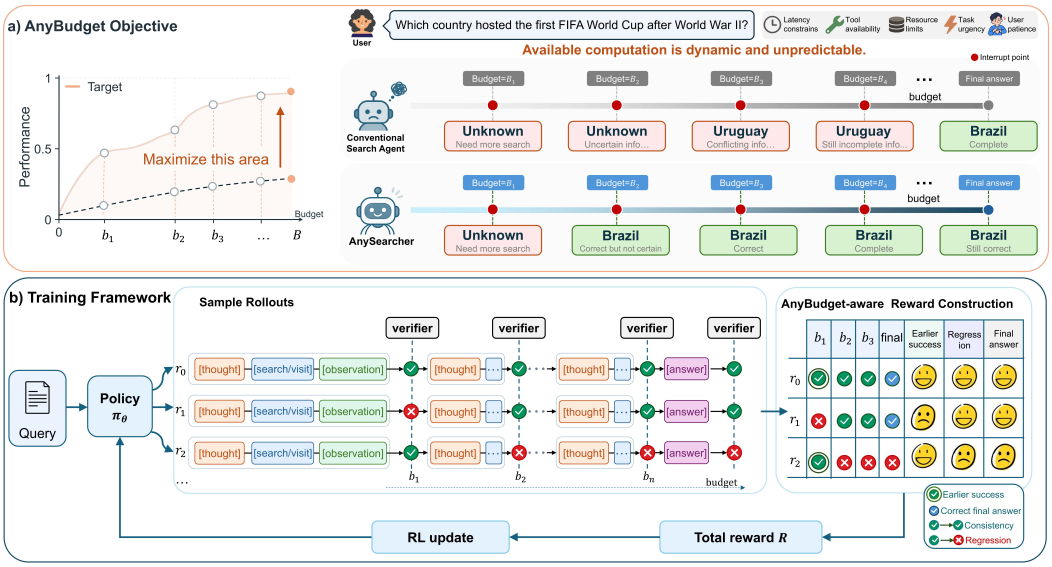


Figure 1: **Overview of AnySearcher. (a) AnyBudget objective.** AnySearcher optimizes expected answer quality over a distribution of interruption budgets, encouraging a single search policy to remain reliable at different possible stopping points along the same trajectory. **(b) Training framework.** For each sampled trajectory, intermediate answers are evaluated at multiple interruption checkpoints. The resulting correctness sequence is aggregated into an AnyBudget-aware reward, which is then used for reinforcement learning updates.

Budget Aware Reasoning. Most prior works on budget aware inference focus on reasoning models, where computation is measured primarily by generated tokens. These methods estimate an appropriate reasoning budget, explicitly condition generation on the available budget, or optimize policies across multiple token limits, enabling reasoning depth to adapt to different computational constraints (Han et al., 2026; Yu et al., 2025; Wen et al., 2025; Qi et al., 2026). More recent works extend budget awareness to search agents, where external tool interactions constitute an additional inference resource. Budget aware agents condition search behavior on the remaining tool call budget or adapt resource allocation during execution (Liu et al., 2025a; Lin et al., 2026). These approaches adapt inference to specified resource constraints, whereas AnySearcher addresses the anytime setting, where the stopping budget is undetermined ahead of time. By optimizing solutions throughout the search trajectory, our approach encourages early success and preserves correctness as additional computation becomes available.

3 METHOD

AnySearcher targets search settings where the available inference budget is not known in advance. Instead of optimizing for a predetermined budget, we train a single policy to remain reliable across possible interruption points while continuing to benefit from additional search. We first introduce the tool-augmented search setting and define inference budget in terms of tool interactions in section 3.1. We then formulate the AnyBudget objective over possible interruption budgets in section 3.2, and derive its exact decomposition into first-success utility and correctness regression. Finally, in section 3.3, we develop an interruption-aware training procedure that turns the AnyBudget objective into a practical reinforcement learning signal for policy optimization.

3.1 PRELIMINARIES

Search Agent. We consider a tool augmented search agent parameterized by a policy π_θ under the ReAct framework (Yao et al., 2022). Given a query q, the agent alternates between internal reasoning and interactions with an external search environment before producing a final answer.

The agent has access to two primitive actions:

- SEARCH(q) retrieves a ranked list of search results $\{(u_j, s_j)\}_{j=1}^{M}$ for query q, where u_j denotes the URL of the jth result and s_j denotes its associated snippet;
- VISIT(u) visits a URL u selected from the retrieved search results and returns its page content.

We count each invocation of either SEARCH or VISIT as one unit of inference budget, so that our inference budget is measured by the total number of tool calls.

3.2 ANYBUDGET SEARCH

AnyBudget Objective. We consider an interactive search setting in which the available tool-call budget is unknown when search begins. Because search may be interrupted at arbitrary tool-call budgets due to resource, latency, or external constraints, we optimize a single policy across possible interruption points rather than for one predetermined budget.

