# What Moves the Pareto Frontier in Tool-Using Agents? A Compute-Aware Study of ReAct Variants

**Rishi N. Simhadri**

## Abstract

Tool-using LLM agents are typically compared by accuracy alone, despite deployments being constrained by inference cost. We present a budgeted evaluation of common strategies for improving ReAct-style tool agents (multi-sample aggregation, iterative self-correction, and post-hoc answer revision) using Pareto analysis of cumulative accuracy versus token budget on three benchmarks (HotPotQA, FEVER, GSM8K) with Gemini 2.5 Flash. All experiments use three random seeds ($N$=500 per seed for HotPotQA/FEVER; $N$=1,015 for GSM8K); budgeted curves are computed post hoc from per-instance token logs. In our offline evaluation, Reflexion attains the highest accuracy on tool-heavy benchmarks (HotPotQA, FEVER), while CoT-SC leads on GSM8K. Reflexion's reported token costs are optimistic lower bounds because retries are stopped using ground-truth feedback, and its accuracy is similarly optimistic—a deployment without access to ground-truth labels would not achieve the same accuracy because the gold-label stopping criterion would be unavailable; both costs and accuracy would differ in practice. Sampling-based approaches often spend 3–5$\times$ more tokens for comparatively small gains on tool-heavy tasks. GSM8K, a pure-math benchmark with minimal tool interaction, shows substantially larger gains for CoT-SC, TCAR, and Reflexion—larger than on tool-heavy benchmarks, though less sharply separated than headline accuracy alone would suggest—consistent with repeated tool trajectories being an important contributor to the observed efficiency gap in our tool-heavy settings. We provide a compute-aware evaluation protocol (frontier analysis and marginal-cost metrics) and practical guidance for choosing agent designs under different budget regimes.

## 1 Introduction

Tool-using LLM agents are typically compared by accuracy alone, yet the most common improvement strategies (i.e. sampling more trajectories or iterating longer) directly increase inference cost. In practice, the decision is rarely "highest accuracy at any cost," but rather highest accuracy under a fixed budget. This paper argues that evaluating tool agents purely by aggregate accuracy can be misleading: it hides compute-inefficient methods and overstates those that are accurate but cost-prohibitive.

We ask: *Which agent improvement strategies actually pay off under realistic token budgets?* We compare five agent configurations spanning four improvement mechanism classes around a ReAct-style agent: single-pass tool use, multi-sample aggregation, iterative self-correction, and post-hoc revision. Our experiments span two tool-heavy benchmarks (HotPotQA, FEVER) and one reasoning-dominated contrast benchmark (GSM8K) using Gemini 2.5 Flash, with three random seeds per configuration. Across tool-heavy settings, iterative self-correction (Reflexion) achieves the highest accuracy but at large token cost; on GSM8K, CoT-SC edges ahead on accuracy while sampling-based strategies overall often consume 3–5$\times$ more tokens for modest gains. Including GSM8K, a pure mathematical reasoning task with minimal tool interaction, reveals that multi-trace and revision methods can yield larger absolute gains when reasoning (rather than tool interaction) is the primary bottleneck. To make these trade-offs actionable, we introduce a budgeted evaluation protocol based on Pareto analysis of cumulative accuracy versus token budget.

Our goal is not to claim that compute-aware evaluation itself is new. Prior work has already argued that accuracy should be considered jointly with inference budget (Snell et al., 2024; Wang et al., 2024; Wu et al., 2025), and recent studies have highlighted the importance of cost-aware agent evaluation (Kapoor et al., 2025; Liu et al., 2025; Erol et al., 2025). Instead, we ask how the budget-aware perspective changes in tool-using agent settings, where



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 923–940

extra compute frequently means repeating full action trajectories rather than merely extending a single reasoning trace. This motivates a controlled comparison of common ReAct-style improvement mechanisms under unified token accounting, with emphasis on deployment-relevant tradeoffs rather than accuracy alone.

*Tool-agent-specific compute-aware evaluation:* We adapt budget-aware evaluation to tool-using agents, where additional inference effort often reruns full thought–action–observation trajectories and tool interactions.

*Controlled mechanism-level comparison of ReAct variants:* We compare five mechanism-inspired ReAct-style improvement strategies across three benchmarks and three random seeds per configuration, reporting mean $\pm$ standard deviation under unified token accounting over full agent trajectories.

*Tool-heavy vs. reasoning-heavy contrast:* By including GSM8K (pure mathematical reasoning, minimal tool use) alongside HotPotQA and FEVER (tool-heavy), we find that multi-trace methods are substantially more cost-effective when reasoning, rather than tool interaction, is the primary bottleneck, consistent with repeated tool trajectories being an important contributor to the efficiency gap.

*Deployment-oriented analysis from run logs:* We provide a tool-agent-focused instantiation of budget-aware evaluation (post-hoc budget curves, Pareto comparisons, and marginal-cost views) to expose when sampling, iterative self-correction, and post-hoc revision are worthwhile under different budgets.

## 2 Related Work and Method Variants

**Tool-augmented reasoning agents.** ReAct (Yao et al., 2023b) introduced the "thought–action–observation" loop for tool-augmented agents, which forms the backbone of the methods we study. Self-consistency (Wang et al., 2023) improves reasoning by sampling multiple paths and aggregating via majority voting; when each sample involves tool calls, the cost multiplier is substantial. Tree of Thoughts (Yao et al., 2023a) generalizes chain-of-thought sampling into a search procedure over reasoning paths, further increasing the compute envelope. Reflexion (Shinn et al., 2023) adds a trial-and-error loop with critique-guided retries; related approaches include Self-Refine (Madaan et al., 2023) and RARR (Gao et al., 2023). All of these strategies improve accuracy by spending more inference-time compute, yet they are almost universally evaluated by accuracy alone.

**Agent benchmarks.** Agent evaluation suites such as AgentBench (Liu et al., 2024) and SWE-bench (Jimenez et al., 2024) have broadened the scope of agent evaluation to realistic, multi-step tasks including code repair, web navigation, and database querying. While these benchmarks have been instrumental in measuring agent *capability*, they predominantly report aggregate accuracy or pass rates and do not systematically account for the inference cost required to achieve those scores. Our work complements these benchmarks by demonstrating that the same agent strategies can look very different once token cost is measured alongside accuracy.

**Test-time compute scaling and budget-aware reasoning.** A growing line of work studies how additional inference-time compute translates into performance gains. Snell et al. (2024) show that optimally allocating test-time compute (e.g., via repeated sampling or verification) can outperform scaling model parameters, establishing diminishing-return curves for reasoning tasks. Wu et al. (2025) derive empirical inference scaling laws that characterize compute-optimal strategies for problem-solving. Wang et al. (2024) introduce the *token economy* framework, evaluating reasoning strategies (chain-of-thought, self-consistency, Tree of Thoughts, and others) under explicit token budgets and showing that strategy rankings can invert when cost is considered. These studies firmly establish that accuracy should be evaluated jointly with inference budget.

**Cost-aware evaluation and deployment efficiency.** Beyond reasoning, several efforts address inference cost more broadly. Chen et al. (2024) propose model cascading and caching strategies to reduce LLM serving cost while preserving quality. Erol et al. (2025) formalize *cost-of-pass*, the expected monetary cost per correct solution, and define frontier cost-of-pass as a Pareto-optimal metric for comparing models. Most directly related to our setting, Kapoor et al. (2025) argue that many state-of-the-art agents are needlessly complex and costly, presenting accuracy–cost Pareto frontiers for several agent benchmarks and advocating for joint accuracy–cost optimization. Liu et al. (2025) pro-

pose budget-aware tool-use scaling (BATS), which equips agents with explicit budget trackers combining token and tool-call costs to improve efficiency at deployment time.

