香港城市大學
City University of Hong Kong

Inference-Time Budget Control for LLM Search Agents

Zhengru Fang[1,*], Senkang Hu[1,*], Zhonghao Chang[2], Yu Guo[1], Yihang Tao[1], Hongyao Liu[1], Mengzhe Ruan[3], Jun Huang[1], Yuguang Fang[1]

[1] City University of Hong Kong, [2] Tsinghua University, [3] Alibaba Ant Group, * Equal contribution

LLM search agents increasingly rely on tools at inference time, but their trajectories are often constrained by hard limits on both tool calls and generated tokens. Under such dual budgets, better answers require not only stronger models, but also explicit control over which search action should receive the next budget unit and when the accumulated evidence is sufficient to commit a final answer. We study this problem in multi-hop question answering (QA) and formulate it as two-stage inference-time budget control. At search time, our controller assigns each feasible action a task-level Value-of-Information (VOI) score, defined as an operational estimate of marginal task value per unit budget under the current search state and remaining dual budget, and uses this score to choose among retrieval, decomposition, and answer commitment. After search, a selective evidence-grounded finalizer compares the trajectory answer with a refined candidate and rewrites only when the residual error appears to be a low-risk answer-form error. Across four multi-hop QA benchmarks, three LLM backbones, and four budget levels, the method yields positive aggregate gains over four audited baselines under the same hard dual-budget protocol. Ablations show that search-time budget control, especially budget-dependent penalty, provides the main performance gain, while answer-time control helps mainly when the retrieval path is already adequate. These results suggest that inference-time budget control for LLM search agents should govern both how budget is spent during search and how the final answer is committed.

1. Introduction

Large language models (LLMs) are increasingly deployed as search agents that use external tools during inference. ReAct [69], Toolformer [43], BATS [28], and Search-o1 [21] treat retrieval as part of the inference trajectory rather than a fixed preprocessing step. Broader agent systems add reflection, tree search, collaboration, and software tools [46, 68, 7, 53, 78, 77, 30]. In this paradigm, answer quality depends not only on model capability, but also on how the agent spends tool-call and token budgets while inference is still unfolding.

This paradigm also changes test-time scaling from a token-only problem into an inference-time control problem. Prior work studies how additional inference compute improves outputs through repeated sampling, self-refinement, and adaptive allocation [56, 32, 47, 12, 25]. For search agents, however, compute is not token-only: extra budget can create more branches, intermediate states, and low-yield actions. Systems such as Search-R1 [17], BATS [28], and recent browsing agents [80, 57, 45, 15] already operate in this tool-heavy regime. Deployment studies likewise show that unconstrained tool use leads to budget-inefficient failures, including redundant retrieval, wasted branching, and brittle context growth [6, 72, 19, 31]. Under explicit budgets, more inference-time compute and better answers are no longer equivalent.

This raises the central inference-time control problem of *how to decide, under explicit tool-call and output-token*

budgets, what to do next, and how to commit a final answer once search stops. Existing budget-aware approaches either route across models to reduce cost [6, 72] or inject budget tracking into the agent loop [28, 74], but none of them explicitly control both action allocation during search and answer resolution after search under the same hard dual-budget audit, which motivates this work. Three difficulties make this control problem nontrivial.

Challenge 1: Budget must be allocated across heterogeneous actions. At each step a search agent can retrieve evidence, decompose the question, or answer directly. These actions differ in cost and return, and the right choice depends on the evidence state and remaining budget. Infrastructure analyses show that naive scaling leads to over-search and diminishing returns [19, 3], while token-level budgeting cannot resolve this action-level trade-off under coupled tool-call and output-token budgets [12, 25].

Challenge 2: Final-answer errors persist after successful search. Even with the right evidence, the final answer can fail on exactness: yes/no polarity, binary choice, typed slots, or alias completion. Self-refinement [32], stepwise verification [5, 48], and reasoning-aware selection [52] can help, but they target open-ended generation rather than budgeted search where every extra call has a cost.

Challenge 3: Rewriting introduces intervention risk. Unconditional answer rewriting can erase bridge structure or reverse comparative semantics in multi-hop settings [51, 50]. Reflexion-style feedback loops [46] mitigate some errors but do not explicitly model the risk of damaging a correct answer. Thus, an answer controller must intervene only when the expected gain outweighs risk.

To address these challenges, we propose a training-free, two-stage inference-time controller built on a tree-search backbone inspired by BAVT [23]. At search time, the controller scores each feasible action by task-level VOI: an operational estimate of marginal task value per unit budget under the current trajectory state and remaining dual budget, rather than Shannon information gain or Bayesian posterior value. The score combines critic-derived value change, structural signals, cost normalization, budget penalty, and conservative guards to choose among retrieval, decomposition, and answer commitment. After search, an evidence-grounded finalizer rewrites only when expected exactness gain outweighs the risk of damaging an otherwise adequate trajectory answer. The method therefore controls two distinct decision points: how budget is spent during search, and how the final answer is committed after search. We evaluate VOI under a shared hard dual-budget audit across four multi-hop QA benchmarks, three LLM backbones, and a four-level budget ladder. The results support explicit two-stage control while also exposing its boundary conditions across datasets, budgets, and backbones. Our main contributions are as follows:

❶ **Two-stage inference-time budget-control formulation.** We formulate budget-aware inference in LLM search agents as a two-stage inference-time control problem: search-time action allocation under coupled tool-call and output-token budgets, followed by answer-time finalization under intervention risk.

❷ **Task-level VOI control for search actions.** We introduce a training-free scorer that ranks retrieval, decomposition, and answer commitment by estimated marginal task value per unit budget. The scorer combines critic-derived progress, structural signals, cost normalization, budget-dependent penalty, and conservative guards.

❸ **Risk-controlled answer finalization.** We introduce an evidence-grounded finalizer that rewrites only for low-risk answer-form errors, such as yes/no polarity, binary choices, typed-slot mismatches, or supported factoid completion, and abstains when bridge or comparative reasoning remains unresolved.

❹ **Empirical analysis under hard dual-budget audit.** We evaluate across four benchmarks, three backbones, and four budget levels under the same hard dual-budget protocol. The results show positive aggregate gains and many low- and mid-budget improvements, while ablations identify budget-dependent penalty as the dominant component and expose backbone- and dataset-dependent boundary

conditions where the gains diminish.

2. Related Work

Search agents and tool-augmented inference. Tool use shifted LLMs from static text generation to interactive decision-making. ReAct [69] interleaves reasoning and acting, Toolformer [43] studies self-supervised tool use, and later systems add reflection, tree search, collaboration, environment grounding, and software tools [46, 68, 7, 29, 53, 78, 77, 39, 63, 66, 30]. Agent evaluation has also expanded toward real-world and proactive-assistance settings [49]. Search-oriented agents such as Search-R1 [17], BATS [28], Search-o1 [21], BrowseComp [57], and recent browsing systems [62, 45, 79, 20, 11, 54, 10, 59, 36] push toward open-ended information seeking. We instead study QA agents under explicit tool-call and output-token budgets.

Test-time scaling and budget-aware agent scaling. Test-time scaling work turns extra inference compute into better outputs through repeated sampling, self-refinement, early stopping, meta-generation, adaptive compute, efficient task adaptation, and token-budgeted reasoning [56, 32, 22, 58, 47, 52, 12, 25, 14, 34, 2, 5, 33, 4, 38, 73]. Budget-aware inference adds a harder question: not just how much extra compute to spend, but where. FrugalGPT [6] and EcoAssistant [72] route across models, while agent-scaling work [80, 28, 19, 31] argues that tool use changes the scaling problem itself. Related edge and mobile intelligence studies further highlight that LLM and AI deployment is constrained by communication, caching, model downloading, continual adaptation, and split or federated execution costs [40, 41, 61, 60, 8]. We follow this resource-aware framing at a finer granularity: step-level action control during search and selective intervention at finalization.

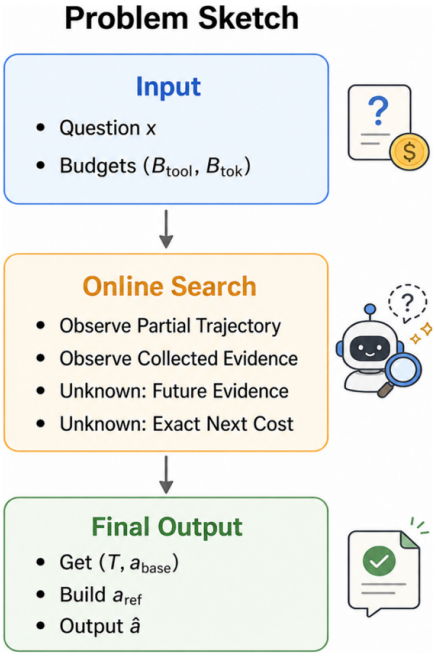

Figure 1: **Problem sketch.** The agent chooses the next search action from the observed trajectory, evidence, and remaining budget; future evidence and exact next cost are unknown.

Workflow search and tree-based control. DSPy [18], PromptAgent [55], Promptbreeder [9], ADAS [16], AgentSquare [44], AFlow [71], AutoFlow [24], EvoFlow [70], MermaidFlow [75], and HyEvo [64] optimize prompts, module graphs, or workflows before deployment. Related LLM-based automatic algorithm-design work also uses language models to evolve heuristics or heuristic sets before inference [26, 27]. A separate line treats inference as structured search over states, thoughts, or plans, as in Tree of Thoughts [68], Graph of Thoughts [1], Language Agent Tree Search [77], BAVT [23], and CATS [74]. Our method belongs to this second family: given an existing search backbone, it controls which action spends the next budget unit and how the final answer is resolved.

Answer verification and selective intervention. Self-Refine [32] shows that post-hoc editing can improve outputs, while stepwise debugging [76], VerifAI [48], Reasoning-Aware Self-Consistency [52], and SETS [5] show that revision quality depends on trigger timing and intervention risk. Our final-answer layer applies that lesson to budgeted search QA: the finalizer keeps the trajectory answer unless the expected gain clearly outweighs the intervention risk, since aggressive answer replacement can erase bridge structure or comparative semantics even when the trajectory is adequate.

3. Problem Formulation

We study inference-time budget control for an LLM search agent answering a question x under dual budgets $B = (B_{\text{tool}}, B_{\text{tok}})$, where B_{tool} limits tool calls and B_{tok} limits budgeted output tokens. The agent faces two coupled decisions. As illustrated in Figure 1, the agent must choose the next search action from the currently observed trajectory, collected evidence, candidate answer state, and remaining budget, before future evidence and the exact next token cost are known. The agent faces two coupled decisions. During search, it must decide how to allocate the remaining budget across retrieval, decomposition, and answer commitment. After search terminates, it must decide whether to keep the trajectory answer or replace it with a refined answer derived from the collected evidence. The problem is therefore to control both budget allocation during search and answer commitment after search.

Let μ denote the search-time decision rule. Given x and B, the search process produces a trajectory and base answer $(\mathcal{T}, a_{\text{base}}) = \mathcal{M}(x, B; \mu)$. The decision rule is applied online. Before choosing the next operation, the agent observes the question, the partial trajectory, the evidence and decompositions collected so far, any candidate answer states, and the remaining tool-call and token budgets. It does not know future evidence, final answer quality, or the exact output-token cost of the next operation. Therefore, action selection uses an estimated local budget charge available before execution, while the realized tool-call and output-token costs are debited after the operation is executed.

The completed trajectory $\mathcal{T}$ records the executed operations, collected evidence, intermediate states, candidate answers, and realized costs. After the trajectory is complete, a refinement function g_{ref} constructs a candidate refined answer $a_{\text{ref}} = g_{\text{ref}}(\mathcal{T}, a_{\text{base}})$, and a second policy σ chooses the final prediction $\hat{a}$ from $\{a_{\text{base}}, a_{\text{ref}}\}$. This stage is intentionally narrow: it is not unrestricted rewriting, but selective intervention between preserving the trajectory answer and replacing it with a refined one. This distinction matters in multi-hop QA, where an unnecessary rewrite can remove bridge entities, alter comparison structure, or replace a precise answer with a broader but less faithful one. We use $R(a, y) \in [0, 1]$ as an abstract bounded QA reward, where a is a predicted answer and y is the gold answer. In experiments, answer quality is reported using exact match and token-level F1; the formulation only requires a bounded scalar reward. The expectation is over the evaluation distribution of (x, y) and the search trajectory induced by the policies. The cost functions $C_{\text{tool}}(\mathcal{T})$ and $C_{\text{tok}}(\mathcal{T})$ denote realized trajectory costs. We formulate the overall problem as

$$\max_{\mu,\sigma} \ \mathbb{E}[R(\hat{a}, y)] \quad \text{s.t.} \quad C_{\text{tool}}(\mathcal{T}) \le B_{\text{tool}},\ C_{\text{tok}}(\mathcal{T}) \le B_{\text{tok}},\ \mathbb{E}[L_{\text{harm}}(\hat{a}, a_{\text{base}}, y)] \le \rho_{\text{harm}}, \tag{1}$$

where ρ_{harm} is the allowable expected harm from answer replacement.

The harm term measures the damage caused by replacing the base answer with a worse final answer:

$$L_{\text{harm}}(\hat{a}, a_{\text{base}}, y) = \max\{0, R(a_{\text{base}}, y) - R(\hat{a}, y)\}.$$

This quantity is zero when the final decision preserves or improves answer quality and positive only when finalization makes the answer worse than the original trajectory answer. Under this formulation, the first stage is an online budget-allocation problem during search, and the second stage is a risk-controlled answer-finalization problem after search. Our method instantiates these two decisions using a search-time controller and an answer-time selector with abstention.