Let π_θ denote the search policy and let τ denote a trajectory sampled from this policy. If the trajectory is interrupted after B tool calls, the policy produces an answer $\hat{a}_B$ from the corresponding prefix $\tau_{\leq B}$. Given an answer verifier $V(q, a) \in \{0, 1\}$, we define the correctness of the interrupted answer at budget B as

$$r_B(\tau) = V(q, \hat{a}_B). \tag{1}$$

We evaluate the policy by averaging this correctness over possible interruption budgets. Given a distribution $p(B)$ over interruption budgets, the AnyBudget objective is

$$J_{\text{Any}}(\theta) = \mathbb{E}_{B\sim p(B),\, \tau\sim\pi_\theta}\left[r_B(\tau)\right]. \tag{2}$$

In practice, we approximate $p(B)$ using K discrete interruption checkpoints $0 < B_1 < \cdots < B_K$. Let $w_k = p(B = B_k)$ denote the probability mass assigned to checkpoint B_k, with $w_k \geq 0$ and $\sum_{k=1}^{K} w_k = 1$. We then define the trajectory-level AnyBudget utility as

$$U_{\text{Any}}(\tau) = \sum_{k=1}^{K} w_k r_k(\tau), \tag{3}$$

where $r_k(\tau)$ denotes correctness when trajectory τ is interrupted at checkpoint B_k. The policy objective can equivalently be written as

$$J_{\text{Any}}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[U_{\text{Any}}(\tau)\right]. \tag{4}$$

For uniform w_k, U_{Any} is the mean correctness across the selected interruption budgets. The objective therefore rewards overall reliability across possible stopping points rather than minimizing trajectory length or forcing the agent to terminate as early as possible. We provide additional background analysis how this AnyBudget objective relates to a standard objective in Appendix A.4.

AnyBudget Utility Decomposition. We further decompose the AnyBudget utility into two trajectory-level behaviors: when correctness is first achieved and whether it is preserved as search continues.

To identify when a correct answer first becomes available, we define the first successful checkpoint as

$$H(\tau) = \min\{k : r_k(\tau) = 1\}, \tag{5}$$

with $H(\tau) = \infty$ if no checkpoint is successful.

Under the ideal anytime property, a correct answer obtained at a smaller budget should remain correct as more budget becomes available. We therefore define the first-success utility as

$$W_H(\tau) = \sum_{k=H(\tau)}^{K} w_k, \tag{6}$$

with $W_H(\tau) = 0$ when no checkpoint is successful. As illustrated in fig. 2(a), an earlier first success covers a larger portion of the interruption distribution and therefore yields greater utility.

Real trajectories, however, may violate this property: continued search can overturn a previously correct answer, as illustrated in fig. 2(b). To identify such correctness regressions, we define

$$e_k(\tau) = \max_{j \le k} r_j(\tau), \qquad g_k(\tau) = e_k(\tau) - r_k(\tau), \quad (7)$$

where $e_k(\tau)$ indicates whether correctness has been achieved by checkpoint k, and $g_k(\tau) = 1$ indicates that a previously achieved correct answer is not preserved at checkpoint k.

We aggregate these regressions across interruption budgets using the weighted regression loss

$$D_{\text{reg}}(\tau) = \sum_{k=1}^{K} w_k g_k(\tau). \quad (8)$$

These two quantities give an exact decomposition of the original AnyBudget utility:

$$U_{\text{Any}}(\tau) = \underbrace{W_H(\tau)}_{\text{first-success utility}} - \underbrace{D_{\text{reg}}(\tau)}_{\text{correctness regression}}. \quad (9)$$

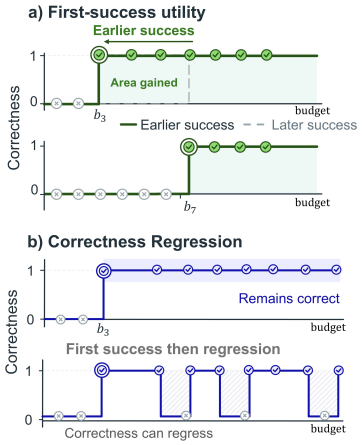


Figure 2: **First Success and Correctness Regression.** **(a)** Earlier first success yields greater accumulated utility. **(b)** Subsequent correctness regression reduces accumulated utility.

This decomposition is algebraically equivalent to the original AnyBudget utility and is used directly in our trajectory-level reward computation.

3.3 Optimizing the AnyBudget Search Agent

The AnyBudget objective evaluates a trajectory at multiple possible interruption budgets, whereas a standard rollout provides supervision only from its terminal answer. We therefore construct interruption-level correctness signals from each rollout and use them to derive a practical trajectory-level reward for policy optimization.

AnyBudget Reward Computation. For each query q, we sample G trajectories $\{\tau_i\}_{i=1}^{G}$ from the current policy. Let L_i denote the number of tool calls made by trajectory τ_i, and let $y_i \in \{0, 1\}$ denote the correctness of its terminal answer.

For each interruption checkpoint $B_k < L_i$, we truncate the trajectory after B_k tool calls and require the policy to produce an immediate answer $\hat{a}_{i,k}$ from the corresponding prefix. The correctness at checkpoint B_k is then defined as

$$r_{i,k} = \begin{cases} V(q, \hat{a}_{i,k}), & B_k < L_i, \\ y_i, & B_k \ge L_i, \end{cases} \quad (10)$$

where the terminal correctness is carried forward once the trajectory has already terminated.

The resulting correctness sequence allows us to compute the first-success utility $W_H(\tau_i)$ and correctness-regression loss $D_{\text{reg}}(\tau_i)$ defined in section 3.2. Their exact decomposition gives the trajectory-level AnyBudget reward

$$R_i^{\text{Any}} = W_H(\tau_i) - D_{\text{reg}}(\tau_i). \quad (11)$$

While the AnyBudget reward directly optimizes performance across interruption budgets, it can become noisy for low-solvability queries, where rare and unstable intermediate successes may dominate the training signal. We therefore adapt the contribution of terminal correctness and AnyBudget supervision according to query solvability.