**What prior cost-aware work does not cover.** The cost-aware studies above—whether focused on test-time compute scaling (Snell et al., 2024; Wang et al., 2024; Wu et al., 2025) or on deployment efficiency (Chen et al., 2024; Erol et al., 2025; Kapoor et al., 2025)—primarily evaluate *single-model reasoning*: extra compute extends a single chain of thought, samples additional reasoning traces, or selects among models in a cascade. Tool-using agents introduce a qualitatively different cost structure: extra compute often means re-running entire thought–action–observation trajectories, issuing repeated tool calls, and re-encoding growing observation histories. This trajectory-level cost amplification is not captured by reasoning-oriented scaling studies, and no prior work provides a controlled, mechanism-level comparison of standard ReAct improvement strategies under this cost structure. Our work fills this gap.

**Positioning of this work.** Our work is complementary rather than competing on the general claim that compute-aware evaluation matters. The key distinction is *where* and *how* the additional compute is spent. Prior budget-aware studies (Snell et al., 2024; Wang et al., 2024; Wu et al., 2025) focus on pure-reasoning settings where extra compute extends a single chain of thought or samples additional reasoning traces. In tool-using agent settings, however, additional inference effort often corresponds to re-running entire thought–action–observation trajectories, issuing more tool calls, and re-encoding longer observation histories, a qualitatively different cost structure. Kapoor et al. (2025) identify the importance of cost-aware agent evaluation but do not provide a controlled, mechanism-level comparison of specific improvement strategies under unified token accounting. Liu et al. (2025) focus on improving agents via budget trackers rather than on evaluation methodology.

Relative to all of this prior work, we provide a controlled mechanism-level comparison of common ReAct-style improvement strategies under unified token accounting, evidence that pure-reasoning compute-scaling insights transfer only partially to tool-using agents (where cost amplification arises from repeated tool interaction), and a tool-agent-focused instantiation of budget-aware evaluation that makes these tradeoffs actionable for practitioners.

Our variants are mechanism-inspired instantiations rather than canonical reproductions of prior methods. In particular, our CoT-SC uses LLM-based synthesis rather than strict majority vote, and HotPotQA majority voting uses an LLM semantic aggregation step. Additionally, our Reflexion uses ground-truth-based early termination (stopping once the answer is judged correct against the reference), which differs from the original self-evaluation stopping in Shinn et al. (2023); and TCAR is a single-pass post-hoc revision rather than the multi-round self-refine loop of Madaan et al. (2023). All results should therefore be read as comparisons of these specific instantiations, not as benchmarks of the original methods. We retain the established names for readability but note these differences where relevant.

**ReAct (baseline) (Yao et al., 2023b):** A single trajectory (composed of a thought, action, and observation loop) that proceeds linearly to a final answer. This serves as the standard performance baseline.

**CoT-SC (Self-Consistency, LLM synthesis variant) (Wang et al., 2023):** Runs the agent multiple times ($n = 3$) to generate independent reasoning trajectories. Unlike standard self-consistency which uses majority voting, our implementation employs an additional LLM call (same base model) for synthesis: a separate invocation reviews all generated trajectories and their conclusions to synthesize a final, consolidated answer. This increases compute by repeating the full trajectory generation process.

**Majority Voting (LLM-assisted semantic voting on HotPotQA):** We run $n = 3$ independent ReAct trajectories. The aggregation mechanism differs by dataset:

- **FEVER:** The final label is chosen by strict majority vote over {SUPPORTED, REFUTED, NOT ENOUGH INFO}; ties default to NOT ENOUGH INFO.
- **HotPotQA:** Because correct answers can be phrased differently, we use an LLM semantic aggregation step that clusters/normalizes the three candidate answers into equivalence groups and selects the dominant group (tie $\rightarrow$ null). This introduces one additional LLM

call, and we include its tokens in the total token cost reported for this method.

- **GSM8K:** Strict majority vote over extracted numerical answers; ties default to the first answer.

**Reflexion (Shinn et al., 2023):** An iterative self-correction approach using a trial-based loop (up to 7 trials). Each trial consists of a full ReAct trajectory followed by a critique step. After a trajectory completes, an additional verifier call (same base model, different prompt) critiques the reasoning trace for internal consistency and evidence usage. If the verifier deems the answer incorrect, it generates a "Plan" to guide the next trial. Subsequent trials are conditioned on these past plans, significantly increasing token usage due to the iterative nature. For offline evaluation, Reflexion uses ground-truth-based early termination: after each trial, we check correctness (FEVER exact-label match; HotPotQA judged against the reference answer; GSM8K numerical match) and stop additional trials once the answer is correct. This estimates the minimum offline compute needed to reach a correct answer under our capped retry scheme, but it is not deployment-realistic because ground truth is unavailable at inference time.

**Trajectory-Conditioned Answer Revision (TCAR):** A post-hoc revision step that operates on a single completed ReAct trajectory, related to prior work on iterative refinement (Madaan et al., 2023) and retrieval-augmented revision (Gao et al., 2023). An additional verifier call (same base model) analyzes the reasoning trace and the initial answer. If it detects an error in logic or evidence usage, it rewrites the final answer based solely on the existing evidence in the trace, without executing any new tool actions.

### 2.1 Evaluation Metrics

We report both task performance and inference-time compute, measured in tokens. Formal definitions of all metrics are in Appendix B.

**Accuracy:** The fraction of evaluated instances answered correctly.

**Token cost:** We measure total tokens consumed by a method across the evaluated set, including all text processed during inference (prompts, intermediate steps, tool observations, and completions) under a consistent accounting scheme. Token totals include all LLM calls made by the agent, including any synthesis or aggregation steps. From this we derive two normalized metrics:

**Non-token operational costs (appendix).** In addition to token metrics, we report run-log operational counters in Appendix D.6. These include LLM/API calls per task (`n_calls`), failed calls per task (`n_badcalls`), and framework-specific counters (`num_trials` for Reflexion; `num_traces_run` for CoT-SC/Majority Voting). Latency is excluded from core comparisons because per-instance timing is not consistently logged across all datasets.

**Tokens/Task:** Average tokens per evaluated instance.

**Tokens/Correct:** Average tokens per correct instance, capturing how much compute is spent on failures; methods with high Tokens/Correct are compute-inefficient even when their accuracy is slightly higher.

**Relative deltas:** In some tables we also report accuracy gain and additional tokens relative to the ReAct baseline. We report these relative deltas for all three benchmarks in Table 2.

### 2.2 Compute-Aware Evaluation

Aggregate accuracy alone does not capture whether a method is attractive under realistic compute constraints. We analyze methods using three compute-aware views (formal definitions in Appendix B).

**(1) Accuracy–compute tradeoff and Pareto efficiency (Miettinen, 1999):** We treat each method as a point in (Tokens/Task, Accuracy) space. A method is Pareto-dominated if another method achieves higher (or equal) accuracy with lower (or equal) cost, with at least one strict improvement. Methods that are not dominated form the Pareto frontier.

**(2) Offline budgeted throughput curves:** We construct cumulative accuracy vs. token budget curves from per-instance token usage. These curves answer: *If we can spend at most $B$ tokens total, how many instances can be processed (and with what accuracy)?* Curves are computed post hoc from realized costs using an offline "cheapest-first" ordering; we do not run agents under explicit budgets or perform early stopping. This post-hoc budget construction is distinct from Reflexion's ground-truth-based early termination during offline evaluation.