4. Method

Our method instantiates the two decisions in Section 3 with two lightweight layers on top of a generic search backbone. Figure 2 summarizes the full pipeline. In Stage 1, the backbone maintains candidate trajectories and evidence, while the VOI controller scores feasible actions under the remaining dual budget

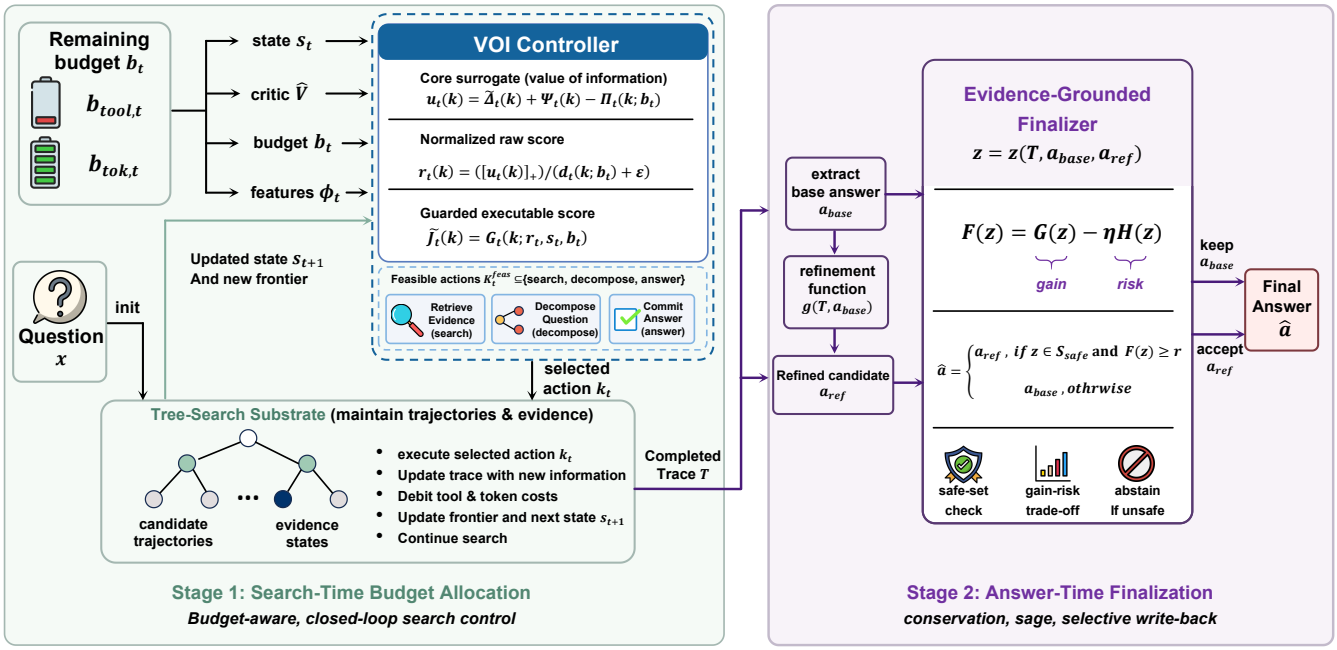

Figure 2: **Two-stage budget control with task-level VOI.** Stage 1 uses a controller based on the task-level VOI score to choose whether the next step should retrieve, decompose, or answer under the remaining dual budget. Stage 2 finalizes the answer conservatively, rewriting only when the case is safe and the expected gain outweighs rewrite risk.

and selects the next operation. The selected operation is executed by the tree-search substrate, which updates the trace and debits realized tool-call and token costs. In Stage 2, the completed trace yields a base answer and a refined candidate; the finalizer keeps the base answer unless a safe, evidence-supported refinement passes the gain–risk check. The search-time layer chooses among evidence acquisition, question decomposition, and answer commitment under the remaining budget. The answer-time layer then chooses between a base answer a_{base} extracted from the trace and a refined answer a_{ref} constructed from the same trace. The final output is $\hat{a}$.

4.1. Search-Time Budget Allocation

Task-level VOI control. At each search step, the controller ranks feasible actions using a task-level VOI score: an operational estimate of marginal task value per unit budget under the current search state and remaining dual budget. This score is used only for action selection; it is not Shannon information gain, Bayesian posterior value, or a learned value model.

We instantiate the online decision rule in Section 3 as follows. At step t, the observed search status is represented by s_t, the remaining budget by $b_t = (b_{\text{tool},t}, b_{\text{tok},t})$, and the feasible action set by $\mathcal{K}_t^{\text{feas}} \subseteq \mathcal{K}$, where $\mathcal{K} = \{\textsc{Search}, \textsc{Decompose}, \textsc{Answer}\}$. For each feasible action k, the controller assigns a pre-execution charge vector $g_t(k) = (g_{\text{tool},t}(k), g_{\text{tok},t}(k))^\top$ and extracts fixed features from s_t summarizing unresolved evidence, compositional structure, answer readiness, stagnation, loop pressure, and premature-answer risk.

The controller scores each feasible action by first forming a budget-shaped utility, then normalizing it as value per budget, and finally applying guards. First, it forms

$$u_t(k) = \underbrace{\widehat{\Delta}_t(k)}_{\text{progress signal}} + \underbrace{\Psi_t(k)}_{\text{structural signals}} - \underbrace{\Pi_t(k; b_t)}_{\text{budget-dependent penalty}}, \tag{2}$$

where $\widehat{\Delta}_t(k)$ is a critic-derived progress signal, $\Psi_t(k)$ aggregates structural signals such as bridge structure, loop pressure, readiness, and early-answer risk, and $\Pi_t(k; b_t)$ is a signed budget-dependent penalty term: it is positive for SEARCH and DECOMPOSE under tight budgets. $\Pi_t(k; b_t)$ serves as a tractable proxy for the oracle budget shadow-cost $\lambda_t^{\star\top} g_t(k)$, with approximation error ε_t^{Π} formalized in Assumption C.5. Second, this task-level action utility is clipped and normalized into the task-level VOI score used for action selection:

$$r_t(k) = \frac{[u_t(k)]_+}{d_t(k;\, b_t) + \epsilon}, \tag{3}$$

where $[\cdot]_+ = \max\{0, \cdot\}$, $\epsilon > 0$ is a small constant, and $d_t(k; b_t) > 0$ is an action-specific budget scale derived from the pre-execution charge and current budget state. We call $r_t(k)$ the task-level VOI score: an operational estimate of marginal task value per unit budget under the current state and remaining dual budget.

Definition 4.1 (Task-level VOI score)**.** For feasible $k \in \mathcal{K}_t^{\text{feas}}$, the task-level VOI score is $r_t(k) = [u_t(k)]_+ / (d_t(k; b_t) + \epsilon)$, where $u_t(k)$ is the task-level action utility and $d_t(k; b_t)$ is the budget-aware action scale. This score estimates marginal task value per unit budget for action selection under the current search state and remaining dual budget.

In the released implementation, $u_t(k)$ is instantiated by fixed coefficients over explicit trajectory features, $d_t(k; b_t)$ by action-specific budget-aware cost terms, and the executable score $\widetilde{\mathcal{J}}_t(k)$ by these normalized scores after conservative guard adjustments.

Third, action-specific guards produce the executable score:

$$\widetilde{\mathcal{J}}_t(k) = \mathfrak{G}_t\big(k;\, r_t,\, s_t,\, b_t\big), \tag{4}$$

where $\mathfrak{G}_t$ is a deterministic guard operator that masks or adjusts raw scores, including premature-answer suppression, factoid-decomposition suppression, and minimum-search enforcement on compositional cases. The controller selects $k_t = \arg\max_{k \in \mathcal{K}_t^{\text{feas}}} \widetilde{\mathcal{J}}_t(k)$.

This design is adaptive, training-free, and interpretable: scores are recomputed at every step, no extra value model is learned, and each term corresponds to an explicit search or budget signal. Appendix A provides empirical controller diagnostics.

4.2. Answer-Time Finalization

After search terminates, the system extracts a base answer a_{base} from the best trajectory and constructs a refined candidate a_{ref} from the same trace $\mathcal{T}$. Finalization is narrow but important: the search path may be largely correct while the final answer still contains local answer-form errors, such as wrong yes/no polarity, wrong binary choice, incomplete alias, or typed-slot mismatch. Unnecessary rewriting remains risky because fluent refinements can erase bridge entities, distort comparisons, or replace precise answers with broader but less faithful ones.

We therefore formulate finalization as selective intervention over the two-candidate set $\{a_{\text{base}}, a_{\text{ref}}\}$. From the completed trace and the candidate pair, we construct a consistency feature vector $z = z(\mathcal{T}, a_{\text{base}}, a_{\text{ref}})$. This vector records evidence relevant to safe revision: support type, slot type, contradiction indicators, and unresolved bridge or comparative reasoning. For exposition, finalization is written as a gain–risk threshold over this feature representation; in implementation, it is a deterministic selector over explicit feature conditions, not a learned scorer or extra LLM call:

$$F(z) = G(z) - \eta H(z), \qquad \hat{a} = \begin{cases} a_{\text{ref}}, & \text{if } z \in \mathcal{S}_{\text{safe}} \text{ and } F(z) \geq \tau, \\ a_{\text{base}}, & \text{otherwise,} \end{cases} \tag{5}$$

where $G(z)$ measures the potential gain, $H(z)$ measures the intervention risk, $\eta > 0$ controls risk aversion, τ is an abstention threshold, and $\mathcal{S}_{\text{safe}}$ is the set of reliable revision cases. In the released implementation, the abstract map g is instantiated by a fixed construction $a_{\text{ref}} = g_{\text{ref}}(\mathcal{T}, a_{\text{base}})$, and the selector is a deterministic rule over explicit feature conditions. Appendix C characterizes the exact rule in Proposition C.15.

The safe set $\mathcal{S}_{\text{safe}}$ excludes unresolved bridge structure, unresolved comparative semantics, and missing direct support, where rewriting is most likely to harm a correct trajectory answer. Thus the finalizer is a conservative answer selector, not a generic editor: it intervenes only when the remaining error appears local to answer-form correction rather than retrieval or path selection. It adds no tool calls and issues no additional LLM call during finalization.

4.3. End-to-End Inference

At inference time, the search procedure maintains a frontier of trajectories while the controller repeatedly selects the next feasible operation using Eqs. (2)–(4). It debits realized tool-call and output-token costs after each operation and stops when the frontier or budget is exhausted, or when the search procedure's termination condition fires. The system then extracts a_{base}, constructs a_{ref}, and applies Eq. (5). Full pseudocode is in Appendix B.

Theoretical support. Appendix C provides the formal support for the two controller layers. For search-time allocation, Appendix C.2 shows that the task-level utility in Eq. (2) locally approximates an oracle budget-charged one-step lookahead value, yielding ranking consistency under a margin condition and a one-step value-gap bound when guards are compatible with the utility ranking. For answer-time finalization, Appendix C.4 derives the gain–risk threshold in Eq. (5) from the harm-constrained objective in Eq. (1). The analysis supports budget-aware action selection and conservative answer replacement, but does not claim global optimality of the full search tree.

5. Experiments

5.1. Setup

We instantiate the LLM search agent with three backbones: Qwen3-32B [65], Qwen3.5-122B [42], and GPT-5.4-Mini [35], chosen to span different model scales, families, and deployment regimes rather than tuning to a single LLM. We evaluate on four multi-hop QA benchmarks: HotpotQA [67], 2WikiMultihopQA [13], MuSiQue [50], and Bamboogle [37]. All methods use the same retrieval configuration: Search-R1 retrieval, question-only queries, and `top_k=5`. We evaluate under four dual-budget levels, `low`, `lower-mid`, `upper-mid`, and `high`, corresponding to $(1, 100)$, $(2, 200)$, $(2, 300)$, and $(3, 500)$, where each pair denotes the tool-call cap and output-token cap.

The audit is strict at the example level: all five methods are scored under the same hard tool/output-token budget constraint, and any example exceeding either constraint is counted as failed. We compare VOI with BAVT [23], BATS [28], AFlow [71], and Search-o1 [21]. We report EM, token-level F1, average tool calls, and average budget output tokens; accounting details, cross-backbone confidence intervals, and feasible-only usage diagnostics are provided in Appendix H.

5.2. Main Results

Figure 3 shows the main budget-scaling pattern on two representative backbones. On Qwen3-32B, whose full EM/F1 table is reported in Appendix E, VOI is best-F1 in 7 of 16 dataset-budget cells and tied best

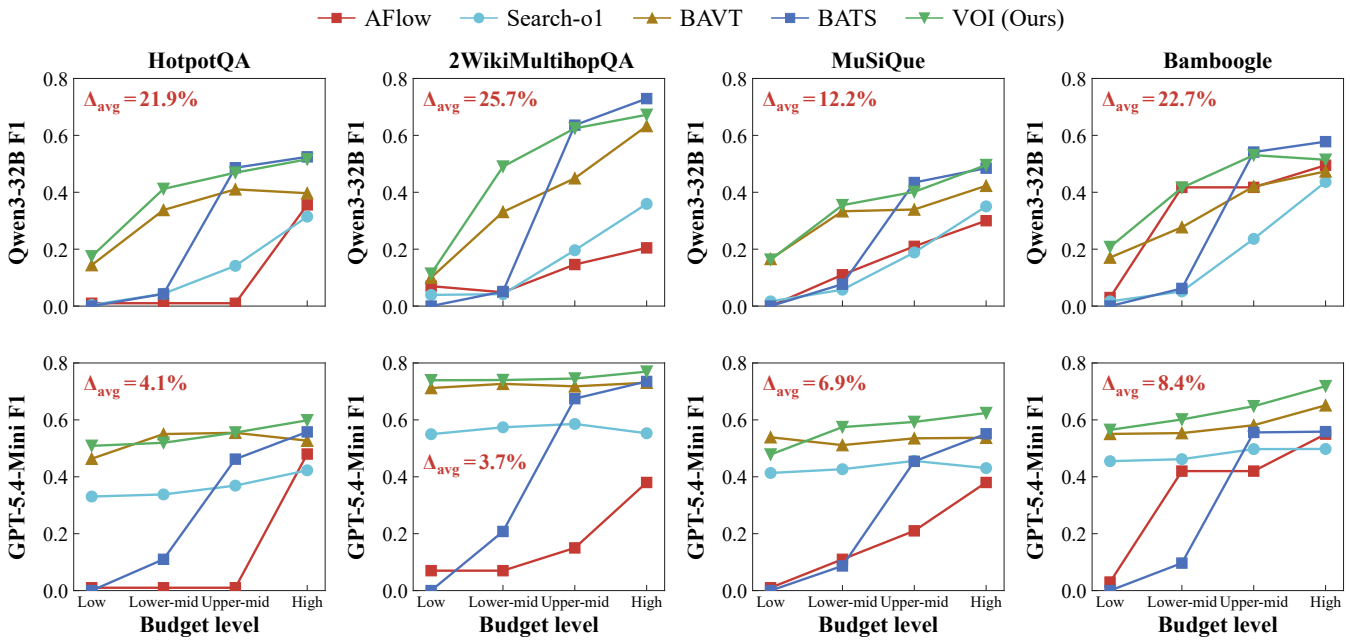

Figure 3: **Cross-model budget scaling curves across four datasets.** Rows report Qwen3-32B and GPT-5.4-Mini; columns report the four multi-hop QA benchmarks. All methods use the shared hard dual-budget audit. Qwen3.5-122B results are reported in Appendix F as a backbone-sensitivity analysis.

in 2 additional cells. It improves over AFlow, BATS, Search-o1, and BAVT in 14/16, 10/16, 16/16, and 15/16 cells, respectively. The strongest gains appear in low and lower-mid budgets, where choosing the next action carefully is most important because wasted retrieval or premature answering quickly exhausts the trajectory. The pattern is positive but not a dominance claim. BATS remains strong in several upper-budget regimes, especially when broader beam-style evidence accumulation is useful, and AFlow leads on 2WikiMultihopQA `lower-mid`. These exceptions are important because they show that budget control interacts with dataset structure and budget level. The main claim is therefore not that VOI wins every cell, but that explicit action-level budget control improves most audited settings under the same hard budget constraints.