Capability-Adaptive Training. For each query q, we estimate empirical solvability using the terminal pass rate within its rollout group:

$$p_q = \frac{1}{G} \sum_{i=1}^{G} y_i. \quad (12)$$

Using p_q as the capability weight, we define

$$R_i^{\text{correct}} = (1 - p_q)y_i + p_q R_i^{\text{Any}}. \tag{13}$$

Including the format reward, the final trajectory reward is

$$R_i = R_i^{\text{format}} + R_i^{\text{correct}}. \tag{14}$$

Thus, training relies more on terminal correctness for low-solvability queries, while progressively shifting toward AnyBudget supervision as query solvability improves. This produces the final-trajectory level reward for final policy optmization.

Policy Optimization. Given the trajectory rewards, we compute the group-relative advantage

$$A_i = \frac{R_i - \mu_R}{\sigma_R + \epsilon}, \tag{15}$$

where μ_R and σ_R are the mean and standard deviation of rewards within the rollout group.

Let T_i denote the number of generated policy tokens in trajectory τ_i. We optimize the clipped GRPO objective

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T_i}\sum_{t=1}^{T_i}\min\left(\rho_{i,t}A_i, \text{clip}\left(\rho_{i,t}, 1-\epsilon_c, 1+\epsilon_c\right)A_i\right)\right], \tag{16}$$

where

$$\rho_{i,t} = \frac{\pi_\theta(a_{i,t} \mid s_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t})}. \tag{17}$$

This trains a single policy to remain reliable across varying interruption budgets without conditioning on a predetermined target budget.

4 Experiments

4.1 Experimental Setup

Training Data. We use the LiteResearcher training set (Li et al., 2026), containing 16,199 questions across 25 curriculum buckets covering direct search, Wikipedia QA, multi-hop reasoning, citation-subgraph reasoning, and scientific QA.

Search Environment. Training uses the LiteResearcher web corpus (Li et al., 2026), containing approximately 32M webpages. The corpus is encoded with BGE-M3 and indexed using Milvus HNSW. SEARCH retrieves relevant documents, while VISIT returns query-conditioned webpage summaries generated by the model.

Benchmarks. We evaluate on four deep-search benchmarks: *GAIA-Text* (Mialon et al., 2024), *FRAMES* (Krishna et al., 2025), *BrowseComp-Plus* (Chen et al., 2026), and *WebWalker* (Wu et al., 2025), covering multi-step reasoning, multi-hop evidence aggregation, reasoning-intensive retrieval, and hierarchical web navigation.

Baselines. We compare against baselines representing complementary budget-control strategies. GRPO (Shao et al., 2024) provides a standard outcome-based RL baseline without explicit budget control. LENGTH-PENALTY, adapted from Kimi k1.5 (Kimi Team et al., 2025), directly penalizes excessive tool usage, while MULTI-BUDGET L1-MAX, adapted from L1-Max (Aggarwal & Welleck, 2025), trains a shared policy under multiple predefined budgets. We further include the training-free BUDGET TRACKER and BATS (Liu et al., 2025a), which explicitly expose the remaining budget at inference time. Together, these baselines cover standard RL, training-time efficiency control, budget-conditioned training, and inference-time budget awareness.

Evaluation Metrics. We evaluate both *anytime performance* and *budget efficiency*. For each method, we first construct an accuracy–budget curve, where $\text{Acc}(B)$ is the answer accuracy when the agent is limited to at most B tool calls. *AUC@5* and *AUC@10* summarize anytime accuracy under small budgets by computing the normalized area under the accuracy–budget curve over

$B \in [0, 5]$ and $B \in [0, 10]$, respectively. *AUC* measures the same quantity over the full evaluation range $B \in [0, 50]$. Higher AUC therefore indicates that a method maintains higher accuracy across the corresponding range of possible interruption budgets. *Budget@90% Best* measures how quickly a method reaches near-best performance. It is defined as the minimum tool-call budget at which the method reaches 90% of the highest final accuracy achieved by any compared method on the same benchmark. Lower values indicate greater budget efficiency.