**(3) Marginal cost of additional correct instances:** To characterize how expensive it becomes to obtain additional correct predictions, we compute a marginal-cost curve over the instances each method correctly solves, sorted by realized token cost. This visualizes the cost distribution conditioned on correctness and highlights heavy-tail behavior.

**(4) Multi-seed evaluation:** LLM inference can be non-deterministic, so we repeat all experiments with three random seeds and report mean $\pm$ standard deviation. This helps distinguish genuine method differences from run-to-run variance, though we still avoid over-interpreting small accuracy differences and emphasize conclusions supported by large effect sizes and consistent dominance patterns.

## 3 Results

We present results on HotPotQA, FEVER, and GSM8K using both accuracy and token-based compute metrics. All results report mean $\pm$ standard deviation across three random seeds using Gemini 2.5 Flash (500 examples per seed for HotPotQA and FEVER; 1,015 examples per seed for GSM8K using a corrected stop-sequence rerun). Our goal is not to crown a single universal best method, but to characterize which improvement mechanism pays off under different compute budgets. Non-token operational counters are summarized in Appendix D.6.

HotPotQA and FEVER evaluate 500 instances per seed; GSM8K evaluates 1,015 instances per seed, drawn from the same deterministic slice of each dataset (HotPotQA dev distractor, FEVER paper dev, GSM8K test; see Appendix D.9 for details). We retain the larger GSM8K sample from the corrected-stop-sequence rerun rather than discard completed runs to match the 500-cap, and report the benchmark-specific sample sizes explicitly throughout. Agent trajectories are capped at 7 tool steps; if the cap is reached without a `Finish` action, a forced-finish default is applied (FEVER: NOT ENOUGH INFO; HotPotQA: null). Per-instance latency is not consistently logged, so token cost serves as our primary compute measure.

### 3.1 Overall comparison

Table 1 summarizes performance and cost across all three benchmarks. Three consistent patterns emerge: (1) Iterative self-correction (Reflexion) achieves the highest accuracy on the tool-heavy benchmarks, while CoT-SC leads on GSM8K; importantly, Reflexion's accuracy advantage depends on ground-truth-based early termination, so deployment accuracy—where ground truth is unavailable to stop retries—would be lower than reported. (2) Multi-sample aggregation (CoT-SC, Majority Voting) is often compute-inefficient in tool-agent settings: it increases tokens per task by a large factor while delivering comparatively modest gains over a single-pass agent. (3) GSM8K, where reasoning dominates and tool interaction is minimal, shows especially large gains for CoT-SC, TCAR, and Reflexion, suggesting that repeated tool trajectories are an important contributor to the observed efficiency gap on tool-heavy benchmarks.

Relative deltas vs. ReAct are reported in Appendix Table 2.

### 3.2 HotPotQA: accuracy gains vs. compute amplification

HotPotQA strongly penalizes methods that repeat full tool-using trajectories: additional attempts rerun search/lookup steps and re-encode observations, causing compute to scale rapidly. Reflexion improves accuracy by $+7.7$ pp relative to ReAct, but at $5.08\times$ the tokens per task, while multi-sample baselines (CoT-SC, Majority Voting) are even more expensive for comparatively smaller gains (Table 2).

**Looping vs. sampling:** Reflexion's gains plausibly come from plan changes across trials (recovering from earlier retrieval/tool-use mistakes), whereas sampling methods often repeat similar failures while multiplying trajectory cost.

**Why Tokens/Correct matters:** Tokens/Correct (Table 1) normalizes total run cost by the number of correct outputs, capturing the compute spent on failures; methods with high Tokens/Correct are compute-inefficient even when their accuracy is slightly higher.

**Practical takeaway:** On HotPotQA, repeated full trajectories are costly; unless budgets are high, single-pass or lightweight post-hoc methods are typically more attractive than multi-sample aggregation.

### 3.3 FEVER: strong gains from reflection, moderate gains elsewhere

On FEVER, iterative self-correction provides the largest accuracy improvement ($+16.2$ pp) relative to ReAct (Table 2), consistent with the idea that critique-and-retry helps verify evidence and avoid

| HotPotQA | | | |
|---|---|---|---|
| Method | Accuracy (%) | Tokens/Task | Tokens/Correct |
| ReAct | $38.3 \pm 2.2$ | $9{,}162 \pm 194$ | $23{,}974 \pm 1{,}904$ |
| Reflexion† | $46.0 \pm 1.4$ | $46{,}567 \pm 1{,}382$ | $101{,}310 \pm 6{,}149$ |
| TCAR | $39.2 \pm 2.3$ | $12{,}399 \pm 142$ | $31{,}716 \pm 2{,}237$ |
| Majority Voting | $44.4 \pm 3.8$ | $27{,}846 \pm 670$ | $63{,}098 \pm 7{,}280$ |
| CoT-SC | $42.8 \pm 4.2$ | $35{,}398 \pm 640$ | $83{,}453 \pm 10{,}131$ |
| **FEVER** | | | |
| Method | Accuracy (%) | Tokens/Task | Tokens/Correct |
| ReAct | $62.2 \pm 4.1$ | $3{,}508 \pm 258$ | $5{,}676 \pm 816$ |
| Reflexion† | $78.4 \pm 2.4$ | $18{,}715 \pm 3{,}399$ | $23{,}972 \pm 5{,}117$ |
| TCAR | $63.7 \pm 3.7$ | $5{,}996 \pm 247$ | $9{,}455 \pm 957$ |
| Majority Voting | $63.3 \pm 5.5$ | $10{,}752 \pm 650$ | $17{,}129 \pm 2{,}577$ |
| CoT-SC | $66.7 \pm 1.6$ | $15{,}037 \pm 638$ | $22{,}550 \pm 1{,}124$ |
| **GSM8K** | | | |
| Method | Accuracy (%) | Tokens/Task | Tokens/Correct |
| ReAct | $75.7 \pm 0.4$ | $3{,}720 \pm 10$ | $4{,}914 \pm 40$ |
| Reflexion† | $84.7 \pm 0.5$ | $5{,}676 \pm 77$ | $6{,}702 \pm 125$ |
| TCAR | $85.9 \pm 0.2$ | $4{,}752 \pm 15$ | $5{,}531 \pm 7$ |
| Majority Voting | $79.2 \pm 1.2$ | $11{,}434 \pm 61$ | $14{,}443 \pm 271$ |
| CoT-SC | $87.7 \pm 0.1$ | $13{,}945 \pm 54$ | $15{,}909 \pm 72$ |

Table 1: Accuracy and token cost (Gemini 2.5 Flash, mean $\pm$ std across 3 seeds; $N$=500 per seed for HotPotQA/FEVER and $N$=1,015 for GSM8K). † Reflexion uses ground-truth-based early termination during offline evaluation; reported token costs are optimistic lower bounds relative to deployment. See Appendix D for model and tooling details.

overconfident misclassification. In contrast, multi-sample aggregation yields comparatively small improvements while multiplying the cost of tool-using trajectories.

**Why reflection helps:** FEVER rewards careful evidence checking and calibration (including predicting NOT ENOUGH INFO), which aligns well with explicit critique and revision.

**Tokens/Correct perspective:** Tokens/Correct (Table 1) provides an end-to-end efficiency view; a method can be more expensive per task yet still favorable if it converts substantially more tasks into correct outputs.

**Practical takeaway:** If highest offline accuracy is the priority and one accepts optimistic lower-bound cost estimates for Reflexion, Reflexion is the strongest performer on FEVER in our setup; otherwise, ReAct remains the more compute-efficient baseline, with TCAR as a modest-cost middle option.