On GPT-5.4-Mini, VOI remains consistently competitive despite the stronger base model. This is useful evidence that the controller is not merely compensating for a weak backbone. The gains are smaller in some cells than on Qwen3-32B, which is expected: stronger backbones can already resolve more answer-form and evidence-selection errors without as much controller help. Still, the curves show that the two-stage controller continues to provide value when the model family changes.

We report Qwen3.5-122B separately in Appendix F. That backbone exhibits a more mixed regime, with several cells led by BATS or BAVT at higher budgets. We treat it as a backbone-sensitivity case rather than the main trend. For completeness, Appendix H reports cross-backbone macro deltas over all three backbones, four benchmarks, and four budgets; the aggregate effect remains favorable, while the detailed curves confirm substantial backbone dependence.

5.3. Final-Answer Control Improves Exactness

Final-answer control is most useful when residual errors are answer-form errors rather than search-depth failures, clearest on Bamboogle. Evidence-grounded finalization improves the measured Bamboogle budget ladder without changing tool usage: F1 rises from 0.4382 to 0.4628 at `low`, from 0.5786 to 0.6047 at `upper-mid`, and from 0.5056 to 0.5576 at `high`. An additional `(1,200)` probe shows the same trend,

Table 1: Stage-1 component ablation on Qwen3-32B (`upper-mid` budget). Each cell reports `EM/F1` under the hard dual-budget audit. The 1st, 2nd, and 3rd best results are highlighted in **first**, second, and third colors. Cell background colors are ranked by the average of EM and F1 scores. Best EM and F1 are independently bolded. ↑ indicates higher is better.

Benchmark	BAVT	w/o penalty	w/o norm.	w/o struct.	w/o guards	VOI (Full)
	EM/F1 ↑	EM/F1 ↑	EM/F1 ↑	EM/F1 ↑	EM/F1 ↑	EM/F1 ↑
HotpotQA	0.34/0.41	0.35/0.40	0.34/0.38	0.32/0.37	0.34/0.39	**0.39/0.47**
2WikiMultihopQA	0.40/0.45	0.38/0.43	0.44/0.50	0.39/0.45	0.42/0.48	**0.54/0.63**
MuSiQue	0.26/0.34	0.24/0.30	0.25/0.33	0.30/0.39	0.29/0.38	**0.36/0.43**
Bamboogle	0.33/0.42	0.31/0.40	0.35/0.43	0.28/0.37	0.28/0.36	**0.46/0.56**

improving from 0.4631 to 0.4911 on Bamboogle and from 0.4907 to 0.5174 on 2WikiMultihopQA. Thus, answer-time control yields visible gains when the trajectory already contains adequate evidence but the final answer still has an answer-form error.

Successful Stage 2 interventions are typed-slot corrections, binary-choice repairs, yes/no polarity repairs, and supported factoid completions. This matches the design target: the finalizer is a conservative exactness layer, not a generic rewrite module. It creates no new evidence and launches no additional search or additional LLM call during finalization. Its gains therefore concentrate on answer-form errors, while path-level failures remain outside its scope. This also explains the weaker contribution on MuSiQue, where residual errors more often involve unresolved bridge structure rather than a local answer span.

5.4. Search-Time Controller Component Ablation

Section 4.1 defines the Stage 1 pipeline: task-level action utility, value-per-cost normalization, and guard-adjusted executable scoring in Eqs. (2)–(4). We ablate these components under the `upper-mid` budget on Qwen3-32B by removing budget-dependent penalty $\Pi_t(k; b_t)$, replacing $r_t(k)$ with $[u_t(k)]_+$, removing $\Psi_t(k)$, or bypassing $\mathfrak{G}_t$. All variants use the same search procedure, Search-R1 retrieval, question-only queries, `top_k=5`, and hard dual-budget audit.

Table 1 reports the benchmark-level results. The full controller achieves the best F1 on all four benchmarks, while every component removal reduces the macro average. The largest drop comes from removing budget-dependent penalty, showing that the remaining-budget term is not a cosmetic penalty but a central part of the action-allocation rule. Removing normalization, structural signals, or guards also hurts, although the affected benchmark differs. This pattern supports the method design: the improvement is tied to the three-stage task-level VOI score controller rather than to a generic prompt constraint, search-procedure change, or answer-time finalization effect.

Inference-time cost. Table 2 reports an end-to-end wall-clock timing probe under the Qwen3-32B `upper-mid` setting with 100 examples per dataset. The full VOI controller reduces mean inference time relative to BAVT on all four benchmarks, from 20.91s to 15.23s on average, a 27.2% reduction. Thus the added deterministic controller does not introduce a visible latency burden in this setting; its action choices often shorten trajectories by avoiding low-value operations. The 2WikiMultihopQA row also shows a boundary case where some ablated variants are faster than the full controller, indicating dataset-dependent runtime behavior.

Table 2: End-to-end inference time under Qwen3-32B and the `upper-mid` budget. Each cell reports mean wall-clock seconds per example; parenthesized values give relative change against BAVT. Lower is better. The 1st, 2nd, and 3rd best results, ranked solely by mean inference time, are highlighted in **first**, second, and third colors.

Benchmark	BAVT	w/o penalty	w/o norm.	w/o struct.	w/o guards	VOI (Full)
	Mean Time (s) ↓	Mean Time (s) ↓	Mean Time (s) ↓	Mean Time (s) ↓	Mean Time (s) ↓	Mean Time (s) ↓
Bamboogle	19.92	16.25 (-18.4%)	16.38 (-17.8%)	17.15 (-13.9%)	16.34 (-18.0%)	**14.41** (-27.7%)
HotpotQA	22.01	17.44 (-20.7%)	17.79 (-19.1%)	17.57 (-20.1%)	17.45 (-20.7%)	**14.41** (-34.5%)
MuSiQue	20.23	16.65 (-17.7%)	16.98 (-16.1%)	16.47 (-18.6%)	16.22 (-19.8%)	**14.79** (-26.9%)
2WikiMultihopQA	21.49	15.81 (-26.4%)	**15.25** (-29.0%)	15.53 (-27.7%)	15.64 (-27.2%)	17.30 (-19.5%)
Average	20.91	16.54 (-20.9%)	16.60 (-20.6%)	16.68 (-20.2%)	16.41 (-21.5%)	**15.23** (-27.2%)

5.5. Two-Stage Component Ablation

Figure 4 isolates the two stages under the same audited setting. Stage 1 alone improves F1 on all four benchmarks, confirming that most of the gain comes from search-time budget allocation. The full method yields relative F1 gains of +5.7%, +11.8%, +14.7%, and +18.4% over BAVT on HotpotQA, 2WikiMultihopQA, MuSiQue, and Bamboogle, respectively. Stage 2 contributes no additional gain on HotpotQA, but accounts for 13.4%, 41.9%, and 27.8% of the total gain on 2WikiMultihopQA, MuSiQue, and Bamboogle. This supports the intended division of labor: Stage 1 provides broad budget-aware search control, while Stage 2 acts as a sparse exactness layer when the retrieved trajectory is already mostly adequate. The finalizer is sparse, conservative, and exactness-oriented: it avoids broad rewriting and provides targeted corrections when the retrieved trajectory is adequate but the answer still has an answer-form error.

6. Discussion and Limitations

Our results show that budget-aware search is not a monotone scaling problem. Larger tool-call and output-token budgets can improve evidence coverage, but may also introduce redundant search or noisier finalization. This is visible on 2WikiMultihopQA and Bamboogle, where middle budgets sometimes outperform `high`. The task-level VOI controller mitigates this trade-off by making budget spending state-dependent, but does not remove the tension between exploration and commitment.

The main boundary appears in high-budget regimes and across backbones. BATS remains competitive in several upper-budget cells, and Qwen3.5-122B shows a more mixed regime in Appendix F because stronger backbones narrow the relative gap, consistent with the diminishing-returns regime expected for any inference-time controller layered on a more capable base model. The finalizer is also exactness-oriented rather than reasoning-complete: it can repair local answer-form errors, but cannot recover from wrong retrieval paths or unresolved bridge relations. Extending the controller to stronger retrieval backends, richer query rewriting, and longer-horizon browsing remains future work.

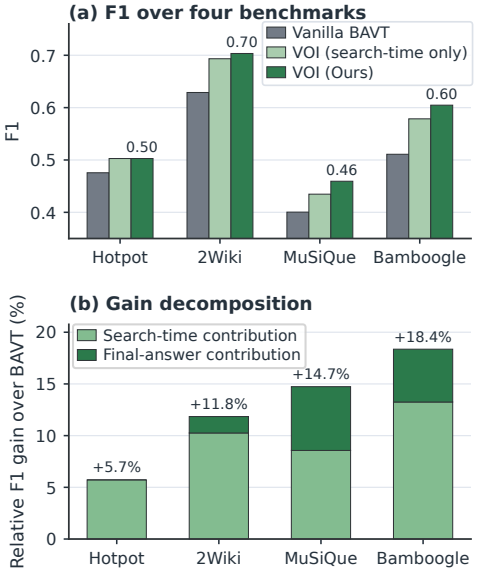

Figure 4: Two-stage component ablation. Top: absolute F1 for BAVT, search-time-only VOI, and full VOI. Bottom: relative F1 gain over BAVT, split into search-time and final-answer contributions; labels report total gain and the Stage 2 share.

7. Conclusion

This paper studies tool-augmented LLM search under explicit tool-call and output-token budgets. We formulate the problem as two-stage constrained inference: a search-time decision over how to spend the next unit of budget, followed by an answer-time decision over whether a refined answer is worth the intervention risk. Our method instantiates this formulation with a training-free controller based on the task-level VOI score over SEARCH, DECOMPOSE, and ANSWER, together with a conservative evidence-grounded finalizer that rewrites only under low-risk exactness conditions. Across four multi-hop QA benchmarks, four budget levels, and three LLM backbones, the results show that budget control is useful but not uniform. The search-time controller provides the main gains by improving action allocation under strict budgets, while the finalizer adds a sparse exactness benefit when the trajectory already contains adequate evidence. The remaining failures clarify the boundary of the approach: more budget is not always better, backbone behavior matters, and answer-time control cannot repair incorrect retrieval paths.

Broader impacts. Budget-aware search can improve the deployability of tool-augmented agents by reducing unnecessary tool use, making inference cost more predictable, and exposing when an agent chooses to search, decompose, or answer. These properties are useful for applications where latency, token usage, or external tool calls must be controlled. At the same time, more efficient search agents could also be used in harmful information-seeking workflows. We therefore emphasize hard budget audits, explicit accounting, conservative finalization, and failure analysis, so that budgeted agent behavior is easier to inspect rather than easier to hide.

References

[1] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 17682–17690, 2024.

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

[3] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

[4] 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, 2025. doi: 10.48550/ARXIV.2504.00762.

[5] Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, and Sercan Oz. Arik. SETS: leveraging self-verification and self-correction for improved test-time scaling. *CoRR*, abs/2501.19306, 2025. doi: 10.48550/ARXIV.2501.19306.

[6] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

[7] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2023.

[8] Zihao Ding, Beining Wu, Jun Huang, and Shiwen Mao. Application-aware twin-in-the-loop planning for federated split learning over wireless edge networks. *arXiv preprint arXiv:2604.26105*, 2026.

[9] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktaschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *ICML*. OpenReview.net, 2024.

[10] 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.

[11] Rujun Han, Yanfei Chen, Zoey CuiZhu, Lesly Miculicich, Guan Sun, Yuanjun Bi, Weiming Wen, Hui Wan, Chunfeng Wen, Solene Maitre, 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, 2025. doi: 10.48550/ARXIV.2507.16075.

[12] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855, 2025.

[13] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[14] Senkang Hu, Xudong Han, Jinqi Jiang, Yihang Tao, Zihan Fang, Yong Dai, Sam Tak Wu Kwong, and Yuguang Fang. Distribution-aligned decoding for efficient llm task adaptation. *arXiv preprint arXiv:2509.15888*, 2025.

[15] Senkang Hu, Yong Dai, Yuzhi Zhao, Yihang Tao, Yu Guo, Zhengru Fang, Sam Tak Wu Kwong, and Yuguang Fang. Optimizing agentic reasoning with retrieval via synthetic semantic information gain reward. *arXiv preprint arXiv:2602.00845*, 2026.

[16] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems, 2024.

[17] 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.

[18] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[19] 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. *arXiv preprint arXiv:2506.04301*, 2025.

[20] 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 super-human reasoning for web agent. *CoRR*, abs/2507.02592, 2025. doi: 10.48550/ARXIV.2507.02592.

[21] 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, 2025. doi: 10.48550/ARXIV.2501.05366.

[22] 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.

[23] Yushu Li, Wenlong Deng, Jiajin Li, and Xiaoxiao Li. Spend less, reason better: Budget-aware value tree search for llm agents, 2026.

[24] Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. Autoflow: Automated workflow generation for large language model agents. *CoRR*, abs/2407.12821, 2024.

[25] Zheng Li, Qingxiu Dong, Jingyuan Ma, Di Zhang, Kai Jia, and Zhifang Sui. Selfbudgeter: Adaptive token allocation for efficient llm reasoning. *arXiv preprint arXiv:2505.11274*, 2025.

[26] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

[27] Fei Liu, Yilu Liu, Qingfu Zhang, Tong Xialiang, and Mingxuan Yuan. Eoh-s: Evolution of heuristic set using llms for automated heuristic design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 37090–37098, 2026.

[28] Tengxiao Liu, Zifeng Wang, Jin Miao, I-Hung Hsu, Jun Yan, Jiefeng Chen, Rujun Han, Fangyuan Xu, Yanfei Chen, Ke Jiang, Samira Daruki, Yi Liang, William Yang Wang, Tomas Pfister, and Chen-Yu Lee. Budget-aware tool-use enables effective agent scaling, 2025.

[29] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*, 2024.

[30] Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.

[31] Ruofan Lu, Yichen Li, and Yintong Huo. Exploring autonomous agents: A closer look at why they fail when completing tasks. *arXiv preprint arXiv:2508.13143*, 2025.

[32] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

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

[34] 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.

[35] OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, March 2026. Accessed: 2026-05-05.

[36] 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.

[37] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.

[38] 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.

[39] Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. Autoact: Automatic agent learning from scratch via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[40] Guanqiao Qu, Qiyuan Chen, Wei Wei, Zheng Lin, Xianhao Chen, and Kaibin Huang. Mobile edge intelligence for large language models: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 27(6):3820–3860, 2025.