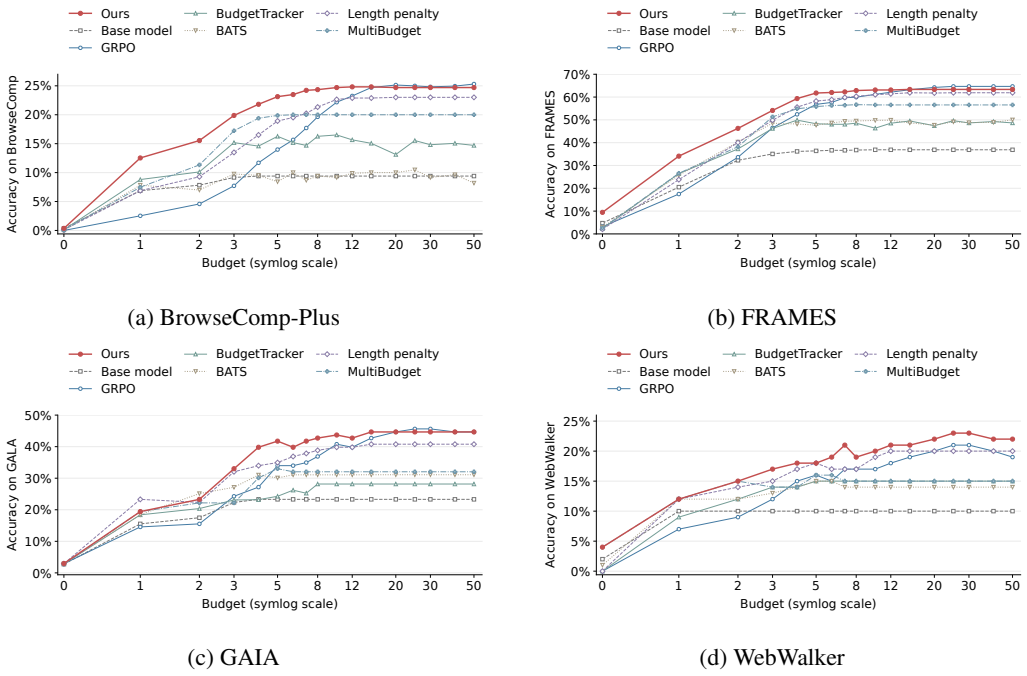
Training Details. All methods are initialized from Qwen3-4B-Instruct (Yang et al., 2025). We use maximum prompt and generation lengths of 40k tokens. Each step samples 128 questions with $G = 8$ rollouts, yielding 1,024 trajectories per update. We use AdamW with a learning rate of 1×10^{-6}, temperature 1.0, top-$p = 1.0$, up to 100 assistant turns, and at most four queries per tool call. The format reward is weighted by 0.1. Training uses two NVIDIA H200 GPUs with 141 GB memory each. Following the staged training strategy used in Kimi k1.5 (Kimi Team et al., 2025), AnySearcher and Length-Penalty are initialized from a Standard GRPO checkpoint at step 200, allowing the policy to first learn effective search before optimizing budget efficiency. AnySearcher and Multi-Budget are trained for 300 steps, whereas GRPO and Length-Penalty are stopped at 250 steps because GRPO has already reached a validation plateau despite continued growth in tool usage, while Length-Penalty begins to collapse beyond this point. Training takes approximately 4–5 days.

4.2 Experimental Results

Table 1: **Summary of Anytime performance across four benchmarks.** AUC@5 and AUC@10 denote the normalized area under the accuracy–budget curve over the first 5 and 10 tool calls, respectively, while AUC is computed over the full evaluation range. Budget@90% Best is the minimum number of tool calls required to reach 90% of the best final accuracy achieved by any compared method on the same benchmark. Higher AUC and lower Budget@90% Best indicate better performance. Best results are shown in bold, second-best results are underlined, and AnySearcher is highlighted with a light-blue background.

	BrowseComp-Plus				FRAMES			
Method	**AUC@5↑**	**AUC@10↑**	**AUC↑**	**Budget@ 90% Best↓**	**AUC@5↑**	**AUC@10↑**	**AUC↑**	**Budget@ 90% Best↓**
Base	7.63	8.51	9.22	> 50	28.92	32.82	36.08	> 50
Tracker	11.38	13.58	14.61	> 50	37.03	42.46	47.54	> 50
BATS	7.66	8.45	9.34	> 50	37.61	43.40	47.85	> 50
GRPO	6.70	12.55	<u>22.35</u>	<u>12</u>	36.00	47.68	<u>60.92</u>	7
Multi Budget	<u>13.09</u>	<u>16.54</u>	19.31	> 50	<u>40.04</u>	48.24	54.89	> 50
Length Penalty	11.13	15.95	21.58	<u>12</u>	39.89	<u>49.88</u>	59.43	<u>5</u>
AnySearcher	**16.30**	**20.20**	**23.82**	**5**	**45.89**	**54.20**	**61.50**	**4**
	GAIA				**WebWalker**			
Method	**AUC@5↑**	**AUC@10↑**	**AUC↑**	**Budget@ 90% Best↓**	**AUC@5↑**	**AUC@10↑**	**AUC↑**	**Budget@ 90% Best↓**
Base	18.34	20.82	22.80	> 50	9.20	9.60	9.92	> 50
Tracker	19.73	23.26	27.18	> 50	11.30	13.15	14.63	> 50
BATS	23.64	27.31	30.34	> 50	11.80	13.05	13.81	> 50
GRPO	20.00	28.20	<u>41.14</u>	<u>10</u>	10.20	13.35	18.58	20
Multi Budget	22.35	27.25	31.08	> 50	13.00	14.15	14.83	> 50
Length Penalty	<u>26.12</u>	<u>32.09</u>	38.97	15	<u>13.40</u>	<u>15.45</u>	<u>19.07</u>	<u>12</u>
AnySearcher	**27.58**	**34.81**	**42.57**	**5**	**14.60**	**17.05**	**21.09**	**7**

Main Results. Table 1 summarizes the overall anytime performance across four benchmarks. AnySearcher achieves the highest AUC@5, AUC@10, and full-range AUC across all benchmarks, indicating consistently stronger performance over different inference-budget ranges. It also reaches 90% of the best final accuracy within only 5, 4, 5, and 7 tool calls on BrowseComp-Plus, FRAMES, GAIA, and WebWalker, respectively. These results show that AnySearcher combines strong anytime



(a) BrowseComp-Plus (b) FRAMES

(c) GAIA (d) WebWalker

Figure 3: **Anytime performance across inference budgets.** Budget–accuracy curves on (a) BrowseComp-Plus, (b) FRAMES, (c) GAIA, and (d) WebWalker. Each curve reports answer accuracy when the search process is interrupted at different tool-call budgets.

performance with improved budget efficiency. To better understand how these gains emerge across different budgets, we next examine the full budget–accuracy curves.