### 3.4 GSM8K: reasoning-dominated tasks amplify improvement gains

GSM8K presents a contrasting picture to the tool-heavy benchmarks. Here, several improvement methods yield substantially larger accuracy gains over ReAct (Table 2): CoT-SC achieves $+12.0$ pp, TCAR $+10.2$ pp, and Reflexion $+9.0$ pp. These gains are larger than on HotPotQA or FEVER for these methods, where most gains are single-digit; the spread is narrower than in the earlier (stop-sequence-uncorrected) run.

**Why gains are larger on GSM8K:** GSM8K is a pure mathematical reasoning task where errors stem from calculation mistakes or reasoning gaps, not from tool retrieval failures. When multiple traces independently reason about a math problem, they are more likely to produce meaningfully different reasoning paths, making aggregation effective. Similarly, post-hoc revision (TCAR) can catch arithmetic errors without needing new evidence.

**Compute multipliers are similar, but payoff is higher:** CoT-SC costs $3.75\times$ ReAct on GSM8K, comparable to its cost ratio on FEVER ($4.29\times$) and HotPotQA ($3.86\times$), but delivers $+12.0$ pp on GSM8K vs. $+4.5$ pp on FEVER. This asymmetry highlights that the *efficiency* of multi-trace methods depends critically on whether the bottleneck is reasoning (where resampling helps) or tool interaction (where resampling often reproduces similar failures).

**TCAR is the most compute-efficient improvement on GSM8K:** TCAR achieves $85.9\%$ accuracy at only $1.28\times$ ReAct cost. CoT-SC attains the highest accuracy ($87.7\%$) at $3.75\times$ ReAct cost. This suggests that for reasoning-dominated tasks, a single trajectory often contains sufficient information for a revision step to correct errors, while majority aggregation further reduces variance at

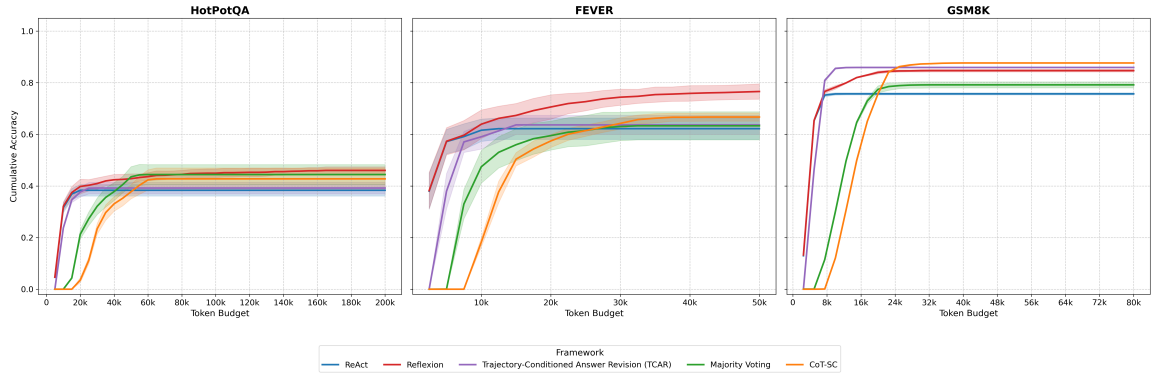


Figure 1: Cumulative accuracy vs. token budget for HotPotQA, FEVER, and GSM8K (mean across 3 seeds; shaded bands denote $\pm 1$ standard deviation). Curves are computed post hoc from per-instance agent token logs (no runtime budget enforcement), using an offline "cheapest-first" ordering by realized token cost. † Reflexion uses ground-truth-based early termination during offline evaluation; reported token costs are optimistic lower bounds relative to deployment. These curves are post-hoc descriptive summaries under an offline cheapest-first schedule, not online budget-allocation policies.

higher cost.

**Practical takeaway:** On reasoning-dominated tasks, multi-trace and revision methods are highly cost-effective. The pattern observed on HotPotQA and FEVER, consistent with a "tool trajectory penalty," is largely absent, making the standard compute-scaling intuitions from pure reasoning more applicable.

### 3.5 Offline budget-regime analysis

The Pareto curves (Figure 1) reveal distinct regimes where different strategies dominate:

**Low-budget regime** ($\approx$ ReAct-level budgets): Under our offline post-hoc ordering, single-pass agents are the strongest performers. Methods that require multiple independent trajectories (CoT-SC, Majority Voting) are not competitive because they multiply tool usage and token consumption.

**Medium-budget regime** (a few thousand extra tokens/task): Lightweight add-ons (e.g., post-hoc revision) may be attractive because they preserve tool behavior and increase cost modestly; however, any accuracy deltas in this band should be treated cautiously if they are small.

**High-budget regime** ($\gtrsim 5\times$ ReAct tokens/task): Reflexion achieves the highest accuracy on HotPotQA and FEVER in our offline evaluation, while CoT-SC reaches the highest GSM8K accuracy. Reflexion's token costs are favorably estimated because retries are terminated using ground-truth feedback, so its advantage should be interpreted as an optimistic upper-bound view of deployment utility.

Overall, the compute-aware lens makes a key point: under our offline post-hoc budget construction, methods that repeat tool-using trajectories can be dominated across wide budget ranges, whereas iterative self-correction offers clear accuracy benefits but only becomes a favorable choice when the budget allows.

### 3.6 Marginal cost of additional correct instances

Marginal-cost curves (Appendix Figure 2) show that across all three benchmarks, the cost of additional correct instances rises sharply in the tail: Reflexion exhibits the strongest heavy-tail behavior, while single-pass methods remain stable. Sampling-based methods incur elevated marginal costs throughout, consistent with the multiplicative overhead of repeating tool-using trajectories.

### 3.7 Reflexion failure modes and unique-solve behavior

A trajectory-level deep-dive (Appendix G) reveals that the most expensive Reflexion instances are almost always incorrect cases that exhaust the 7-trial cap, with token usage strongly coupled to trial count ($r = 0.939$) and negatively correlated with correctness. Despite this, Reflexion uniquely solves 40 FEVER and 11 HotPotQA instances that no other method resolves, suggesting that retry loops can help on a subset of hard examples even as the extreme-cost tail shows sharply diminishing returns.

## 4 Discussion

Our main takeaway is not that one method universally "wins," but that different improvement mechanisms occupy different regions of the accuracy–compute tradeoff, and their relative efficiency depends critically on whether the task bottleneck is tool interaction or reasoning.

### 4.1 Mechanism-level interpretation

In tool-using agents, each additional attempt triggers tool calls, longer trajectories, and observation re-encoding; compute therefore grows faster than in tool-free settings. This explains why multi-sample aggregation becomes costly quickly (it repeats entire tool-interaction trajectories), while approaches that reuse an existing trajectory (e.g., post-hoc revision) remain low-cost.

The GSM8K vs. tool-heavy contrast sharpens this point: CoT-SC achieves $+12.0$ pp over ReAct on GSM8K at $3.75\times$ cost, but only $+4.5$ pp on FEVER at a comparable multiplier. On reasoning tasks, different traces explore genuinely different paths, so aggregation reduces variance; on tool-heavy tasks, traces often retrieve similar evidence and reproduce similar errors, so resampling tends to multiply cost without proportional benefit in our experiments. Before investing in multi-trace methods, practitioners should assess whether the primary error mode is reasoning (where resampling helps) or tool interaction (where it often does not).