[41] Guanqiao Qu, Zheng Lin, Qian Chen, Jian Li, Fangming Liu, Xianhao Chen, and Kaibin Huang. Trimcaching: Parameter-sharing edge caching for ai model downloading. *IEEE Transactions on Networking*, 2026.

[42] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[43] 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.

[44] Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space, 2024.

[45] 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, 2025.

[46] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[47] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[48] Nan Tang, Chenyu Yang, Ju Fan, Lei Cao, Yuyu Luo, and Alon Y. Halevy. Verifai: Verified generative AI. In *CIDR*. www.cidrdb.org, 2024.

[49] Yuanbo Tang, Huaze Tang, Tingyu Cao, Lam Nguyen, Anping Zhang, Xinwen Cao, Chunkang Liu, Wenbo Ding, and Yang Li. Proagentbench: Evaluating llm agents for proactive assistance with real-world data. *arXiv preprint arXiv:2602.04482*, 2026.

[50] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475.

[51] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557.

[52] Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. Reasoning aware self-consistency: Leveraging reasoning paths for efficient LLM sampling. In *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*, pages 3613–3635. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.184.

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

[54] 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, 2025. doi: 10.48550/ARXIV.2508.02694.

[55] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P. Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *ICLR*. OpenReview.net, 2024.

[56] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[57] 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. *arXiv preprint arXiv:2504.12516*, 2025.

[58] Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *Trans. Mach. Learn. Res.*, 2024, 2024.

[59] 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.

[60] Beining Wu and Jun Huang. Lifecycle-aware federated continual learning in mobile autonomous systems. *arXiv preprint arXiv:2604.20745*, 2026.

[61] Beining Wu, Zihao Ding, and Jun Huang. A review of continual learning in edge ai. *IEEE Transactions on Network Science and Engineering*, 2026.

[62] Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webdancer: Towards autonomous information seeking agency. *CoRR*, abs/2505.22648, 2025. doi: 10.48550/ARXIV.2505.22648.

[63] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

[64] Beibei Xu, Yutong Ye, Chuyun Shen, Yingbo Zhou, Cheng Chen, and Mingsong Chen. Hyevo: Self-evolving hybrid agentic workflows for efficient reasoning, 2026.

[65] 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.

[66] John Yang, Carlos E 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.

[67] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[68] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

[69] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[70] Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. Evoflow: Evolving diverse agentic workflows on the fly, 2025.

[71] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2024.

[72] Jieyu Zhang, Ranjay Krishna, Ahmed H Awadallah, and Chi Wang. Ecoassistant: Using llm assistant more affordably and accurately. *arXiv preprint arXiv:2310.03046*, 2023.

[73] 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, 2025. doi: 10.48550/ARXIV.2503.24235.

[74] Zihao Zhang, Hui Wei, Kenan Jiang, Shijia Pan, Shu Kai, and Fei Liu. Cost-awareness in tree-search llm planning: A systematic study, 2025.

[75] Chengqi Zheng, Jianda Chen, Yueming Lyu, Wen Zheng Terence Ng, Haopeng Zhang, Yew-Soon Ong, Ivor Tsang, and Haiyan Yin. Mermaidflow: Redefining agentic workflow generation via safety-constrained evolutionary programming, 2025.

[76] Li Zhong, Zilong Wang, and Jingbo Shang. Debug like a human: A large language model debugger via verifying runtime execution step by step. In *ACL (Findings)*, pages 851–870. Association for Computational Linguistics, 2024.

[77] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

[78] 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. *arXiv preprint arXiv:2307.13854*, 2023.

[79] 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, 2025. doi: 10.48550/ARXIV.2506.15741.

[80] 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.

A. Controller Behavior Analysis

This appendix complements Section 4 with descriptive evidence on the realized behavior of the search-time controller. These results are not additional algorithmic components; they visualize how the implemented controller scores feasible actions and activates guards across budget and compositionality states.

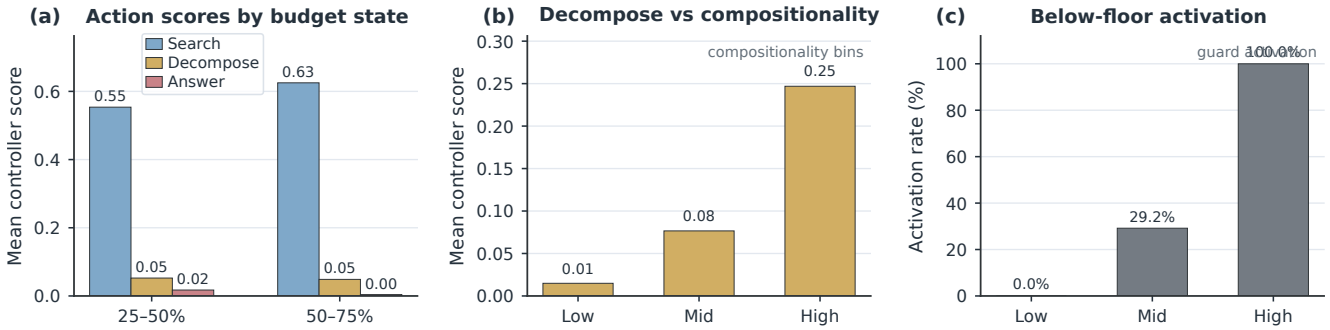

Figure 5: **Empirical behavior of the search-time controller.** **(a)** Mean controller scores for SEARCH, DECOMPOSE, and ANSWER across realized remaining-budget bands. **(b)** The mean DECOMPOSE score increases with question compositionality, consistent with the intended role of decomposition in resolving bridge structure. **(c)** The below-floor guard activates mainly for high-compositionality states, reflecting the conservative minimum-search behavior used to avoid premature answer commitment. All quantities are descriptive statistics computed from realized audited trajectories.

Table 3: **Summary of the search-time controller.**

Action	Main positive signals	Main suppressing signals
SEARCH	unresolved evidence, missing new support, low closure, loop pressure	high budget pressure, repeated search saturation
DECOMPOSE	high compositionality, unresolved bridge structure, stagnation after search	factoid-like question, repeated decomposition saturation
ANSWER	high closure, strong answer support, candidate answer present, tighter budget	early-answer risk, unresolved bridge structure, weak support

As summarized in Table 3, the search-time controller is a fixed rule-based scorer over three actions: SEARCH, DECOMPOSE, and ANSWER. At each decision step, it evaluates four aspects of the current search state. First, it estimates whether important evidence remains unresolved and whether another retrieval step is still likely to help. Second, it checks whether the question appears structurally compositional, for example through unresolved bridge structure, in which case decomposition becomes more valuable than ordinary search. Third, it evaluates answer readiness through closure, answer support, and the presence of a candidate answer span, while penalizing premature commitment when support remains weak. Fourth, it applies budget pressure and action-specific costs so that expensive actions become less attractive as the remaining tool and token budget shrinks. In addition to these smooth score terms, the controller uses a small number of hard guards: it enforces a minimum amount of search on compositional cases, suppresses unnecessary decomposition on factoid-like questions, and blocks overly early answer commitment when structural risk remains high.

B. Inference Algorithm and Prompt Modules

Algorithm 1 Two-Stage Budget-Aware Inference

Input: question x, tool budget B_{tool}, output-token budget B_{tok}
initialize the search procedure, frontier $\mathcal{F}$, trace $\mathcal{T}$, and remaining budget $b_0 = (B_{\text{tool}}, B_{\text{tok}})$
while $\mathcal{F} \neq \emptyset$ and b_t is not exhausted and termination has not fired **do**
  select the current frontier state s_t according to the search procedure
  construct the feasible operation set $\mathcal{K}_t^{\text{feas}}$
  for each feasible action $k \in \mathcal{K}_t^{\text{feas}}$ **do**
    compute action-specific costs and controller features from the current search state
    score k through the three-stage pipeline: task-level action utility $u_t(k)$ (Eq. (2)), normalized raw score $r_t(k)$ (Eq. (3)), guarded executable score $\widetilde{\mathcal{J}}_t(k)$ (Eq. (4))
  end for
  select the highest-scoring feasible action k_t and execute it
  update the trace $\mathcal{T}$, the frontier $\mathcal{F}$, and the remaining budget b_{t+1}
end while
extract the base answer a_{base} from the best available trajectory
construct the refined candidate a_{ref} from the completed trace $\mathcal{T}$
build $z = z(\mathcal{T}, a_{\text{base}}, a_{\text{ref}})$ and apply the finalization rule (Eq. (5))
if $z \in \mathcal{S}_{\text{safe}}$ and $F(z) \geq \tau$ **then**

  return a_{ref}
else

  return a_{base}
end if

B.1. System Prompt

Our method reuses the same single-backbone search procedure as BAVT for planning, generation, and value estimation. The central difference is not that we replace the generator with a new workflow, but that we intervene *before* generation with an explicit search-time controller. Concretely, the controller scores feasible actions through the three-stage pipeline in Eqs. (2)–(4) and selects

$$k_t^\star = \arg\max_{k \in \mathcal{K}_t^{\text{feas}}} \widetilde{\mathcal{J}}_t(k),$$

and only then materializes the selected action as a deterministic prompt insertion. The generator therefore still follows a standard system-prompt plus user-turn format, but its behavior is constrained by a controller-selected instruction whose choice is determined by the task-level VOI score outside the LLM. This is the key interface between the theory and the prompt design: the theory decides *which* action should spend the next unit of budget, and the prompt enforces *how* that chosen action must be executed.

System Prompt: Budget-Aware Generator with VOI-Guided Control

```
You are a precise AI assistant for budget-constrained multi-hop question answering.

Rules:
- Think step by step and explicitly show your reasoning.
- In each turn, do exactly one action: either call one tool or provide the final answer.
- You may use tools multiple times across turns if needed.
- Follow the current dynamic instruction exactly; it defines what kind of action is allowed in this turn.
- If the instruction requires retrieval, you must produce exactly one <tool_call>.
- If the instruction requires answering, you must not call tools.
- Keep the final answer to a single short grounded span, not a full sentence.
- If an entity may be ambiguous, disambiguate the query with a type or role already supported by the
    question or evidence.

<budget>
Tool Budget Used: {tool_budget_used}/{tool_budget_total}
```

```
Tool Budget Remaining: {b_tool}
Output Token Budget Used: {token_budget_used}/{token_budget_total}
Output Token Budget Remaining: {b_token}
</budget>

{plan_context}

{dynamic_instruction}
```

B.2. Planning Module

As in BAVT, the agent maintains an abstract plan before and during search. The role of the planning module is not to guess the answer, but to identify the missing pieces of evidence, keep track of bridge facts, and prevent the search from revisiting already exhausted directions. The new contribution of our method is not the planner itself, but the fact that the remaining budget explicitly changes which kind of step the agent is allowed to take next.

Planning Module: High-Level Search Plan

```
Before taking stepwise actions, write a brief high-level plan.

- Identify the entities, relations, or attributes that still need evidence.
- Separate direct answer clues from bridge facts that require an intermediate hop.
- Keep the plan abstract; do not invent facts.
- Update the plan only when new evidence changes the remaining uncertainty.
```

B.3. Stage 1: Search-Time Budget Allocation

Stage 1 is the main search-time contribution of our method. Instead of relying only on BAVT's widen-versus-deepen routing, we explicitly allocate the next unit of budget among three action types: SEARCH, DECOMPOSE, and ANSWER. The action decision is made by the three-stage pipeline based on the task-level VOI score in Eqs. (2)–(4); the prompt is only the execution layer that instantiates this decision inside the generator. In other words, the prompt does not choose the action; the controller computes the task-level VOI score. The controller computes a deterministic score from explicit trajectory features, fixed coefficients, budget-aware action costs, and conservative guards, and then injects the chosen action as a deterministic instruction.

User Turn Format for Stage 1

```
{history}

Continue reasoning.
If evidence is not yet sufficient, call exactly one search tool.
If evidence is sufficient, answer now.

Output format:
<thought>your reasoning</thought>
<tool_call>search query</tool_call>
<answer>FINAL_ANSWER</answer>

Rules for this turn:
- Emit either <tool_call> or <answer>, not both, unless the dynamic instruction explicitly allows
    answering now.
- When you use <answer>, include only the shortest correct answer span and no explanation.
- When you use <tool_call>, the query must be concrete, grounded in the question, and non-redundant with
    the immediately preceding failed path.
```

```
- If the question asks for a specific place, date, person, title, or slot value, the answer must use that
    grounded string rather than a broader paraphrase.
- If the current hypothesis is ambiguous, disambiguate the next query explicitly with a type or role.
```

The controller-to-prompt interface is therefore

$$(s_t, b_t) \xrightarrow{\text{Eqs. (2)–(4)}} k_t^\star \in \{\textsc{Search}, \textsc{Decompose}, \textsc{Answer}\} \xrightarrow{\text{prompt injection}} \texttt{dynamic\_instruction}.$$

The four action templates below are the paper-specific prompt components that implement this interface.

Stage 1 Instruction: Search

```
Instruction: SEARCH.
The expected marginal utility of one more retrieval step is highest.
You must call exactly one search tool now.
Do not answer and do not continue with tool-free reasoning.

Inside <thought>, do all of the following:
- State the most specific unresolved fact that still blocks the answer.
- Explain briefly why another retrieval step is preferable to answering now.
- Reuse concrete entities already grounded by the question or evidence.
- If there is ambiguity, state how the query will disambiguate it.

Then emit exactly one <tool_call>.
The query should be targeted, entity-specific, and aimed at closing one missing factual gap rather than
    broad exploration.
Do not output <answer> in this turn.
```

Stage 1 Instruction: Decompose

```
Instruction: DECOMPOSE.
You have been searching without enough progress, and the remaining uncertainty appears to be
    compositional.
You must call exactly one search tool now.
Do not answer and do not continue with tool-free reasoning.

Inside <thought>, do all of the following:
- Identify the missing bridge entity, intermediate fact, or latent relation that connects the hops.
- Summarize what has already been established from the current trajectory.
- Explain why another ordinary search step is less useful than an explicit bridge-fact query.
- Formulate one precise retrieval target that would resolve the compositional gap.

Then emit exactly one <tool_call> with that targeted query.
The query must focus on the missing bridge information, not on re-searching the full question from
    scratch.
Do not output <answer> in this turn.
```

Stage 1 Instruction: Answer

```
Instruction: ANSWER.
The controller judges that answer commitment has higher utility-per-cost than further retrieval.
Do not call tools.

Inside <thought>, briefly verify that the answer is directly supported by the evidence already gathered.
Then return exactly one line in the form <answer>FINAL_ANSWER</answer>.

Answer requirements:
- The answer must be the shortest grounded answer span.
- Do not output explanation outside the tag.
```

```
- If the question is yes/no, FINAL_ANSWER must be exactly yes or no.
- If the question is a binary choice, output only the selected option.
```

Stage 1 Instruction: Budget Backstop

```
Instruction: BUDGET BACKSTOP.
The remaining budget is exhausted or too small for another meaningful retrieval step.
You must answer now and you must not call tools.

Inside <thought>, use only the evidence already present in the trajectory.
Do not speculate beyond the grounded information.
If the evidence is partial, prefer the most directly supported short answer rather than an expansive
    paraphrase.

Return exactly one line in the form <answer>FINAL_ANSWER</answer>.
Keep the answer as short as possible.
If the question is yes/no, FINAL_ANSWER must be exactly yes or no.
```

Table 4 summarizes the Stage-1 interface. The key distinction from vanilla BAVT is that our controller makes an explicit budget-allocation decision over action types before generation. In the released code, we also include simple guardrails to prevent tool-free stalling and to enforce a required search step whenever the controller allocates budget to retrieval, but we do not treat these guardrails as separate method components.