Anytime Performance Across Budgets. Figure 3 shows how accuracy changes with increasing tool-call budgets. Standard GRPO performs relatively weakly under small budgets and improves mainly with additional search. Length-Penalty favors performance under smaller budgets but provides more limited gains as computation increases, while MultiBudget improves performance across several budget levels but tends to plateau earlier. Training-free methods, including BATS and BudgetTracker, provide modest improvements but generally remain below training-based approaches. In contrast, AnySearcher performs strongly under tight budgets while continuing to benefit from additional computation. This behavior is consistent across benchmarks, indicating that performance gains obtained at smaller budgets are preserved rather than traded off against performance at larger budgets. While the budget–accuracy curves demonstrate stronger aggregate performance, they do not reveal whether individual trajectories preserve correctness as search continues.

Correctness Preservation. To directly assess this property, we measure the rate of *negative flips*, i.e., how often a previously correct answer becomes incorrect during continued search due to overthinking or other issues. As shown in Fig. 4, Standard GRPO exhibits substantial correctness regression, with flip rates of 20.5% on BrowseComp-Plus and 15.6% on FRAMES. AnySearcher reduces these rates to 8.8% and 5.9%, respectively. These results show that AnySearcher not only improves aggregate accuracy across budgets, but also better preserves correctness within individual search trajectories as additional computation becomes available. We next examine whether capability-adaptive training is necessary for effectively optimizing this behavior.

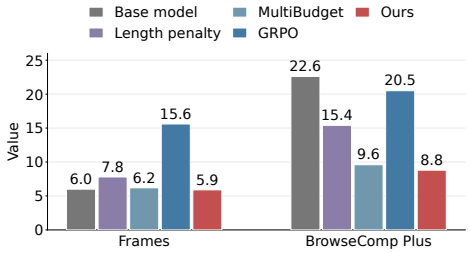


Figure 4: **Correctness preservation during continued search.** Conditional negative flip rates on BrowseComp-Plus and FRAMES. Lower values indicate better correctness preservation.

Effect of Capability-Adaptive Training. We compare AnySearcher with an AnyBudget-only variant that applies R_i^{Any} uniformly to all queries. For difficult queries that the current policy cannot yet solve reliably, the training signal may be dominated by sporadic correct intermediate answers. GRPO's group normalization can amplify these small reward differences (Liu et al., 2025b; Wang, 2026). Since longer trajectories provide more opportunities to obtain such intermediate successes, trajectory-level credit can inadvertently favor continued search over producing a reliable answer at early interruption points. This can degrade small-budget performance, as shown in fig. 5. Capability-adaptive weighting reduces this noisy supervision on difficult queries while retaining the benefits of the AnyBudget objective.

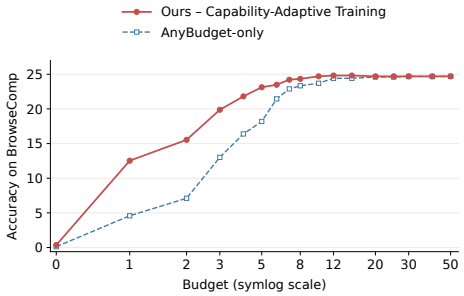
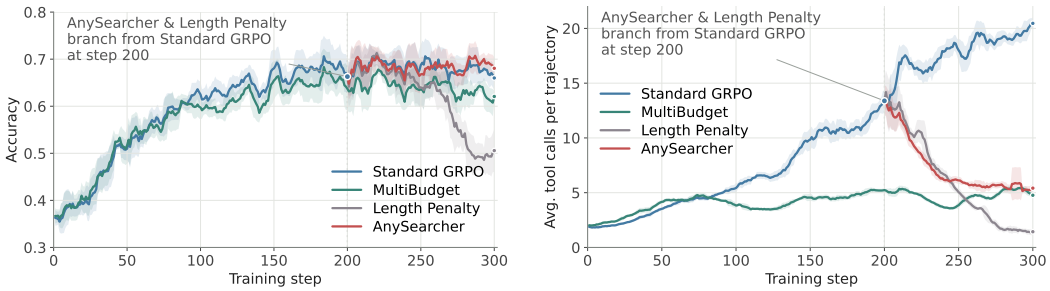


Figure 5: **Effect of Capability-Adaptive Training on BrowseComp-Plus.** Budget–Accuracy Curves comparing AnySearcher with AnyBudget-only variant

Training Behavior. Finally, we examine how the different optimization objectives shape agent behavior over the course of training. Figure 6 shows the evolution of training accuracy and average tool usage for different methods. Standard GRPO continues to increase tool usage even after accuracy begins to plateau, indicating that additional search does not necessarily translate into further performance gains. MultiBudget initially increases tool usage before gradually stabilizing. AnySearcher and Length-Penalty are initialized from Standard GRPO at step 200, after the policy has acquired a basic search capability. Length-Penalty subsequently reduces tool usage but eventually exhibits a joint collapse in accuracy and search length. In contrast, AnySearcher gradually reduces unnecessary tool interactions while preserving accuracy, leading to a more stable balance between search effectiveness and budget efficiency.



(a) Training accuracy

(b) Average tool calls

Figure 6: **Evolution of search behavior during training.** **(a)** Training accuracy across different methods. **(b)** Average number of tool calls per trajectory. AnySearcher and Length-Penalty are initialized from Standard GRPO at step 200.

5 Conclusion

Real-world agents operate in dynamic environments, where the moment at which they must act or stop computation is often unknown in advance. An ideal agent should therefore exhibit an *anytime* property: it should remain useful throughout computation, maintaining a reliable solution whenever it is interrupted while continuing to improve when additional computation is available. We introduced *AnySearcher* to instantiate this property for LLM-based search agents. AnySearcher optimizes an AnyBudget objective over possible interruption points and uses interruption-aware, capability-adaptive training to turn this objective into a practical reinforcement learning procedure. Across multiple search benchmarks, AnySearcher improves accuracy across inference budgets, reaches strong performance with fewer tool interactions, and better preserves correctness during continued search. More broadly, we view anytime behavior as an important capability for agents operating under dynamic and unpredictable execution conditions, where reliable performance may be required at any point during computation.