Qualitative inspection of trajectories further supports this picture. Failures on GSM8K are dominated by arithmetic and multi-step reasoning errors: the agent reaches a final answer but the calculation is wrong, so multiple independent traces that re-reason about the problem can correct these errors through aggregation or revision. By contrast, HotPotQA failures are dominated by tool-call loops—the agent issues repeated `Search` and `Lookup` actions without converging on the needed facts—and by hallucinated retrievals in which the agent fabricates intermediate facts not present in the actual observations. FEVER failures are disproportionately premature finishes: the agent concludes after insufficient evidence gathering, often defaulting to NOT ENOUGH INFO rather than continuing to search. These qualitative patterns, based on manual inspection of sampled trajectories rather than rigorous tagging, reinforce the paper's central claim that trajectory cost structure differs structurally between tool-heavy and reasoning-heavy tasks: in reasoning-heavy settings errors are self-contained in the reasoning trace and amenable to revision, while in tool-heavy settings errors arise from the retrieval process itself and are harder to correct without new tool interactions.

### 4.2 Reflection and post-hoc revision

Reflexion improves outcomes through error diagnosis, recovery behavior (altering plans across trials), and decision recalibration, mechanisms that differ qualitatively from simple resampling. However, these benefits come at predictable cost: additional LLM calls, longer prompts, and often additional tool interactions. Reflexion is best understood as a high-accuracy, high-cost option for high-budget regimes.

Post-hoc revision (TCAR) avoids additional tool calls entirely, making it inherently lower-cost. Its accuracy gains are task-dependent: modest on tool-heavy benchmarks ($+0.9$–$1.6$ pp) but substantial on GSM8K ($+10.2$ pp at only $1.28\times$ ReAct cost), suggesting it is most effective when the trajectory already contains sufficient information to diagnose errors.

### 4.3 Budget regimes and practical guidance

**Low budget (near single-pass cost):** Under our offline setup, single-pass ReAct is the strongest low-budget baseline; lightweight add-ons that avoid new tool calls may help, but small accuracy deltas should be treated cautiously. **High budget (accuracy prioritized):** Under our optimistic offline stopping protocol, Reflexion is strongest on the tool-heavy benchmarks, while CoT-SC reaches the highest GSM8K accuracy; Reflexion's deployment costs would exceed our reported estimates. **Sampling caution:** CoT-SC and majority voting were not compute-efficient in our two tool-heavy settings; whether this generalizes to other tool-use environments remains an open question. This is the main value of Pareto-style reporting: it converts "which method is best?" into "which method is best for my budget?"

### 4.4 Implications for reporting

Report compute alongside accuracy (tokens/task, tokens/correct), use budgeted curves to avoid misleading comparisons where one method's gains come from a 5–$10\times$ compute increase, and interpret small deltas conservatively unless supported by multiple seeds or paired statistical tests.

### 4.5 Limitations

Three limitations affect the strength of our conclusions: (1) three random seeds provide some variance estimates but are insufficient for formal significance testing; (2) three benchmarks may not cover all tool-use settings (e.g., cheaper or more deterministic tools); (3) absolute token values depend on prompt formatting, so we focus on relative differences. Additional methodological caveats (offline budget construction, ground-truth-based Reflexion early termination, and failure-mode deep-dive scope) are detailed in Appendix E.

Our study targets terminal-answer tool-use tasks (retrieval + reasoning + final decision). Interactive, long-horizon environments introduce a distinct axis: compute must be allocated across sequential actions and state transitions. Extending our compute–accuracy framework requires episode-level cost (tokens + tool calls + environment steps) and a step-wise compute controller; we leave this as future work.

All experiments use a single base model (Gemini 2.5 Flash); whether the reported Pareto patterns generalize across model families (e.g., open-source or differently-sized models) remains an open question for future work. Additionally, a deployment-realistic evaluation of Reflexion—using a self-contained stopping criterion that does not require ground-truth labels—is left to future work; such an evaluation would likely yield both lower accuracy and higher token costs than the offline figures we report.

## 5 Conclusion

Tool-using LLM agents are often compared primarily by aggregate accuracy, even though real deployments operate under strict inference budgets. In this work, we evaluated representative agent improvement strategies: multi-sample aggregation (CoT self-consistency, majority voting), iterative self-correction (Reflexion), and post-hoc revision, across three benchmarks (HotPotQA, FEVER, GSM8K) using Gemini 2.5 Flash, with three random seeds per configuration.

Across all settings, we observe consistent trade-offs: iterative self-correction yields the largest accuracy gains on tool-heavy tasks but at substantially higher token cost, while sampling-based aggregation can be compute-inefficient, consuming large additional tokens for modest improvements when tool interaction (rather than reasoning) is the primary bottleneck. GSM8K, included as a reasoning-dominated contrast benchmark, helps interpret when the tool-heavy pattern breaks: multi-trace and revision methods yield larger gains ($+12$ pp for CoT-SC, $+10$ pp for TCAR) when reasoning errors, rather than tool retrieval failures, drive incorrect answers. These results motivate compute-aware reporting as a default for tool-agent research: in addition to accuracy, future evaluations should include token costs and budgeted performance curves. More broadly, our results clarify *when* test-time compute scaling is cost-effective for agents: on reasoning-dominated tasks it can be highly efficient, but on tool-heavy tasks the cost of repeated tool trajectories often dominates.

## Acknowledgements

**Disclosure of AI Use.** AI assistants (Gemini, GPT models) were used for coding assistance during agent implementation, iterative refinement of agent prompts, and copy-editing of manuscript prose. All experimental design, analysis, and scientific claims are the authors' own.

## References


Lingjiao Chen, Matei Zaharia, and James Zou. 2024. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research (TMLR)*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Mehmet Hamza Erol, Batu El, Mirac Suzgun, Mert Yuksekgonul, and James Zou. 2025. Cost-of-pass: An economic framework for evaluating language models. *Preprint*, arXiv:2504.13359.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Yoon Kim, Shankar Natarajan, Nanyun Peng, Tao Yu, and 1 others. 2023. Rarr: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL) (Long Papers)*.

Google. 2025. Gemini 2.5 flash model documentation.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. 2025. AI agents that matter. *Transactions on Machine Learning Research (TMLR)*.

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. 2025. Budget-aware tool-use enables effective agent scaling. *Preprint*, arXiv:2511.17006.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations (ICLR)*.

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. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Kaisa Miettinen. 1999. *Nonlinear Multiobjective Optimization*. Springer.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024. Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 19916–19939.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. In *International Conference on Learning Representations (ICLR)*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

# A Relative Deltas vs. ReAct

# B Metric Definitions

This appendix provides formal definitions for the evaluation metrics summarized in Section 2.1 and Section 2.2.

## B.1 Basic Metrics

**Accuracy.** For each method, accuracy is computed as the fraction of evaluated instances answered correctly:

$$\text{Accuracy} = \frac{\#\text{Correct}}{\#\text{Evaluated}}.$$

**Tokens/Task.** Average tokens per evaluated instance:

$$\text{Tokens/Task} = \frac{\sum_i \text{tokens}_i}{N}.$$

**Tokens/Correct.** Average tokens per correct instance:

$$\text{Tokens/Correct} = \frac{\sum_i \text{tokens}_i}{\#\text{Correct}}.$$

## B.2 Compute-Aware Metrics

**Post-hoc budgeted performance curves.** Let $(t_i, y_i)$ denote the tokens consumed and correctness indicator for instance $i$ under a given method. We sort instances by realized cost, $t_{(1)} \le t_{(2)} \le \cdots$. For a total budget $B$, we take the largest prefix $m(B)$ such that $\sum_{j=1}^{m(B)} t_{(j)} \le B$, and report accuracy on this included subset:

$$\text{Acc}(B) = \frac{1}{m(B)} \sum_{j=1}^{m(B)} y_{(j)}.$$

This view can be interpreted as a cost-conditioned / throughput-oriented summary (an offline "cheapest-first" schedule), and it should not be read as the behavior of an online system that knows costs in advance.