Table 4: **Stage 1 compared with related prompting styles.** Our method differs from prior systems not by adding more free-form prompt complexity, but by using an explicit controller to choose which prompt constraint is injected at each step.

Method	Interleaved retrieval	Step-level instruction	Budget-aware action choice	Conservative finalization
Search-o1	✓	Partial	✗	✗
BAVT	✓	✓	Partial	✗
Ours	✓	✓	✓	✓

Here "budget-aware action choice" means that the action type is selected before generation by an explicit controller rather than inferred from free-form continuation. In our case, the controller first selects SEARCH, DECOMPOSE, or ANSWER; the corresponding instruction is then injected into the generator.

B.4. Stage 2: Answer-Time Finalization

Stage 2 is an answer-selection module applied after search terminates. Starting from the completed trajectory, the system compares the trajectory answer with a refined candidate derived from the same evidence. The goal is not to launch another round of free-form rewriting, but to correct local exactness errors when the evidence clearly supports a safer, more specific answer span. If the case still looks structurally risky, such as unresolved bridge reasoning or comparative semantics, the module abstains and keeps the trajectory answer. Importantly, this stage adds no new tool calls and no additional LLM call during finalization.

We therefore present Stage 2 as an answer-selection card rather than as another generator prompt. The point is precisely that this module is *not* a fresh LLM rewrite call. It is a conservative selection policy over two candidates already available from the completed trace.

Stage 2 Selection Card: Conservative Answer Finalization

```
Inputs:
- Question
- Trajectory answer $a_{\mathrm{base}}$
- Refined candidate $a_{\mathrm{ref}}$
- Supporting evidence from the completed trajectory

Decision principle:
- If the remaining error is only local answer exactness, prefer the better-supported candidate.
- If the case still contains bridge ambiguity, comparative semantics, or path-level uncertainty, abstain.
- Never add a new tool call.
- Never launch an additional LLM call during finalization.
- Treat abstention as the default whenever the gain-risk trade-off is unclear.

Typical positive cases:
- yes/no polarity repair
- binary choice repair
- typed-slot correction
- supported factoid completion

Typical abstention cases:
- unresolved bridge reasoning
- comparative questions
- decomposition-heavy trajectories
- cases where rewriting would likely change semantics rather than improve exactness
```

Table 5 summarizes this Stage-2 policy. The organizing principle is conservative answer finalization: intervene only when the remaining error appears local to answer-form exactness, and keep the trajectory answer whenever the evidence suggests the real failure is path-level.

Table 5: **Stage 2: Answer-time finalization policy.** This module is conservative by design: it rewrites only when the refined candidate is better supported and the risk of changing the semantics remains low.

Case type	Finalization behavior
Unresolved bridge or comparative reasoning	Abstain. If the trajectory still reflects unresolved bridge structure, comparison logic, or other path-level uncertainty, keep the trajectory answer.
Yes/no or binary choice	Finalize when the evidence clearly resolves the polarity or the choice between the listed options. This is the cleanest type of answer-time repair.
Slot-filling exactness	Finalize only when the refined answer adds a small but meaningful amount of specificity, such as a capacity, date, or year range, without changing the underlying fact.
Supported factoid completion	Finalize when the refined candidate simply completes an already supported fact span, for example by restoring a canonical phrase or an omitted attribute.
General fallback	Otherwise finalize only when the refined candidate is better supported by the trajectory evidence and remains comparably concise; abstain in all remaining cases.

This policy is the appendix-level counterpart of Eq. (5). In words, the answer-time module behaves like a high-precision selector rather than a generic editor: it prefers abstention whenever the risk of disturbing a correct trajectory answer outweighs the likely exactness gain.

B.5. Representative Cases

Instead of showing long traces, we summarize representative Stage-2 behaviors as compact decision cards. The first three cards are positive exactness repairs, and the last card is an abstention example showing when the finalizer deliberately refuses to rewrite.

Case Card 1: Binary-Choice Exactness Repair

```
Question:
Which writer was from England, Henry Roth or Robert Erskine Childers?

Trajectory answer:
Robert Erskine Childers

Refined candidate:
Robert Erskine Childers DSC

Evidence:
The supporting evidence identifies Childers as the English writer and also supports the fuller canonical
    name span.

Why Stage 2 finalizes:
- This is a binary-choice question rather than an open-ended generation case.
- The refinement does not change which option is selected.
- The refined string is better supported by the trajectory evidence.
- The residual error is local answer exactness, not path-level reasoning.

Outcome:
Finalize to the fuller supported answer span.
```

Case Card 2: Supported Factoid Completion

```
Question:
What distinction is held by the former NBA player who was a member of the Charlotte Hornets during their
    1992--93 season and was head coach for the Charlotte Sting?

Trajectory answer:
shortest player ever to play in the NBA

Refined candidate:
shortest player ever to play in the National Basketball Association

Evidence:
The trajectory evidence ties Muggsy Bogues to both team clues and supports the fuller canonical statement
     of the distinction.

Why Stage 2 finalizes:
- The refined candidate is not a different claim.
- It is a completion of the same already-supported fact.
- The intervention improves exactness while preserving semantics.
- The gain is local and the rewrite risk is low.

Outcome:
Finalize to the fuller supported factoid span.
```

Case Card 3: Typed-Slot Exactness Repair

```
Question:
The arena where the Lewiston Maineiacs played their home games can seat how many people?

Trajectory answer:
3,677

Refined candidate:
3,677 seated

Evidence:
The supporting passage contains the exact capacity-bearing phrase rather than the bare number alone.
```

```
Why Stage 2 finalizes:
- This is a slot-value correction problem rather than a search failure.
- The refined candidate adds only minimal slot-bearing wording.
- The underlying fact is unchanged.
- The support for the refined phrase is more direct than for the shorter numeric span alone.

Outcome:
Finalize to the typed-slot correction.
```

Case Card 4: Abstention under Bridge Risk

```
Question:
Which performance act has a higher instrument to person ratio, Badly Drawn Boy or Wolf Alice?

Trajectory answer:
Badly Drawn Boy

Refined candidate:
Badly Drawn Boy

Evidence:
The trajectory already supports the current answer, but the question still has comparative and bridge-
    sensitive structure.

Why Stage 2 abstains:
- The remaining risk is in comparative multi-hop reasoning, not answer wording.
- A rewrite would offer little exactness gain.
- In this regime, conservative abstention is safer than intervention.
- The finalizer therefore preserves the trajectory answer instead of forcing an unnecessary change.

Outcome:
Abstain and keep the trajectory answer.
```

These cards illustrate the intended role of answer-time control. Positive interventions repair answer-form errors that remain after the search trajectory is already essentially correct. The abstention example shows the opposite boundary: when the residual uncertainty is still about multi-hop structure rather than wording, Stage 2 keeps the original trajectory answer.

C. Theoretical Statements and Proofs

This appendix provides the formal analysis summarized in Section 4. We first state the main theorem-level results, then give the definitions, assumptions, auxiliary lemmas, and proofs. The analysis supports the two controller layers: search-time budget allocation and answer-time finalization. It is local and decision-level; it does not claim global optimality of the full search tree.

C.1. Main Theoretical Statements

For search-time control, fix step t, state s_t, budget b_t, and feasible action set $\mathcal{K}_t^{\text{feas}}$. Let $V_{t+1}(s, b)$ be the analysis-only oracle continuation value: the scalar maximum expected final reward attainable from state s with remaining budget b under the oracle continuation policy. Define

$$Q_t^\star(k) := \mathbb{E}\left[V_{t+1}\big(S_{t+1}^{(k)},\, b_t - g_t(k)\big) \mid s_t, b_t\right]$$

as the oracle one-step lookahead value, where $S_{t+1}^{(k)}$ is the next state and $g_t(k)$ is the budget charge. With V_t^{stop} denoting immediate-stop value, define $\widetilde{Q}_t^\star(k) := Q_t^\star(k) - V_t^{\text{stop}}(s_t, b_t)$ and decompose

$$\widetilde{Q}_t^\star(k) = \Delta_t^\star(k) + \Psi_t^\star(k) - \Lambda_t^\star(k; b_t) + \xi_t(k),$$

where $\Delta_t^\star(k)$ is the oracle immediate-gain term, $\Psi_t^\star(k)$ the structural residual, $\Lambda_t^\star(k; b_t)$ the local oracle budget shadow-cost term, and $\xi_t(k)$ a smoothness remainder. This decomposition is only an analysis device mirroring Eq. (2). Let k_t be the implemented action and k_t^{opt} an oracle-best feasible action. Define

$$\Gamma_t(k) := \varepsilon_t^\Delta + \varepsilon_t^\Psi + \varepsilon_t^\Pi + \beta_t \|g_t(k)\| + \tfrac{L_t}{2}\|g_t(k)\|^2,$$

where $\varepsilon_t^\Delta, \varepsilon_t^\Psi, \varepsilon_t^\Pi$ bound gain, structural, and budget-penalty approximation errors, β_t bounds shadow-price mismatch, and L_t is the smoothness constant. The oracle counterpart of the task-level VOI score in Eq. (3) is $r_t^\star(k) = [U_t^\star(k; b_t)]_+ / (d_t(k; b_t) + \epsilon)$, where $U_t^\star$ is defined below.

Theorem C.1 (Local approximation of the task-level action utility)**.** *Under the budget smoothness, shadow-price homogeneity, and component-wise approximation conditions in Assumptions C.3–C.5, every feasible action $k \in \mathcal{K}_t^{\text{feas}}$ satisfies*

$$\left| u_t(k) - \widetilde{Q}_t^\star(k) \right| \le \Gamma_t(k). \tag{6}$$

Theorem C.1 covers the task-level action utility $u_t(k)$ and the normalized task-level VOI score $r_t(k)$ before guards. Corollary C.8 below gives a margin condition under which the normalized task-level VOI score ranking is preserved before guards, and Corollary C.9 gives a one-step value-gap bound when the guard layer is compatible with the utility ranking. The controller should be reliable when the score clearly separates feasible actions, while failures are expected when approximation terms in $\Gamma_t(k)$ are large or guards intentionally override raw ranking.

For answer-time finalization, let $\mathcal{Z}$ be the range of z, let $Z := z(\mathcal{T}, a_{\text{base}}, a_{\text{ref}})$, and let $\Delta_{\text{fin}} := R(a_{\text{ref}}, y) - R(a_{\text{base}}, y)$ be the score change from accepting the refined answer. With $(v)_+ = \max\{v, 0\}$, define

$$G^\star(z) := \mathbb{E}[(\Delta_{\text{fin}})_+ \mid Z = z], \qquad H^\star(z) := \mathbb{E}[(-\Delta_{\text{fin}})_+ \mid Z = z]$$

as oracle gain and harm. We consider policies $\pi : \mathcal{Z} \to \{0, 1\}$, where $\pi(z) = 1$ accepts a_{ref}; safe policies satisfy $\pi(z) = 0$ for $z \notin \mathcal{S}_{\text{safe}}$ and the harm constraint in Eq. (1).

Theorem C.2 (Optimal threshold form of safe answer replacement)**.** *Under the strong-duality and multiplier-attainment condition in Assumption C.13, there exists $\eta^\star \ge 1$ such that the policy*

$$\pi^\star(z) = \mathbf{1}\{z \in \mathcal{S}_{\text{safe}} \text{ and } G^\star(z) - \eta^\star H^\star(z) \ge 0\} \tag{7}$$

maximizes $\mathbb{E}[R(\hat{a}, y)]$ subject to $\mathbb{E}[L_{\text{harm}}(\hat{a}, a_{\text{base}}, y)] \le \rho_{\text{harm}}$ and $\pi(z) = 0$ for $z \notin \mathcal{S}_{\text{safe}}$.

C.2. Search-Time Controller: Definitions, Assumptions, and Proof

Fix step t, frontier state s_t, remaining budget b_t, and feasible action set $\mathcal{K}_t^{\text{feas}}$. For each feasible action k, let $S_{t+1}^{(k)}$ be the random next state after executing k and let $g_t(k)$ be its budget charge, which is (s_t, b_t)-measurable. For token costs, this means using the conditional expectation of realized tokens given (s_t, b_t, k) and absorbing residual variance into the remainder. Let $V_{t+1}(s, b)$ be the analysis-only oracle continuation value, i.e., the scalar maximum expected final reward from state s and remaining budget b under the oracle continuation policy and induced future-state/evaluation distribution. The oracle one-step lookahead value is

$$Q_t^\star(k) := \mathbb{E}\left[V_{t+1}(S_{t+1}^{(k)},\, b_t - g_t(k)) \mid s_t, b_t \right], \tag{8}$$

and $k_t^{\text{opt}} \in \arg\max_{k \in \mathcal{K}_t^{\text{feas}}} Q_t^\star(k)$ denotes an oracle-optimal feasible action. Define

$$V_t^{\text{stop}}(s_t, b_t) := \mathbb{E}\left[R(a_t^{\text{stop}}, y) \mid s_t, b_t \right], \tag{9}$$

$$\Delta_t^\star(k) := \mathbb{E}\Big[R(a_t^{\text{stop},(k)}, y) - R(a_t^{\text{stop}}, y) \mid s_t, b_t\Big], \tag{10}$$

$$\Psi_t^\star(k) := \mathbb{E}\Big[V_{t+1}(S_{t+1}^{(k)}, b_t) \mid s_t, b_t\Big] - V_t^{\text{stop}}(s_t, b_t) - \Delta_t^\star(k), \tag{11}$$

where a_t^{stop} is the answer if the trajectory is finalized immediately and $a_t^{\text{stop},(k)}$ is the answer after revealing $S_{t+1}^{(k)}$ and applying the fixed immediate-finalization backstop policy at the pre-charge budget b_t. The budget-adjusted oracle one-step gain is $\widetilde{Q}_t^\star(k) := Q_t^\star(k) - V_t^{\text{stop}}(s_t, b_t)$.