AI USE STATEMENT

Generative AI tools were used only to polish the writing. They were not used for research ideation, experimental design, analysis, or figure generation.

ETHICS STATEMENT

This work uses publicly available models and standard benchmarks. It does not involve human subjects and does not release new datasets.

REPRODUCIBILITY STATEMENT

Source code is included as supplementary material. Hyperparameter settings are described in the paper.

REFERENCES

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Research: Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2(3):7, 2025.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, et al. Browsecomp-plus: a fair and disentangled evaluation benchmark for deep search agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22349–22370, 2026.

Thomas L Dean, Mark S Boddy, et al. An analysis of time-dependent planning. In *AAAI*, volume 88, pp. 49–54, 1988.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Rujun Han, Samira Daruki, Tomas Pfister, Chen-Yu Lee, Jiefeng Chen, Ke Jiang, Yanfei Chen, I-Hung Hsu, Jin Miao, Ethan Liang, Tengxiao Liu, Zifeng Wang, and Jun Yan. Cost-effective agent test-time scaling. 2026.

Eric J Horvitz. Reasoning about beliefs and actions under computational resource constraints. *arXiv preprint arXiv:1304.2759*, 2013.

Metod Jazbec, James Allingham, Dan Zhang, and Eric Nalisnick. Towards anytime classification in early-exit architectures by enforcing conditional monotonicity. *Advances in Neural Information Processing Systems*, 36:56138–56168, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Jiin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. The cost of dynamic reasoning: Demystifying ai agents and test-time scaling from an ai infrastructure perspective. In *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1–16. IEEE, 2026.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. 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. 4745–4759, 2025.

Wanli Li, Bince Qu, Bo Pan, Jianyu Zhang, Zheng Liu, Pan Zhang, Wei Chen, and Bo Zhang. Literesearcher: A scalable agentic rl training framework for deep research agent. *arXiv preprint arXiv:2604.17931*, 2026.

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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025.

Yuxiang Lin, Zihan Wang, Mengyang Liu, Yuxuan Shan, Longju Bai, Junyao Zhang, Xing Jin, Boshan Chen, Jinyan Su, Xingyao Wang, et al. Bagen: Are llm agents budget-aware? *arXiv preprint arXiv:2606.00198*, 2026.

Jiayu Liu, Cheng Qian, Zhaochen Su, Qing Zong, Shijue Huang, Bingxiang He, and Yi R Fung. Costbench: Evaluating multi-turn cost-optimal planning and adaptation in dynamic environments for llm tool-use agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12826–12858, 2026.

Tengxiao Liu, Zifeng Wang, Jin Miao, I Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, et al. Budget-aware tool-use enables effective agent scaling. *arXiv preprint arXiv:2511.17006*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Yichuan Ma, Linyang Li, Yongkang Chen, Peiji Li, Xiaozhe Li, Qipeng Guo, Dahua Lin, and Kai Chen. Timely machine: Awareness of time makes test-time scaling agentic. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4619–4636, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Itamar Mishani, Hayden Feddock, and Maxim Likhachev. Constant-time motion planning with anytime refinement for manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10337–10343. IEEE, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Litu Ou, Kuan Li, Huifeng Yin, Liwen Zhang, Zhongwang Zhang, Xixi Wu, Rui Ye, Zile Qiao, Yong Jiang, Pengjun Xie, et al. Browseconf: Confidence-guided test-time scaling for web agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 446–465, 2026.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Optimizing anytime reasoning via budget relative policy optimization. *Advances in Neural Information Processing Systems*, 38:23429–23451, 2026.

Peijia Qin, Qi Cao, and Pengtao Xie. Atlas: Agentic test-time learning-to-allocate scaling. *arXiv preprint arXiv:2606.01667*, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, et al. Thinking vs. doing: Agents that reason by scaling test-time interaction. *arXiv preprint arXiv:2506.07976*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Fali Wang, Hui Liu, Zhenwei Dai, Jingying Zeng, Zhiwei Zhang, Zongyu Wu, Chen Luo, Zhen Li, Xianfeng Tang, Qi He, et al. Agenttts: Large language model agent for test-time compute-optimal scaling strategy in complex tasks. *Advances in Neural Information Processing Systems*, 38:98396–98433, 2026.

Yu Wang. The dark room in the reward channel: Dense prediction rewards collapse grpo-trained llm agents–and what actually works. *arXiv preprint arXiv:2607.21273*, 2026.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Budgetthinker: Empowering budget-aware llm reasoning with control tokens. *arXiv preprint arXiv:2508.17196*, 2025.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10290–10305, 2025.

Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-congruent training: Towards regression-free model updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14299–14308, 2021.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zishun Yu, Tengyu Xu, Di Jin, Karthik Abinav Sankararaman, Yun He, Wenxuan Zhou, Zhouhao Zeng, Eryk Helenowski, Chen Zhu, Sinong Wang, et al. Think smarter not harder: Adaptive reasoning with inference aware optimization. *arXiv preprint arXiv:2501.17974*, 2025.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, 2025.

Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. When more thinking hurts: Overthinking in llm test-time compute scaling. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23967–23977, 2026.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, et al. Scaling test-time compute for llm agents. *arXiv preprint arXiv:2506.12928*, 2025.

Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–73, 1996.

A Appendix

A.1 Datasets

We provide additional details on the four benchmarks used to evaluate AnySearcher: BrowseComp-Plus, FRAMES, GAIA-Text, and WebWalker. These benchmarks cover complementary information-seeking settings, including deep web search, multi-source evidence aggregation, general agentic reasoning, and hierarchical web navigation. For benchmarks with a dedicated retrieval corpus, we use the corresponding fixed corpus; otherwise, search is performed over the open web using the Serper API.

BrowseComp-Plus. BrowseComp-Plus evaluates deep-search agents on challenging questions that require iterative retrieval and evidence synthesis. Unlike benchmarks based on live web search, it provides a fixed corpus of approximately 100K curated web documents containing verified supporting evidence and challenging negatives. We use this corpus as the retrieval environment, enabling controlled evaluation under different tool-call budgets.

FRAMES. FRAMES contains 824 challenging multi-hop questions that require combining information from multiple Wikipedia articles. Each question is associated with relevant source articles, covering reasoning types such as numerical, temporal, tabular, and multi-constraint reasoning. We construct the retrieval environment from the corresponding Wikipedia corpus, allowing the agent to iteratively retrieve evidence under a fixed search space.

GAIA-Text. GAIA-Text is the text-only subset of GAIA and evaluates general-purpose agentic reasoning on questions that may require multiple rounds of information retrieval and reasoning. Since the benchmark does not provide a dedicated closed retrieval corpus for our setting, we use the Serper API as the search backend, allowing the agent to retrieve information from the open web.

WebWalker. WebWalker evaluates information seeking that requires navigating and reasoning over hierarchically organized web content. It complements the other benchmarks by emphasizing multi-step exploration and web traversal. We use the Serper API as the search backend to provide open-web retrieval during evaluation.

Retrieval Backends. BrowseComp-Plus and FRAMES are evaluated with their corresponding fixed retrieval corpora, while GAIA-Text and WebWalker use the Serper API for open-web search. This setup preserves the controlled retrieval environments when provided by the benchmark while supporting realistic web search for benchmarks without a dedicated corpus.

A.2 BASELINES

We provide additional implementation details for the training-time and training-free baselines. All methods are evaluated in the same search environment with the same underlying search and visit tools. We use a shared tool-call accounting protocol to ensure that performance differences are evaluated under comparable interaction budgets.

GRPO. We use Group Relative Policy Optimization (GRPO) as the standard reinforcement learning baseline. The policy is optimized using final-answer correctness as the reward, without explicitly encouraging early success or controlling tool usage. This baseline allows us to examine the effect of introducing budget-aware optimization beyond standard outcome-based reinforcement learning.

Length-Penalty. We implement a tool-call penalty baseline following the training strategy of Kimi (Kimi Team et al., 2025). The policy is first trained using outcome rewards without additional efficiency constraints. After this warmup stage, a penalty proportional to tool usage is introduced to encourage shorter search trajectories. This staged training procedure allows the policy to first acquire task-solving capability before being optimized for search efficiency. Unlike AnySearcher, which directly optimizes the quality of intermediate answers across interaction budgets, this baseline encourages efficiency by penalizing the overall cost of a trajectory.

Multi-Budget L1-Max. We adapt L1-Max(Aggarwal & Welleck, 2025) to the tool-augmented search setting by replacing token-level generation limits with tool-call budgets. Rather than training separate policies for different budgets, we jointly train a single shared policy using trajectories collected under multiple budget constraints. The resulting policy learns to complete search tasks within the specified interaction budget. This baseline evaluates whether training a shared policy across multiple budgets is sufficient to achieve strong anytime performance, without explicitly optimizing intermediate answer quality or correctness preservation.

Budget Tracker. Budget Tracker is a training-free baseline that explicitly provides the agent with its current tool usage and remaining interaction budget. The budget information is updated throughout the search process, enabling the agent to condition its decisions on the available resources without modifying the underlying policy.

BATS. We additionally consider BATS (Liu et al., 2025a), which incorporates budget-aware planning and self-verification into the inference procedure. Compared with Budget Tracker, BATS introduces additional mechanisms for managing the search process according to the remaining budget. This baseline allows us to compare learned budget-aware behavior against inference-time budget management strategies.

A.3 EXPERIMENTS

A.3.1 TRAINING DETAILS

Length Penalty. Following the group-relative length reward in Kimi k1.5, we introduce a length-based reward to encourage efficient tool use. For each group of G trajectories, let $L_{\min}$ and $L_{\max}$ denote the minimum and maximum numbers of tool calls, respectively. The normalized length coefficient for trajectory i is defined as

$$\lambda_i = \frac{1}{2} - \frac{L_i - L_{\min}}{L_{\max} - L_{\min}}, \tag{18}$$

which assigns $+0.5$ to the shortest trajectory and -0.5 to the longest. To avoid rewarding short but incorrect trajectories, the length reward is defined as

$$r_{\text{len},i} = \begin{cases} \lambda_i, & y_i = 1, \\ \min(0, \lambda_i), & y_i = 0. \end{cases} \tag{19}$$

The final reward is $R_i = R_{\text{base},i} + \beta r_{\text{len},i}$, where we set $\beta = 0.5$, bounding the length contribution to $[-0.25, 0.25]$, one quarter of the unit correctness reward gap. When $G = 1$ or all trajectories have identical lengths, the length reward is set to zero. Following the optimization setting of Kimi

k1.5, we compute advantages by subtracting the group mean without dividing by the group standard deviation. This avoids amplifying the length reward in groups with identical correctness outcomes, where standard deviation normalization could otherwise cause the length term to dominate the optimization regardless of the value of β.