**Marginal cost of additional correct instances.** Let $\mathcal{C}$ be the set of correct instances for a method, with per-instance token costs $\{t_i\}_{i \in \mathcal{C}}$. We sort these costs $t_{(1)} \le t_{(2)} \le \cdots \le t_{(K)}$ where $K = |\mathcal{C}|$. The marginal cost of the $j$-th correct instance is $t_{(j)}$. For readability, we report a binned/smoothed version: for bin size $w$ (we use $w = 10$), the plotted value at bin $b$ is

$$\bar{t}_b = \frac{1}{w} \sum_{j=(b-1)w+1}^{bw} t_{(j)}.$$

This curve visualizes the cost distribution conditioned on correctness and highlights heavy-tail behavior (e.g., when the last few correct instances are extremely expensive).

# C Marginal Cost Curves

Across all three benchmarks, the marginal cost rises sharply for the most expensive correct instances, revealing diminishing returns: the "last" correct items are disproportionately costly. Reflexion exhibits the strongest heavy-tail behavior, with marginal costs spiking dramatically in the final bins, whereas single-pass methods (ReAct, TCAR) remain comparatively stable. Sampling-based methods (CoT-SC, Majority Voting) incur elevated marginal costs throughout, consistent with the multiplicative overhead of repeating tool-using trajectories. On GSM8K, the marginal cost curves are notably steeper for high-accuracy methods (CoT-SC, TCAR), reflecting the increasing difficulty of solving the remaining mathematical reasoning problems.

# D Model and Tooling Details

## D.1 Base LLMs

- **Model:** gemini-2.5-flash (Google GenAI API). Used for all experiments reported in the paper.
- The same model is used for all agent LLM calls (trajectory generation, critique, synthesis).

## D.2 What uses the LLM (call taxonomy)

- ReAct trajectory generation (Thought/Action/Finish).
- Reflexion critique and revision / retry planning (when enabled).
- Multi-trace synthesis / aggregation (when enabled).

Each experiment configuration uses exactly one base LLM for all calls.

## D.3 Tools and environment

**HotPotQA and FEVER:**

- **Available actions:** `Search`, `Lookup`, `Finish`.
- **Backend:** Wikipedia search/lookup API via the evaluation environment.

| HotPotQA | | | | |
|---|---|---|---|---|
| Method | ΔAcc (pp) | Tokens/Task | ΔTokens | × ReAct |
| Reflexion† | +7.7 | 46,567 | +37,405 | 5.08× |
| TCAR | +0.9 | 12,399 | +3,237 | 1.35× |
| Majority Voting | +6.1 | 27,846 | +18,684 | 3.04× |
| CoT-SC | +4.4 | 35,398 | +26,236 | 3.86× |
| **FEVER** | | | | |
| Method | ΔAcc (pp) | Tokens/Task | ΔTokens | × ReAct |
| Reflexion† | +16.2 | 18,715 | +15,208 | 5.34× |
| TCAR | +1.5 | 5,996 | +2,488 | 1.71× |
| Majority Voting | +1.1 | 10,752 | +7,244 | 3.07× |
| CoT-SC | +4.5 | 15,037 | +11,529 | 4.29× |
| **GSM8K** | | | | |
| Method | ΔAcc (pp) | Tokens/Task | ΔTokens | × ReAct |
| Reflexion† | +9.0 | 5,676 | +1,957 | 1.53× |
| TCAR | +10.2 | 4,752 | +1,032 | 1.28× |
| Majority Voting | +3.5 | 11,434 | +7,714 | 3.07× |
| CoT-SC | +12.0 | 13,945 | +10,225 | 3.75× |

Table 2: Relative deltas vs. ReAct baseline (Gemini 2.5 Flash, mean across 3 seeds). † Reflexion uses ground-truth-based early termination during offline evaluation; reported token costs are optimistic lower bounds.

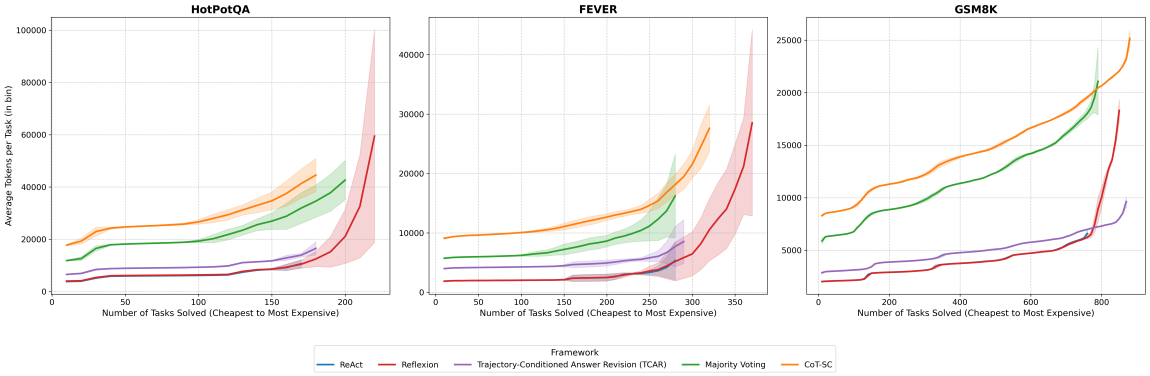


Figure 2: Marginal cost of additional correct instances for HotPotQA, FEVER, and GSM8K (mean across 3 seeds; shaded bands denote ±1 standard deviation). For each method, correct instances are sorted by realized token cost and averaged in bins of 10.

**GSM8K:**

- **Available actions:** `Calculate`, `Finish`.
- **Backend:** Python-based calculator for arithmetic operations. GSM8K is primarily a reasoning task with minimal tool interaction; most compute is spent on the reasoning trace itself.

Tool outputs are fed back into the LLM context for subsequent steps.

### D.4 Similarity / matching (non-pruning)

We do not use an embedding model or other learned semantic similarity model for pruning or matching. Any similarity/matching in wrappers or the environment is deterministic Python (e.g., string/keyword inclusion checks, token overlap / F1 via Python counters, and regex/string matching).

### D.5 Token logging and accounting

Token counts come from our experiment logger (using provider-reported usage metadata when available).

- **Included:** prompt and completion tokens.
- **All calls:** all LLM calls made by a variant (trajectory steps, reflection/critique, and synthesis/aggregation, depending on the method).

### D.6 Non-token Operational Cost Metrics

From seed-42 Gemini run logs, we report additional operational counters that are directly logged by the framework. We treat these as complementary diagnostics to token accounting rather than replacements for it.

Per-instance latency and explicit tool-call counts are not consistently logged across all datasets in

| HotPotQA (Gemini, seed 42) | | | |
|---|---|---|---|
| Method | Calls/task | Failed/task | Other |
| ReAct | 4.290 | 0.010 | — |
| Reflexion | 18.046* | 0.286* | trials=3.18* |
| TCAR | 5.256 | 0.006 | — |
| MajVote | 12.938 | 0.034 | traces=3.0 |
| CoT-SC | 12.938 | 0.034 | traces=3.0 |

Table 3: HotPotQA non-token operational metrics. Calls/task and Failed/task are means of `n_calls` and `n_badcalls`. * Reflexion fields present for 497/500 rows.

| FEVER (Gemini, seed 42) | | | |
|---|---|---|---|
| Method | Calls/task | Failed/task | Other |
| ReAct | 2.980 | 0.008 | — |
| Reflexion | 11.268 | 0.100 | trials=2.53 |
| TCAR | 4.034† | 0.006† | — |
| MajVote | 9.226 | 0.044 | traces=3.0 |
| CoT-SC | 9.226 | 0.044 | traces=3.0 |

Table 4: FEVER non-token operational metrics. Calls/task and Failed/task are means of `n_calls` and `n_badcalls`. † TCAR fields present for 500/501 rows.

these artifacts, so we do not include them in core method comparisons.