We further define the remaining-budget shaped oracle utility

$$U_t^\star(k; b_t) := \Delta_t^\star(k) + \Psi_t^\star(k) - \Lambda_t^\star(k; b_t), \tag{12}$$

where $\Lambda_t^\star(k; b_t) := \lambda_t^{\star\top} g_t(k)$ is the local oracle budget shadow-cost term. The implemented budget-penalty term $\Pi_t(k; b_t)$ may be signed, for example to encourage ANSWER near exhaustion; its deviation from this oracle shadow-cost is absorbed by ε_t^Π in Assumption C.5. The remainder $\xi_t(k) := \widetilde{Q}_t^\star(k) - U_t^\star(k; b_t)$ is bounded by the smoothness constants below.

Assumption C.3 (Budget smoothness and cost measurability)**.** Fix a norm $\|\cdot\|$ on $\mathbb{R}^2$ and let $\|\cdot\|_*$ denote its dual norm. There exists $L_t \geq 0$ such that, for every feasible action $k \in \mathcal{K}_t^{\text{feas}}$: (i) the map $b \mapsto V_{t+1}(S_{t+1}^{(k)}, b)$ is almost surely twice continuously differentiable near b_t with L_t-Lipschitz gradient, so the Taylor remainder

$$\varrho_t(k) := V_{t+1}\big(S_{t+1}^{(k)}, b_t - g_t(k)\big) - V_{t+1}\big(S_{t+1}^{(k)}, b_t\big) + \nabla_b V_{t+1}\big(S_{t+1}^{(k)}, b_t\big)^\top g_t(k) \tag{13}$$

satisfies $|\varrho_t(k)| \leq \frac{L_t}{2}\|g_t(k)\|^2$ almost surely; and (ii) $g_t(k)$ is (s_t, b_t)-measurable.

Assumption C.4 (Local shadow-price homogeneity)**.** There exist $\lambda_t^\star \in \mathbb{R}_+^2$ and $\beta_t \geq 0$ such that, for every feasible action $k \in \mathcal{K}_t^{\text{feas}}$,

$$\Big\|\mathbb{E}\Big[\nabla_b V_{t+1}(S_{t+1}^{(k)}, b_t) \mid s_t, b_t\Big] - \lambda_t^\star\Big\|_* \leq \beta_t. \tag{14}$$

Assumption C.5 (Component-wise approximation)**.** There exist $\varepsilon_t^\Delta, \varepsilon_t^\Psi, \varepsilon_t^\Pi \geq 0$ such that, for every $k \in \mathcal{K}_t^{\text{feas}}$,

$$\big|\widehat{\Delta}_t(k) - \Delta_t^\star(k)\big| \leq \varepsilon_t^\Delta, \quad \big|\Psi_t(k) - \Psi_t^\star(k)\big| \leq \varepsilon_t^\Psi, \quad \big|\Pi_t(k; b_t) - \Lambda_t^\star(k; b_t)\big| \leq \varepsilon_t^\Pi. \tag{15}$$

Assumption C.6 (Ranking-preserving guard condition)**.** The guard operator $\mathfrak{G}_t$ preserves the relevant u_t ranking in the following sense: the implemented action $k_t \in \arg\max_{k \in \mathcal{K}_t^{\text{feas}}} \widetilde{\mathcal{J}}_t(k)$ satisfies $u_t(k_t) \geq u_t(k_t^{\text{opt}})$, where $k_t^{\text{opt}} \in \arg\max_{k \in \mathcal{K}_t^{\text{feas}}} Q_t^\star(k)$. Equivalently, the guard layer does not promote an action with strictly lower u_t score above the oracle-optimal action.

Lemma C.7 (Budget linearization)**.** *Under Assumptions C.3 and C.4, for every* $k \in \mathcal{K}_t^{\text{feas}}$,

$$\widetilde{Q}_t^\star(k) = \Delta_t^\star(k) + \Psi_t^\star(k) - \lambda_t^{\star\top} g_t(k) + \xi_t(k),$$

where $|\xi_t(k)| \leq \beta_t\|g_t(k)\| + \frac{L_t}{2}\|g_t(k)\|^2$.

Proof. Write $S := S_{t+1}^{(k)}$, $g := g_t(k)$, $V := V_{t+1}$. By Assumption C.3(i),

$$V(S, b_t - g) = V(S, b_t) - \nabla_b V(S, b_t)^\top g + \varrho_t(k), \quad |\varrho_t(k)| \leq \tfrac{L_t}{2}\|g\|^2.$$

Taking conditional expectation and using Assumption C.3(ii), so g pulls out of the expectation,

$$Q_t^\star(k) = \mathbb{E}[V(S, b_t) \mid s_t, b_t] - \mathbb{E}[\nabla_b V(S, b_t) \mid s_t, b_t]^\top g + \bar{\varrho}_t(k),$$

where $|\bar{\varrho}_t(k)| \leq \frac{L_t}{2}\|g\|^2$. By definition of $\Psi_t^\star(k)$, $\mathbb{E}[V(S, b_t) \mid s_t, b_t] = V_t^{\text{stop}} + \Delta_t^\star(k) + \Psi_t^\star(k)$. Subtracting V_t^{stop} and writing $\xi_t(k) := (\lambda_t^\star - \mathbb{E}[\nabla_b V(S, b_t) \mid s_t, b_t])^\top g + \bar{\varrho}_t(k)$ gives the identity. The bound follows from Hölder's inequality and Assumption C.4. □

C.3. Proof of Theorem C.1 and Corollary C.9

Proof of Theorem C.1. Fix $k \in \mathcal{K}_t^{\text{feas}}$. By definition of $u_t(k)$ in Eq. (2) and $U_t^\star(k; b_t)$ in Eq. (12),

$$u_t(k) - U_t^\star(k; b_t) = \big(\widehat{\Delta}_t(k) - \Delta_t^\star(k)\big) + \big(\Psi_t(k) - \Psi_t^\star(k)\big) - \big(\Pi_t(k; b_t) - \Lambda_t^\star(k; b_t)\big).$$

Applying the triangle inequality and Assumption C.5 gives $|u_t(k) - U_t^\star(k; b_t)| \le \varepsilon_t^\Delta + \varepsilon_t^\Psi + \varepsilon_t^\Pi$. By Lemma C.7, $|\widetilde{Q}_t^\star(k) - U_t^\star(k; b_t)| = |\xi_t(k)| \le \beta_t \|g_t(k)\| + \frac{L_t}{2}\|g_t(k)\|^2$. The triangle inequality then gives

$$|u_t(k) - \widetilde{Q}_t^\star(k)| \le \varepsilon_t^\Delta + \varepsilon_t^\Psi + \varepsilon_t^\Pi + \beta_t \|g_t(k)\| + \tfrac{L_t}{2}\|g_t(k)\|^2 = \Gamma_t(k),$$

which is Eq. (6). □

Although Theorem C.1 is stated for the unnormalized utility, action selection uses the normalized score in Eq. (3). The next corollary shows that the normalized ranking is stable when the oracle score margin dominates the approximation terms.

Corollary C.8 (Ranking consistency under a margin condition)**.** *If*

$$r_t^\star(i) - r_t^\star(j) > \frac{\Gamma_t(i) + \Gamma_t(j)}{d_{\min} + \epsilon} \tag{16}$$

for some $d_{\min} > 0$ lower-bounding all denominators, then $r_t(i) > r_t(j)$.

Proof. Define $r_t^\star(k) := [U_t^\star(k; b_t)]_+ / (d_t(k; b_t) + \epsilon)$. Since $[\cdot]_+$ is 1-Lipschitz,

$$|r_t(k) - r_t^\star(k)| = \frac{|[u_t(k)]_+ - [U_t^\star(k; b_t)]_+|}{d_t(k; b_t) + \epsilon} \le \frac{|u_t(k) - U_t^\star(k; b_t)|}{d_{\min} + \epsilon} \le \frac{\Gamma_t(k)}{d_{\min} + \epsilon}.$$

If $r_t^\star(i) - r_t^\star(j) > (\Gamma_t(i) + \Gamma_t(j))/(d_{\min} + \epsilon)$, then

$$r_t(i) \ge r_t^\star(i) - \frac{\Gamma_t(i)}{d_{\min} + \epsilon} > r_t^\star(j) + \frac{\Gamma_t(j)}{d_{\min} + \epsilon} \ge r_t(j),$$

which proves the result. □

The ranking statement does not yet include the guard layer. Under the ranking-preserving guard condition, the same local bound implies a one-step value-gap guarantee for the implemented action. This result is conditional: it isolates the loss due to score approximation when the guard layer does not overturn the utility ordering relevant to the oracle action, and it is not a global optimality guarantee for arbitrary guard settings.

Corollary C.9 (One-step value gap under ranking-preserving guards)**.** *Under the conditions of Theorem C.1 and the ranking-preserving guard condition in Assumption C.6, the implemented action satisfies*

$$Q_t^\star(k_t^{\text{opt}}) - Q_t^\star(k_t) \le 2 \max_{k \in \mathcal{K}_t^{\text{feas}}} \Gamma_t(k). \tag{17}$$

Proof of Corollary C.9. Let $\bar{\Gamma}_t := \max_{k \in \mathcal{K}_t^{\text{feas}}} \Gamma_t(k)$. From Eq. (6), $\widetilde{Q}_t^\star(k) \le u_t(k) + \bar{\Gamma}_t$ and $\widetilde{Q}_t^\star(k) \ge u_t(k) - \bar{\Gamma}_t$ for every k. By Assumption C.6, $u_t(k_t) \ge u_t(k_t^{\text{opt}})$. Therefore,

$$\widetilde{Q}_t^\star(k_t^{\text{opt}}) \le u_t(k_t^{\text{opt}}) + \bar{\Gamma}_t \le u_t(k_t) + \bar{\Gamma}_t \le \widetilde{Q}_t^\star(k_t) + 2\bar{\Gamma}_t.$$

Adding V_t^{stop} to both sides gives $Q_t^\star(k_t^{\text{opt}}) - Q_t^\star(k_t) \le 2\bar{\Gamma}_t$, which is Eq. (17). □

Remark C.10. The ranking-consistency result covers the normalized raw score $r_t(k)$ before guards. Guards further modify the executable score $\widetilde{\mathcal{J}}_t(k)$ in action-specific ways, so oracle ranking of r_t does not directly imply oracle ranking of $\widetilde{\mathcal{J}}_t$.

C.4. Answer-Time Finalization: Reward–Harm Decomposition and Optimality

We now collect the answer-time quantities used in Theorem C.2. Let $(x, \mathcal{T}, a_{\text{base}}, a_{\text{ref}}, y)$ be jointly distributed according to the evaluation distribution, where $a_{\text{ref}} = g_{\text{ref}}(\mathcal{T}, a_{\text{base}})$, let $\mathcal{Z}$ denote the range of z, let $Z := z(\mathcal{T}, a_{\text{base}}, a_{\text{ref}})$, and let an answer-replacement policy be a measurable map $\pi : \mathcal{Z} \to \{0, 1\}$. We restrict attention to safe policies satisfying $\pi(z) = 0$ for $z \notin \mathcal{S}_{\text{safe}}$. Define $\Delta_{\text{fin}} := R(a_{\text{ref}}, y) - R(a_{\text{base}}, y)$ and

$$G^\star(z) := \mathbb{E}[(\Delta_{\text{fin}})_+ \mid Z = z], \qquad H^\star(z) := \mathbb{E}[(-\Delta_{\text{fin}})_+ \mid Z = z]. \tag{18}$$

We now turn from search-time action selection to answer-time answer replacement. The first step is to express finalization as a gain–harm trade-off relative to the base trajectory answer.

Lemma C.11 (Reward–harm decomposition). *For any safe answer-replacement policy π,*

$$\mathbb{E}[R(\hat{a}, y)] - \mathbb{E}[R(a_{\text{base}}, y)] = \mathbb{E}\big[\pi(Z)\big(G^\star(Z) - H^\star(Z)\big)\big], \tag{19}$$

$$\mathbb{E}[L_{\text{harm}}(\hat{a}, a_{\text{base}}, y)] = \mathbb{E}[\pi(Z) H^\star(Z)]. \tag{20}$$

Proof. Fix a safe policy π. By definition, $\hat{a} = a_{\text{ref}}$ if $\pi(Z) = 1$ and $\hat{a} = a_{\text{base}}$ if $\pi(Z) = 0$, so $R(\hat{a}, y) - R(a_{\text{base}}, y) = \pi(Z)\Delta_{\text{fin}}$. Taking expectations, conditioning on Z, and using $\Delta_{\text{fin}} = (\Delta_{\text{fin}})_+ - (-\Delta_{\text{fin}})_+$ yields Eq. (19). Likewise, $L_{\text{harm}}(\hat{a}, a_{\text{base}}, y) = \pi(Z)(-\Delta_{\text{fin}})_+$, and conditioning on Z yields Eq. (20). □

By Lemma C.11, maximizing the answer-time part of Eq. (1) is equivalent to

$$\max_{\pi} \ \mathbb{E}\big[\pi(Z)\big(G^\star(Z) - H^\star(Z)\big)\big] \quad \text{s.t.} \quad \mathbb{E}[\pi(Z) H^\star(Z)] \le \rho_{\text{harm}}, \qquad \pi(z) = 0 \text{ for } z \notin \mathcal{S}_{\text{safe}}, \tag{21}$$

where the maximum is over measurable $\{0, 1\}$-valued policies. For $\eta > 0$ and $\tau \in \mathbb{R}$, write $F^\star_{\eta,\tau}(z) := G^\star(z) - \eta H^\star(z) - \tau$.