Multi-Budget. Following Length-Controlled Policy Optimization (L1-Max), we train a budget-conditioned policy by specifying a tool-call budget B in the prompt. For each question, we sample $B \sim \mathrm{Uniform}\{0, \ldots, 50\}$, with all $G = 8$ rollouts of the same question sharing the same budget to ensure meaningful group-relative optimization. The reward is defined as

$$R_i = R_{\mathrm{base},i} - \alpha \max(0, L_i - B), \tag{20}$$

where L_i is the number of tool calls and $\alpha = 0.02$ controls the penalty for exceeding the budget. We adopt the L1-Max penalty because the specified budget represents an upper bound rather than a target length. Consequently, trajectories that finish within the budget incur no penalty, while those exceeding it are penalized in proportion to the number of excess tool calls. This formulation encourages the policy to satisfy varying budget constraints without incentivizing unnecessary tool use to exhaust the allocated budget.

A.4 Relation Between Standard and AnyBudget Objectives

We further examine the relation between conventional terminal-answer optimization and the AnyBudget objective.

Let $y(\tau) \in \{0, 1\}$ denote the correctness of the terminal answer produced by trajectory τ at its termination budget $L(\tau)$. The standard reasoning objective evaluates only this terminal outcome:

$$J_{\mathrm{Std}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[y(\tau)\right]. \tag{21}$$

In contrast, the AnyBudget objective evaluates correctness across possible interruption checkpoints:

$$J_{\mathrm{Any}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{k=1}^{K} w_k r_k(\tau)\right], \tag{22}$$

where $w_k \geq 0$ and $\sum_{k=1}^{K} w_k = 1$.

For checkpoints whose budgets are no smaller than the natural termination budget $L(\tau)$, the terminal answer is carried forward. Define

$$\mathcal{T}(\tau) = \{k : B_k \geq L(\tau)\}. \tag{23}$$

For every $k \in \mathcal{T}(\tau)$,

$$r_k(\tau) = y(\tau). \tag{24}$$

Since all checkpoint rewards and weights are nonnegative, the trajectory-level AnyBudget utility satisfies

$$\begin{aligned} U_{\mathrm{Any}}(\tau) &= \sum_{k=1}^{K} w_k r_k(\tau) \\ &\geq \sum_{k \in \mathcal{T}(\tau)} w_k r_k(\tau) \\ &= \left(\sum_{k \in \mathcal{T}(\tau)} w_k\right) y(\tau). \end{aligned} \tag{25}$$

Let

$$\alpha(\tau) = \sum_{k \in \mathcal{T}(\tau)} w_k \tag{26}$$

denote the total interruption probability assigned to checkpoints at or beyond the natural termination of trajectory τ. Then

$$U_{\mathrm{Any}}(\tau) \geq \alpha(\tau) y(\tau). \tag{27}$$

Taking expectation over trajectories gives

$$J_{\mathrm{Any}}(\theta) \geq \mathbb{E}_{\tau\sim\pi_\theta}\left[\alpha(\tau)y(\tau)\right]. \tag{28}$$

Assuming that every trajectory terminates no later than the largest checkpoint B_K, we have $K \in \mathcal{T}(\tau)$ for every trajectory. Therefore,

$$\alpha(\tau) \geq w_K. \tag{29}$$

It follows that

$$\boxed{J_{\mathrm{Any}}(\theta) \geq w_K J_{\mathrm{Std}}(\theta)}. \tag{30}$$

Equivalently, for $w_K > 0$,

$$\boxed{J_{\mathrm{Std}}(\theta) \leq \frac{1}{w_K} J_{\mathrm{Any}}(\theta)}. \tag{31}$$

This relation does not require monotonic correctness. It follows from the fact that the terminal answer is retained at interruption checkpoints beyond the natural termination of the trajectory.

Monotonic Correctness. We next consider the special case in which correctness, once achieved, is preserved through all subsequent checkpoints and through the terminal answer. Under this condition,

$$r_k(\tau) \leq y(\tau), \qquad \forall k \in \{1, \ldots, K\}. \tag{32}$$

Therefore,

$$\begin{aligned} U_{\mathrm{Any}}(\tau) &= \sum_{k=1}^{K} w_k r_k(\tau) \\ &\leq \sum_{k=1}^{K} w_k y(\tau) \\ &= y(\tau), \end{aligned} \tag{33}$$

where the last equality follows from $\sum_{k=1}^{K} w_k = 1$.

Taking expectation over trajectories yields

$$J_{\mathrm{Any}}(\theta) \leq J_{\mathrm{Std}}(\theta). \tag{34}$$

Combining Eqs. equation 30 and equation 34, under monotonic correctness we obtain

$$\boxed{w_K J_{\mathrm{Std}}(\theta) \leq J_{\mathrm{Any}}(\theta) \leq J_{\mathrm{Std}}(\theta)}. \tag{35}$$

Equivalently,

$$\boxed{J_{\mathrm{Any}}(\theta) \leq J_{\mathrm{Std}}(\theta) \leq \frac{1}{w_K} J_{\mathrm{Any}}(\theta)}. \tag{36}$$

The upper bound relies on monotonic correctness and does not generally hold when correctness regression occurs. In particular, a trajectory may produce a correct answer at an intermediate checkpoint and later terminate with an incorrect answer. Such behavior is captured by the correctness-regression decomposition introduced in the main text.