### D.7 Non-LLM artifact generation

Charts and plots are generated with matplotlib/seaborn, and tables are generated with pandas; these scripts do not call an LLM.

### D.8 Decoding settings

We keep decoding settings fixed across variants; see Appendix D.13 for details.

### D.9 Evaluation Slice and Dataset Handling

**Benchmarks and split files.** We evaluate on:

- HotPotQA dev distractor split: `data/hotpotqa/hotpot_dev_distractor_v1.json` (7,405 examples).
- FEVER paper dev split: `data/fever/paper_dev.jsonl` (9,999 claims).
- GSM8K test split: `data/gsm8k/test.jsonl` (1,319 examples).

**Evaluation slice construction.** For HotPotQA and FEVER, we use the same two-stage deterministic slice procedure:

1. Sample 3,000 task IDs from the full dev split using `random.sample` with seed `20260115`.
2. Shuffle those 3,000 IDs with `random.shuffle` (Fisher–Yates) using seed `42`.
3. Take the first 500 IDs as the evaluation slice (files: `results/*/dataset_sample_tasks/first_500_task_ids.json`).

For FEVER, claim IDs are mapped to line indices in `paper_dev.jsonl` before execution.

| GSM8K (Gemini, seed 42) | | | |
|---|---|---|---|
| Method | Calls/task | Failed/task | Other |
| ReAct | 4.738 | 0.161 | — |
| Reflexion | 7.975 | 0.492 | trials=1.40 |
| TCAR | 5.753 | 0.173 | — |
| MajVote | 14.209 | 0.608 | traces=3.0 |
| CoT-SC | 14.252 | 0.633 | traces=3.0 |

Table 5: GSM8K non-token operational metrics. Calls/task and Failed/task are means of `n_calls` and `n_badcalls`.

**Realized row counts used in aggregate tables.** Some seed–method combinations produced slightly more or fewer than 500 rows in their stored artifacts (e.g., HotPotQA seed-123 produced up to 579 rows; FEVER seed-456 up to 655 rows). For HotPotQA and FEVER, we cap each seed at the first 500 instances to maintain a uniform evaluation size. GSM8K results (Tables 1 and 2) use the full 1,015-instance rerun (`gemini_v2` logs) with a corrected stop-sequence, and are not capped.

| Method | HotPotQA $N$ | FEVER $N$ | GSM8K $N$ |
|---|---|---|---|
| ReAct | 500 | 500 | 1,015 |
| Reflexion | 497–500 | 500 | 1,015 |
| TCAR | 500 | 500 | 1,015 |
| MajVote | 499–500 | 500 | 1,015 |
| CoT-SC | 499–500 | 500 | 1,015 |

Table 6: Per-seed row counts after benchmark-specific capping. Ranges indicate variation across seeds.

**Answer normalization and FEVER label handling.** Exact-match scoring normalizes text by: (i) lowercasing, (ii) stripping punctuation, (iii) removing articles (`a/an/the`), and (iv) collapsing whitespace. FEVER answers are constrained to {`SUPPORTS`, `REFUTES`, `NOT ENOUGH INFO`}. FEVER majority-vote ties default to `NOT ENOUGH INFO`.

### D.10 Agent/Runtime Hyperparameters

- Max tool steps per ReAct trajectory: 7.
- Early stop: terminate immediately when a `Finish[...]` action is produced.

- Forced finish at step cap: `Finish[NOT ENOUGH INFO]` for FEVER, `Finish[null]` for HotPotQA.
- Parse-recovery behavior: if "Thought/Action" parsing fails, make a recovery action-only call; if still invalid, force the dataset fallback finish action above.
- CoT-SC: $n = 3$ traces (temperature 0.7) + 1 synthesis call.
- Majority Voting: $n = 3$ independent trajectories (temperature 0.7).
  - FEVER: strict majority vote over predicted labels; tie-break → `NOT ENOUGH INFO`.
  - HotPotQA: apply an LLM semantic aggregation step to group paraphrased answers and select the dominant group; tie-break → `null`. The aggregation step is counted in token usage.
- Reflexion: max trials = 7; one verifier/reflection call per trial; stop early when current answer is judged correct (FEVER: exact match to GT label, HotPotQA: LLM-judge semantic correctness vs GT), i.e., ground-truth-based early termination during offline evaluation.
- TCAR: one base ReAct trace + one post-hoc verifier/rewrite call; no new tool actions in the revision phase.
- Environment timeout retries (`env.step`): up to 10 retries on request timeouts.
- HotPotQA LLM call retries: request timeout 60s, up to 5 retries with exponential backoff (base 2s, capped at 32s, + jitter).

**Tool interface details.**

- `Search[entity]` resolves one Wikipedia page and returns the first 5 sentence-level snippets from that page.
- If no direct page is found, Search returns up to 5 similar titles.
- `Lookup[keyword]` returns one sentence per call from the cached page that contains the keyword (with `(Result i / N)` indexing).
- Observation formatting in context is literal trajectory append: `Thought i`, `Action i`, `Observation i`.
- Additional token/character truncation is not applied in agent code beyond the 5-sentence Search observation limit.

### D.11 Offline Budgeted Curve Construction (Cheapest-First)

We compute budget curves post hoc from per-instance pairs $(y_i, t_i)$, where $y_i \in \{0, 1\}$ is correctness and $t_i$ is total token cost.

```
Input: {(y_i, t_i)}_{i=1}^N, budgets {B_k}
Sort by cost: (y_(1),t_(1)),...,(y_(N),t_(N))

For each budget B:
  cum_tok = 0; m = 0; correct = 0
  For j = 1..N:
    If cum_tok + t_(j) > B: break
    cum_tok += t_(j); m += 1
    correct += y_(j)
  acc(B) = correct/m  (if m>0, else NA)
  cov(B) = m/N
  Report method, B, m, acc(B), cov(B)
```

This is an offline optimistic throughput schedule (it assumes per-instance costs are known after full runs), not an online routing policy. This offline cheapest-first budget construction is separate from Reflexion's ground-truth-based early termination during evaluation.

### D.12 Prompt Templates (Skeletons)

**ReAct base prompt skeleton.**

```
Instruction: Solve with
  Thought/Action/Observation interleaving.
Actions: Search[...], Lookup[...], Finish[...]

Claim or Question: ...
Thought 1: ...
Action 1: Search[...]
Observation 1: ...
...
Action k: Finish[answer]
```

Constraints: FEVER `Finish` output must be one of {`SUPPORTS, REFUTES, NOT ENOUGH INFO`}; HotPotQA uses free-form answers and `null` when unsupported.

**Reflexion critique prompt skeleton.**

```
Given an unsuccessful past trajectory:
- Identify strategy mistakes tied to
  specific actions.
- Produce a concise "Plan:" for next trial.
- Avoid generic summary; give actionable
  next-step searches/lookups.
```

**TCAR verifier/rewrite prompt skeleton.**

```
Given one completed trajectory and answer:
Verification: [CORRECT or INCORRECT]
Reasoning: ...
Final Answer: [original if correct,
               corrected if incorrect]
```

Constraint: corrected answer must use only evidence present in the trace observations.