Lemma C.12 (Pointwise maximizer of the penalized objective). *Fix $\eta > 0$ and $\tau \in \mathbb{R}$. Among all measurable policies satisfying $\pi(z) = 0$ for $z \notin \mathcal{S}_{\text{safe}}$, the maximizer of $\mathbb{E}[\pi(Z) F^\star_{\eta,\tau}(Z)]$ is*

$$\pi^\star_{\eta,\tau}(z) = \mathbf{1}\Big\{z \in \mathcal{S}_{\text{safe}} \text{ and } F^\star_{\eta,\tau}(z) \ge 0\Big\}. \tag{22}$$

Proof. Fix $z \in \mathcal{S}_{\text{safe}}$. Because $\pi(z) \in \{0, 1\}$, the pointwise contribution $\pi(z) F^\star_{\eta,\tau}(z)$ equals either 0 or $F^\star_{\eta,\tau}(z)$. Hence the maximizing choice is $\pi(z) = 1$ when $F^\star_{\eta,\tau}(z) \ge 0$ and $\pi(z) = 0$ otherwise. Outside $\mathcal{S}_{\text{safe}}$, feasibility forces $\pi(z) = 0$. Since the objective is the expectation of this pointwise-separable integrand, the resulting rule is a global maximizer. □

Assumption C.13 (Strong duality and multiplier attainment). There exists a multiplier $\gamma^\star \ge 0$ such that: (i) strong duality holds—the constrained problem in Eq. (21) and its Lagrangian relaxation have the same optimal value; (ii) the Lagrangian relaxation attains its maximum at $\gamma^\star$; (iii) the policy $\pi_{\gamma^\star}(z) := \mathbf{1}\{z \in \mathcal{S}_{\text{safe}} \text{ and } G^\star(z) - (1 + \gamma^\star) H^\star(z) \ge 0\}$ is primal-feasible, i.e., $\mathbb{E}[\pi_{\gamma^\star}(Z) H^\star(Z)] \le \rho_{\text{harm}}$; and (iv) $\pi_{\gamma^\star}$ attains the constrained optimum, i.e., $\pi_{\gamma^\star}$ achieves the maximum of Eq. (21).

This is a population-level regularity condition for the oracle constrained selection problem; the released deterministic finalizer below is a conservative feature-rule instantiation and is not claimed to know $G^\star$ or $H^\star$.

Proof of Theorem C.2. By Lemma C.11, the safe answer-replacement problem is Eq. (21). For any multiplier $\gamma \geq 0$, its Lagrangian is

$$\mathcal{L}(\pi, \gamma) := \mathbb{E}\big[\pi(Z)\big(G^\star(Z) - H^\star(Z)\big)\big] - \gamma\left(\mathbb{E}[\pi(Z)H^\star(Z)] - \rho_{\text{harm}}\right) \tag{23}$$

$$= \mathbb{E}\big[\pi(Z)\big(G^\star(Z) - (1+\gamma)H^\star(Z)\big)\big] + \gamma\rho_{\text{harm}}. \tag{24}$$

For fixed γ, the term $\gamma\rho_{\text{harm}}$ is constant in π, so maximizing $\mathcal{L}(\pi, \gamma)$ is equivalent to maximizing $\mathbb{E}[\pi(Z)(G^\star(Z) - (1+\gamma)H^\star(Z))]$ subject to the safe-set constraint. Applying Lemma C.12 with $\eta = 1+\gamma$ and $\tau = 0$ shows that the pointwise maximizer is

$$\pi_\gamma(z) = \mathbf{1}\{z \in \mathcal{S}_{\text{safe}} \text{ and } G^\star(z) - (1+\gamma)H^\star(z) \geq 0\}\,. \tag{25}$$

By Assumption C.13(i)–(ii), there exists $\gamma^\star \geq 0$ at which the Lagrangian relaxation attains the constrained optimum. By Assumption C.13(iii), $\pi_{\gamma^\star}$ is primal-feasible. By Assumption C.13(iv), $\pi_{\gamma^\star}$ achieves the primal optimum. Setting $\eta^\star := 1 + \gamma^\star$ gives $\eta^\star \geq 1$, and substituting into Eq. (25) yields exactly the threshold rule in Eq. (7). □

C.5. Released Rule-Based Finalizer: Exact Rule Characterization

Remark C.14. The released deterministic finalizer is *not* a runtime instantiation of the oracle threshold rule in Theorem C.2: the oracle quantities $G^\star(z)$ and $H^\star(z)$ are unknown at inference time. Instead, it is a conservative deterministic rule over explicit feature conditions, constructed as a special case of the threshold form. Theorem C.2 characterizes the performance upper envelope when $G^\star, H^\star$ are known; Proposition C.15 characterizes the rule that the released code actually executes below that envelope. The empirical gap between the two is an open question.

The oracle threshold rule uses unknown conditional quantities $G^\star$ and $H^\star$. The reported system therefore uses a deterministic conservative feature rule. The next proposition characterizes that rule at the paper level.

Define the following paper-level features. Let m_{ref} indicate that a refined candidate is available and passes the preliminary plausibility filter; let c_{risk} be the refinement-risk category; let n_{dec} be the number of decomposition steps in the trajectory; let q_{type} be the detected question/answer type; let q_{slot} be the detected slot type; let Δ_{sup} be the support gain of the refined candidate over the base answer; and let ℓ_{base} and ℓ_{ref} be the token lengths of the base and refined candidates. Let $\mathcal{C}_{\text{block}}$ denote the high-risk refinement categories blocked by the finalizer, let $\mathcal{Q}_{\text{bin}}$ denote yes/no and binary-choice cases, and let $\mathcal{Q}_{\text{typed}}$ denote typed-slot cases such as capacity, date, and year range. Define four branch condition sets:

$$\begin{aligned}
\mathcal{B}_{\text{bin}} &:= \{q_{\text{type}} \in \mathcal{Q}_{\text{bin}}\},\\
\mathcal{B}_{\text{typed}} &:= \{q_{\text{slot}} \in \mathcal{Q}_{\text{typed}},\ \Delta_{\text{sup}} \geq 0.50,\ \ell_{\text{ref}} \leq \ell_{\text{base}} + 1\},\\
\mathcal{B}_{\text{explicit}} &:= \{\text{explicit factoid-slot case},\ \Delta_{\text{sup}} \geq 0,\ \ell_{\text{ref}} \leq \ell_{\text{base}} + 3\},\\
\mathcal{B}_{\text{compact}} &:= \{\Delta_{\text{sup}} > 0,\ \ell_{\text{ref}} \leq \ell_{\text{base}} + 2\}.
\end{aligned}$$

Define the priority-ordered branch selector:

$$\mathcal{B}_{\text{accept}} := \begin{cases}
\mathcal{B}_{\text{bin}}, & \text{if } q_{\text{type}} \in \mathcal{Q}_{\text{bin}},\\
\mathcal{B}_{\text{typed}}, & \text{if } q_{\text{type}} \notin \mathcal{Q}_{\text{bin}} \text{ and } q_{\text{slot}} \in \mathcal{Q}_{\text{typed}},\\
\mathcal{B}_{\text{explicit}}, & \text{if } q_{\text{type}} \notin \mathcal{Q}_{\text{bin}},\ q_{\text{slot}} \notin \mathcal{Q}_{\text{typed}}, \text{ and the case is an explicit factoid slot},\\
\mathcal{B}_{\text{compact}}, & \text{otherwise.}
\end{cases}$$

The accept set is not an unconditional union of the branches: when an earlier applicable branch fails its safety test, the finalizer abstains instead of falling through to a lower-priority branch. The accept set is

$$\mathcal{S}_{\text{det}} := \{m_{\text{ref}} = 1,\ c_{\text{risk}} \notin \mathcal{C}_{\text{block}},\ n_{\text{dec}} = 0\} \cap \mathcal{B}_{\text{accept}}.$$

Proposition C.15 (Exact rule of the released deterministic finalizer). *Let z denote the finalization feature vector extracted from the completed trajectory and the two candidate answers. The deterministic finalizer used in our experiments is equivalent to the indicator rule $\sigma_{\text{det}}(z) = \mathbf{1}\{z \in \mathcal{S}_{\text{det}}\}$ for the priority-ordered safe set $\mathcal{S}_{\text{det}}$. Therefore, the final answer is a_{ref} when $z \in \mathcal{S}_{\text{det}}$ and a_{base} otherwise.*

Proof. Trace the rule in its evaluation order. The first gate requires that a refined candidate is available and passes the preliminary plausibility filter, which enforces $m_{\text{ref}} = 1$. The second gate blocks high-risk refinement categories, enforcing $c_{\text{risk}} \notin \mathcal{C}_{\text{block}}$. The third gate abstains on decomposition-heavy trajectories, enforcing $n_{\text{dec}} = 0$. Conditional on these gates, the branch selector is evaluated in priority order. Binary and yes/no cases are handled first, giving the branch set $\mathcal{B}_{\text{bin}}$. If the case is not binary but has a typed slot, the typed-slot safety test is applied; acceptance requires $\Delta_{\text{sup}} \geq 0.50$ and $\ell_{\text{ref}} \leq \ell_{\text{base}} + 1$, and failure causes abstention rather than falling through. If no earlier branch applies and the case is an explicit factoid slot, acceptance requires nonnegative support gain and $\ell_{\text{ref}} \leq \ell_{\text{base}} + 3$; again, failure causes abstention. The final fallback accepts only compact support-improving refinements, requiring $\Delta_{\text{sup}} > 0$ and $\ell_{\text{ref}} \leq \ell_{\text{base}} + 2$. Thus the rule accepts exactly when the three gates hold and the priority-ordered branch selector accepts, which is precisely $z \in \mathcal{S}_{\text{det}}$. □

C.6. Plug-in Approximation of the Finalizer

The previous proposition describes the released rule. For completeness, we also record a generic perturbation result showing how approximate gain–harm scores affect threshold decisions: Lemma C.16 controls the pointwise score error, and Theorem C.17 converts it into a penalized-value bound.

Fix $\eta > 0$ and $\tau \in \mathbb{R}$. Write $F^\star_{\eta,\tau}(z) := G^\star(z) - \eta H^\star(z) - \tau$ and $F_{\eta,\tau}(z) := G(z) - \eta H(z) - \tau$, define the oracle rule by $\pi^\star_{\eta,\tau}(z) := \mathbf{1}\{z \in \mathcal{S}_{\text{safe}} \text{ and } F^\star_{\eta,\tau}(z) \geq 0\}$, and define the plug-in rule by $\hat{\pi}_{\eta,\tau}(z) := \mathbf{1}\{z \in \mathcal{S}_{\text{safe}} \text{ and } F_{\eta,\tau}(z) \geq 0\}$.

Lemma C.16 (Finalizer score perturbation). *If $|G(z) - G^\star(z)| \leq \delta_G$ and $|H(z) - H^\star(z)| \leq \delta_H$ for all $z \in \mathcal{S}_{\text{safe}}$, then*

$$|F_{\eta,\tau}(z) - F^\star_{\eta,\tau}(z)| \leq \delta_G + \eta\,\delta_H$$

for all $z \in \mathcal{S}_{\text{safe}}$.

Proof. For $z \in \mathcal{S}_{\text{safe}}$, write $F_{\eta,\tau}(z) - F^\star_{\eta,\tau}(z) = \big(G(z) - G^\star(z)\big) - \eta\big(H(z) - H^\star(z)\big)$ and apply the triangle inequality. □

Theorem C.17 (Plug-in excess penalized value). *Fix $\eta > 0$ and $\tau \in \mathbb{R}$, and let $\delta := \delta_G + \eta\,\delta_H$. Under the bounds of Lemma C.16,*

$$\mathbb{E}\Big[\pi^\star_{\eta,\tau}(Z)F^\star_{\eta,\tau}(Z)\Big] - \mathbb{E}\Big[\hat{\pi}_{\eta,\tau}(Z)F^\star_{\eta,\tau}(Z)\Big] \leq \mathbb{E}\Big[|F^\star_{\eta,\tau}(Z)|\mathbf{1}\Big\{|F^\star_{\eta,\tau}(Z)| \leq \delta,\ Z \in \mathcal{S}_{\text{safe}}\Big\}\Big].$$

If, in addition, there exist constants $C > 0$ and $\alpha > 0$ such that

$$\mathbb{P}\Big(|F^\star_{\eta,\tau}(Z)| \leq u,\ Z \in \mathcal{S}_{\text{safe}}\Big) \leq Cu^\alpha \qquad \text{for all } u \geq 0,$$

then

$$\mathbb{E}\Big[\pi^\star_{\eta,\tau}(Z)F^\star_{\eta,\tau}(Z)\Big] - \mathbb{E}\Big[\hat{\pi}_{\eta,\tau}(Z)F^\star_{\eta,\tau}(Z)\Big] \leq C\delta^{1+\alpha}.$$

Proof. Let $D := \{Z \in \mathcal{S}_{\text{safe}} : \pi^\star_{\eta,\tau}(Z) \neq \hat{\pi}_{\eta,\tau}(Z)\}$. Because $\pi^\star_{\eta,\tau}$ is the pointwise maximizer of $\pi(Z)F^\star_{\eta,\tau}(Z)$ over $\pi(Z) \in \{0,1\}$, the difference

$$\mathbb{E}\Big[\pi^\star_{\eta,\tau}(Z)F^\star_{\eta,\tau}(Z)\Big] - \mathbb{E}\Big[\hat{\pi}_{\eta,\tau}(Z)F^\star_{\eta,\tau}(Z)\Big]$$

is supported on D and equals $\mathbb{E}[|F^\star_{\eta,\tau}(Z)|\mathbf{1}_D]$. On D, the oracle and plug-in scores have opposite signs, so $|F^\star_{\eta,\tau}(Z)| \le |F^\star_{\eta,\tau}(Z) - F_{\eta,\tau}(Z)|$. Applying Lemma C.16 gives $D \subseteq \{|F^\star_{\eta,\tau}(Z)| \le \delta,\ Z \in \mathcal{S}_{\text{safe}}\}$ and therefore the first bound. For the second bound, use $|F^\star_{\eta,\tau}(Z)| \le \delta$ on the boundary set to obtain

$$\mathbb{E}\Big[|F^\star_{\eta,\tau}(Z)|\mathbf{1}\Big\{|F^\star_{\eta,\tau}(Z)| \le \delta,\ Z \in \mathcal{S}_{\text{safe}}\Big\}\Big] \le \delta\,\mathbb{P}\Big(|F^\star_{\eta,\tau}(Z)| \le \delta,\ Z \in \mathcal{S}_{\text{safe}}\Big) \le C\delta^{1+\alpha}.$$

□

C.7. Finite Termination of Algorithm 1

Finally, Corollary C.18 records that the inference procedure is finite under a minimal budget-decrement condition. The sum $B_{\text{tool}} + B_{\text{tok}}$ is used only as a bookkeeping potential over two nonnegative budget coordinates, not as a single physical cost metric.

Corollary C.18 (Finite termination). *Algorithm 1 terminates in at most $\lceil (B_{\text{tool}} + B_{\text{tok}})/\zeta \rceil$ iterations under the minimum-decrement condition below.*

Lemma C.19 (Finite termination lemma). *Assume that every executed action has nonnegative cost in each budget coordinate and that, whenever the `while`-loop in Algorithm 1 continues, at least one coordinate of the remaining budget decreases by at least a fixed amount $\zeta > 0$. If the initial budget $b_0 = (B_{\text{tool}}, B_{\text{tok}})$ is finite, then the `while`-loop terminates after at most $\left\lceil \frac{B_{\text{tool}}+B_{\text{tok}}}{\zeta} \right\rceil$ iterations. This proves Corollary C.18.*

Proof. Write $S_t := b_{\text{tool},t} + b_{\text{tok},t}$. Since each continuing iteration decreases at least one coordinate by at least ζ and neither coordinate increases, we have $S_{t+1} \le S_t - \zeta$. Because $S_0 = B_{\text{tool}} + B_{\text{tok}}$ and $S_t \ge 0$ for all t, the loop can continue at most $\lceil (B_{\text{tool}} + B_{\text{tok}})/\zeta \rceil$ times. The post-loop extraction and finalization steps are finite, so Algorithm 1 always returns in finitely many steps. □

D. Experimental Protocol and Implementation Details

Fixed controller implementation. All audited experiments were run on a workstation with 4 NVIDIA RTX 5880 Ada GPUs. The controller itself is deterministic and lightweight; the dominant cost comes from LLM inference and retrieval calls. The Stage-1 controller is a fixed, training-free scoring rule added on top of the BAVT search procedure. BAVT provides the planner, generator, critic interface, search procedure, and remaining-budget state; our added component is the task-level VOI score action scorer over SEARCH, DECOMPOSE, and ANSWER. No coefficients are learned from labeled trajectories, no regression model is fitted, and no benchmark-specific hyperparameter search is performed over the reported audit cells. The same controller parameters are used across all benchmarks, budget levels, and LLM backbones.

Budget pressure and action scoring. We use the normalized remaining-budget pressure

$$\rho_t = 1 - \min\left\{\frac{b_{\text{tool},t}}{B_{\text{tool}}}, \frac{b_{\text{tok},t}}{B_{\text{tok}}}\right\},$$

clipped to $[0,1]$. This scalar increases as either the tool-call or output-token budget becomes tight. It enters both the budget-dependent penalty term $\Pi_t(k; b_t)$ and the action-specific cost scale $d_t(k; b_t)$: additional SEARCH and DECOMPOSE steps are penalized more strongly under tight budgets, while ANSWER becomes relatively more attractive when the trajectory has sufficient support.

Fixed coefficients and guards. The main fixed Stage-1 coefficients are a cost-penalty scale of 0.7, a decomposition bonus of 0.14, and an early-answer penalty of 0.18. These values remain fixed in all reported experiments. After forming the utility numerator, the controller divides positive utility by an action-specific cost scale and then applies deterministic guards. The guards suppress premature answer commitment under weak support, suppress decomposition for low-compositionality or factoid-like questions, downweight repeated decomposition after stagnation, and enforce minimum search for sufficiently compositional states. These guards only change the executable action score; they do not change the retrieval backend, generator, sample order, or budget accounting protocol.

Stage-2 finalization. The Stage-2 finalizer is a deterministic answer-selection rule over the completed trajectory. It compares the base answer a_{base} with a refined candidate a_{ref} derived from the same trace and accepts the refined candidate only under low-risk answer-form conditions, such as yes/no polarity repair, binary-choice repair, typed-slot correction, or supported factoid completion. It abstains under unresolved bridge structure, unresolved comparative reasoning, or missing direct support. This stage adds no tool calls and issues no additional LLM call during finalization.

E. Full Qwen3-32B Main Results

F. Qwen3.5-122B Backbone-Sensitivity Results

Figure 6 reports the full budget scaling curves for the Qwen3.5-122B backbone across all four benchmarks. Compared with Qwen3-32B and GPT-5.4-Mini (Figure 3), Qwen3.5-122B exhibits a more mixed competitive regime: BATS and BAVT lead in several cells, particularly at upper budgets, while VOI maintains an advantage at lower budgets where explicit budget penalty provides the largest marginal benefit.

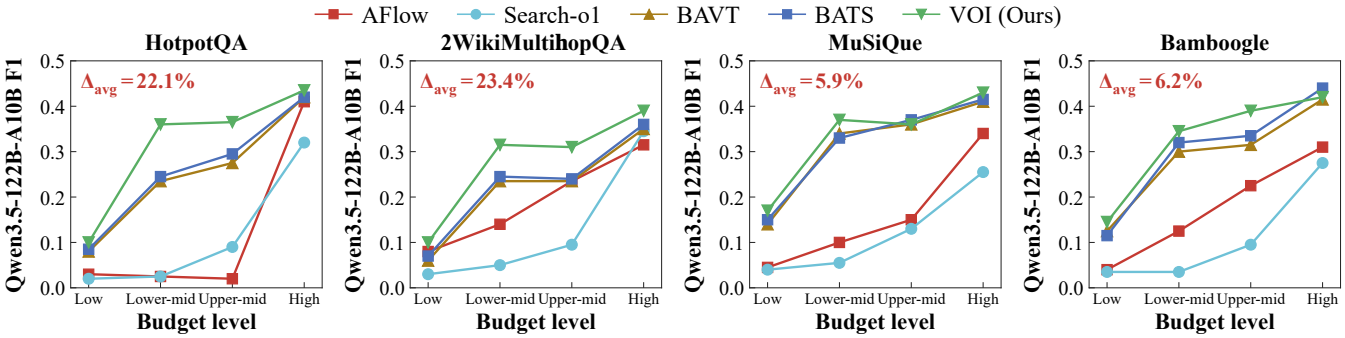

Figure 6: **Budget scaling curves for Qwen3.5-122B across four datasets.** Each column corresponds to one benchmark; all methods are evaluated under the shared dual-budget protocol. See Figure 3 for Qwen3-32B and GPT-5.4-Mini results.

G. Stage-1 Component Ablation Protocol

The ablation in Section 5.4 is designed to match the Stage 1 controller in Section 4.1. Each variant removes exactly one component from the three-stage scoring pipeline while keeping the search procedure, prompts, retrieval backend, sample set, and hard-budget audit unchanged. The *w/o penalty* variant removes the budget-dependent penalty term $\Pi_t(k; b_t)$ from Eq. (2); *w/o normalization* replaces Eq. (3) with the unnormalized score $[u_t(k)]_+$; *w/o structural* removes $\Psi_t(k)$ from Eq. (2); and *w/o guards* bypasses $\mathfrak{G}_t$ in Eq. (4). The full benchmark-level results are reported in Table 1 in the main text.

Table 6: **Main strict-budget results on Qwen3-32B.** Each method reports `EM/F1` under the same symmetric strict dual-budget protocol with tool-call and output-token caps $(1,100)$, $(2,200)$, $(2,300)$, and $(3,500)$. VOI denotes our full two-stage budget-control method. The 1st, 2nd, and 3rd best results are highlighted in **first**, second, and third colors. Cell background colors are ranked by the average of EM and F1 scores. Best EM and F1 are independently bolded. ↑ indicates higher is better.

Benchmark	Budget	BATS	BAVT	AFlow	Search-o1	VOI
		EM/F1 ↑	EM/F1 ↑	EM/F1 ↑	EM/F1 ↑	EM/F1 ↑
Bamboogle	`low`	0.02/0.03	0.11/0.17	0.03/0.03	0.00/0.02	**0.15/0.21**
Bamboogle	`lower-mid`	0.05/0.06	0.21/0.28	**0.33/0.42**	0.03/0.05	**0.33**/0.42
Bamboogle	`upper-mid`	**0.42/0.54**	0.33/0.42	0.33/0.42	0.20/0.24	**0.43**/0.53
Bamboogle	`high`	0.45/0.58	0.39/0.47	0.37/0.50	0.33/0.44	**0.48/0.62**
HotpotQA	`low`	0.01/0.02	0.11/0.14	0.01/0.01	0.00/0.00	**0.14/0.17**
HotpotQA	`lower-mid`	0.02/0.04	0.28/0.34	0.01/0.01	0.03/0.04	**0.36/0.41**
HotpotQA	`upper-mid`	**0.40/0.49**	0.34/0.41	0.01/0.01	0.09/0.14	0.39/0.47
HotpotQA	`high`	**0.43/0.52**	0.34/0.40	0.29/0.36	0.22/0.31	**0.43**/0.52
MuSiQue	`low`	0.02/0.03	0.11/**0.17**	0.01/0.01	0.00/0.02	**0.12**/0.16
MuSiQue	`lower-mid`	0.06/0.08	0.26/0.33	0.03/0.04	0.03/0.06	**0.28/0.35**
MuSiQue	`upper-mid`	**0.35/0.43**	0.26/0.34	0.06/0.08	0.16/0.19	0.33/0.40
MuSiQue	`high`	0.37/0.48	0.34/0.42	0.11/0.16	0.29/0.35	**0.40/0.50**
2WikiMultihopQA	`low`	0.02/0.02	0.08/0.10	0.07/0.07	0.00/0.04	**0.09/0.11**
2WikiMultihopQA	`lower-mid`	0.05/0.05	0.28/0.33	**0.47/0.55**	0.03/0.04	0.44/0.49
2WikiMultihopQA	`upper-mid`	0.53/**0.64**	0.40/0.45	0.47/0.55	0.15/0.20	**0.54**/0.62
2WikiMultihopQA	`high`	0.62/**0.73**	0.56/0.63	0.58/0.70	0.27/0.36	**0.64**/0.71
Average		0.24/0.30	0.28/0.34	0.20/0.24	0.11/0.16	**0.35/0.42**

Table 7: **Cross-backbone macro deltas for VOI against four baselines.** ΔF1 and ΔEM are averaged over the 48 audited cells (3 backbones × 4 benchmarks × 4 budgets), and their 95% CI is a cell-level bootstrap confidence interval.

Comparison	ΔEM (est.)	95% CI	ΔF1	95% CI
VOI vs AFlow	0.2456	[0.2100, 0.2819]	0.3021	[0.2556, 0.3491]
VOI vs BATS	0.1081	[0.0437, 0.1725]	0.1219	[0.0439, 0.1999]
VOI vs Search-o1	0.2734	[0.2360, 0.3112]	0.2993	[0.2625, 0.3370]
VOI vs BAVT	0.0492	[0.0382, 0.0603]	0.0530	[0.0423, 0.0639]

H. Additional Audit Results and Usage Diagnostics

Token accounting. For BAVT-family runs, `avg API tokens` includes prompt and completion tokens across planner, generator, and critic calls. `avg budget output tokens` is the quantity debited from the controller-side budget and reflects only the output-token budgeted component. All methods in the main table are evaluated under the same dual-budget constraint; the budget output token column reflects the output tokens charged against the budget in each case.

Method-specific audit notes. AFlow, Search-o1, BATS, BAVT, and VOI are all scored under the same hard dual-budget audit used in the main table: the same Search-R1 retrieval backend, question-only queries, `top_k=5`, and the same tool-call and output-token caps. Any example whose realized tool calls or output tokens exceed the target budget is counted as failed for that cell. For AFlow, this audit is implemented over replayed workflows: a replayed sample contributes to the scored cell only if its realized tool calls and answer output tokens remain within the target cap. Search-o1 and BATS are reported under the same audit semantics, and BAVT/VOI share the same BAVT-family search backbone under the same budget ladder.

Main-table comparator set. The primary empirical comparator set in this paper is AFlow, BATS, Search-o1, BAVT, and VOI. All five appear in the main table because all five are evaluated under the same hard tool/output-token audit.

Feasible-only usage audit. Table 8 reports realized usage on the feasible subset of each dataset-budget cell. For scoring, all main-table methods use the same hard tool/output-token audit. This table is a secondary feasibility-conditioned diagnostic: for each benchmark and budget level, we retain only examples satisfying both caps and then average realized tool calls and tokens over the retained subset. Different methods can therefore have different feasible subsets. For VOI, BATS, BAVT, and Search-o1, tokens are controller-debited output tokens. In these Qwen3-32B runs, AFlow feasibility is mainly tool-cap limited.

Table 8: **Feasible-only average resource usage by dataset and budget.** Each method cell reports average tool calls / average output tokens, i.e., `Avg. Tools / Avg. Tok`, computed only over examples that remain feasible under the corresponding tool/output-token cap. Budget labels follow the four-level budget ladder defined in Section 5.1. For scoring, all methods are subject to the same hard tool/output-token caps. The table is a feasible-only usage diagnostic rather than a scoring rule: token entries report the output tokens available from each audited execution. For AFlow, which is evaluated through replayed workflows, the reported token entry corresponds to answer-output tokens from the replay.

Benchmark	Budget	VOI (Ours)	BATS	BAVT	AFlow†	Search-o1
		`Avg. Tools / Avg. Tokens`				
Bamboogle	`low`	0.97 / 73.7	0.93 / 86.4	0.94 / 69.7	1.00 / 2.0	0.02 / 100.0
Bamboogle	`lower-mid`	1.61 / 138.8	1.00 / 190.9	1.43 / 135.4	1.96 / 7.8	0.69 / 193.9
Bamboogle	`upper-mid`	1.64 / 171.3	1.58 / 240.2	1.62 / 179.7	1.96 / 7.8	0.87 / 249.6
Bamboogle	`high`	1.91 / 195.9	1.91 / 268.5	1.89 / 211.3	2.19 / 7.0	1.06 / 350.5
HotpotQA	`low`	0.92 / 76.6	0.88 / 84.9	0.84 / 67.7	1.00 / 2.0	0.04 / 100.0
HotpotQA	`lower-mid`	1.49 / 134.2	1.00 / 189.6	1.56 / 138.6	1.00 / 2.0	0.63 / 193.7
HotpotQA	`upper-mid`	1.61 / 162.7	1.62 / 241.2	1.61 / 158.8	1.00 / 2.0	0.93 / 262.1
HotpotQA	`high`	1.91 / 197.7	1.91 / 269.0	1.88 / 218.9	2.98 / 8.9	1.09 / 361.5
MuSiQue	`low`	0.95 / 64.6	0.91 / 82.7	0.92 / 67.3	1.00 / 2.1	0.03 / 100.0
MuSiQue	`lower-mid`	1.60 / 141.1	1.00 / 193.8	1.46 / 137.3	1.48 / 4.6	0.68 / 192.5
MuSiQue	`upper-mid`	1.62 / 174.5	1.49 / 239.8	1.55 / 178.9	1.76 / 5.8	0.88 / 251.2
MuSiQue	`high`	1.85 / 191.5	1.95 / 278.7	1.87 / 226.0	2.21 / 7.2	1.09 / 364.4
2WikiMultihopQA	`low`	0.81 / 74.8	0.86 / 85.6	0.73 / 69.7	1.00 / 2.3	0.13 / 100.0
2WikiMultihopQA	`lower-mid`	1.73 / 140.7	1.17 / 189.8	1.64 / 132.9	1.90 / 6.8	0.50 / 193.6
2WikiMultihopQA	`upper-mid`	1.82 / 166.2	1.91 / 249.7	1.72 / 176.9	1.90 / 6.6	1.00 / 268.5
2WikiMultihopQA	`high`	2.04 / 193.9	2.09 / 271.4	1.91 / 222.2	2.14 / 6.1	1.22 / 376.4