**CoT-SC synthesis prompt skeleton.**

```
Input: claim/question + n trajectories (n=3)
Task: select best answer from evidence.
Output:
- FEVER: one of {SUPPORTS, REFUTES,
                 NOT ENOUGH INFO}
- HotPotQA: one final answer string only
```

### D.13 Decoding and API Settings

**Gemini 2.5 Flash (primary).**

- Model: `gemini-2.5-flash` (Google, 2025).
- Thinking budget: disabled (`thinking_budget=0`).
- `top_p = 1.0`.
- `max_output_tokens = 512`.
- Inter-call pacing: 0.5s sleep before FEVER LLM calls; 3.0s sleep before HotPotQA LLM calls.
- Temperature:
  - 0.0 for single-trace, verifier, and synthesis calls.
  - 0.7 for diversified multi-trace calls ($n = 3$: CoT-SC and Majority Voting; per-trace generation in those modes).
- Stop sequences by call type:
  - ReAct generation call: stop at `"\nObservation i:"`.
  - Recovery action-only call: stop at `"\n"`.
  - Verifier/reflection/judge/synthesis calls use either empty stop list or single-newline stop depending on helper function.
- Safety/MIME constraints: no additional custom safety settings and no response MIME-type constraints are set in agent code.

## E Additional Limitations and Caveats

**Offline budget construction.** Our budgeted curves sort examples by realized token cost ("cheapest-first"), which is a post-hoc analysis and can be seen as an optimistic throughput-oriented schedule rather than an online policy with unknown costs.

**Ground-truth-based early termination for Reflexion.** In offline evaluation, Reflexion stops retrying once the current answer is judged correct against the reference answer. This estimates the minimum compute needed for Reflexion to obtain a correct answer under our capped retry scheme, but it underestimates deployment cost because correctness is unknown at inference time. Accordingly, Reflexion's reported token costs and Pareto position should be interpreted as optimistic relative to a deployable stopping policy.

**Failure-mode deep-dive scope.** Our Reflexion failure-mode/trajectory analysis is post hoc and based on seed-42 Gemini logs. Tag-level interpretations (e.g., repetitive retry patterns) are useful diagnostics but remain heuristic; the multi-seed aggregate results confirm the high-level patterns but individual trajectory-level findings may vary across seeds.

## F Benchmark Details

**HotPotQA.** A multi-hop question answering benchmark where answering a question typically requires chaining information across multiple pieces of evidence (often across two or more distinct Wikipedia articles). Many questions are designed so that no single paragraph contains the full answer directly; instead, the agent must identify an intermediate entity or fact (via `Search`/`Lookup`), then follow that trail to recover the missing link needed to answer. Performance is measured by answer correctness (as defined in our evaluation), and the task naturally stresses (i) tool-use planning, (ii) iterative retrieval, and (iii) evidence integration across hops.

**FEVER.** A fact verification benchmark where the system is given a natural-language claim and must predict whether it is SUPPORTED, REFUTED, or NOT ENOUGH INFO based on retrievable evidence. Unlike standard QA, the correct behavior may require abstaining (NOT ENOUGH INFO) when the available evidence is insufficient. This benchmark stresses (i) targeted retrieval (finding the right page/sentences), (ii) careful evidence checking, and (iii) avoiding overconfident hallucinated justifications when evidence does not exist.

**GSM8K.** A grade-school math benchmark (Cobbe et al., 2021) consisting of 1,319 math word problems requiring multi-step arithmetic reasoning. Unlike HotPotQA and FEVER, GSM8K is

primarily a reasoning task where tool interaction (a calculator) is minimal; most compute is spent on the chain-of-thought reasoning trace itself rather than on tool calls. This makes GSM8K a useful contrast to the tool-heavy benchmarks: it allows us to isolate the effect of repeated *reasoning* (where resampling helps) from repeated *tool interaction* (where it often does not). Performance is measured by exact match on the final numerical answer.

## G Reflexion Deep-Dive Tables

This appendix reports the detailed trajectory-level analysis summarized in Section 3.6. All values come from the same logged runs used in the main paper (seed 42, gemini-2.5-flash, identical evaluation slices).

### G.1 Overall Reflexion cost/accuracy summary

| Dataset | $N$ | Acc. | Mean Tok. | Med. Tok. | Max Tok. |
|---|---|---|---|---|---|
| FEVER | 500 | 80.6% | 16,656 | 5,096 | 111,470 |
| HotPotQA | 500 | 44.2% | 47,747 | 20,696 | 197,386 |

Table 7: Overall Reflexion run statistics used in the deep-dive analysis.

### G.2 Most expensive Reflexion instances

| task_id | total tokens | correct | #trials | stop reason |
|---|---|---|---|---|
| 9202 | 111,470 | no | 7 | max_trials |
| 5922 | 96,612 | no | 7 | max_trials |
| 2168 | 95,048 | no | 7 | max_trials |
| 9581 | 100,669 | no | 7 | max_trials |
| 4242 | 78,066 | no | 7 | max_trials |
| 3026 | 76,562 | no | 5 | max_trials |
| 5518 | 54,674 | no | 6 | max_trials |
| 2939 | 54,614 | no | 7 | max_trials |
| 8028 | 51,122 | no | 4 | max_trials |
| 4425 | 44,632 | no | 7 | max_trials |

Table 8: FEVER: top expensive Reflexion instances in the deep-dive logs (all are incorrect and terminate at the trial cap).

### G.3 Token-decile behavior for Reflexion

### G.4 Cross-method solve overlap and unique solves

### G.5 Unique Reflexion wins and expensive misses

### G.6 Failure-mode tag distributions and subset contrast

### G.7 Deep-dive narrative findings

- Reflexion is the top-accuracy method in these runs (FEVER 80.6%, HotPotQA 44.2%) but with roughly 5× higher token cost than ReAct.
- Higher Reflexion token usage is associated with lower success probability in both datasets (negative token–correctness correlation).
- Many extreme-cost failures are trial-cap exhaustions with repetitive behavior, especially "stuck-same-answer" patterns.
- Reflexion still contributes meaningful unique solves (40 FEVER, 11 HotPotQA), showing that retry loops are useful on a non-trivial subset.
- On expensive subsets, baseline ReAct accuracy is very low (5.6% FEVER, 1.6% HotPotQA), supporting a hardness interpretation rather than purely wasted compute.

| task_id | total tokens | correct | #trials | stop reason |
|---|---|---|---|---|
| 672 | 197,386 | no | 7 | max_trials |
| 1927 | 174,610 | no | 7 | max_trials |
| 6081 | 173,638 | no | 7 | max_trials |
| 2517 | 167,548 | no | 7 | max_trials |
| 7244 | 165,823 | no | 7 | max_trials |
| 5187 | 163,575 | no | 7 | max_trials |
| 3693 | 163,544 | no | 7 | max_trials |
| 4057 | 160,955 | no | 7 | max_trials |
| 6569 | 158,756 | no | 7 | max_trials |
| 628 | 158,387 | no | 7 | max_trials |

Table 9: HotPotQA: top expensive Reflexion instances in the deep-dive logs (all are incorrect and terminate at the trial cap).

| Dec. | Acc. | Trials | Act./trial | Token range |
|---|---|---|---|---|
| 0 | 1.000 | 1.00 | 3.1 | 1,128–2,577 |
| 1 | 1.000 | 1.00 | 3.7 | 2,578–3,199 |
| 2 | 1.000 | 1.00 | 4.2 | 3,210–3,825 |
| 3 | 1.000 | 1.00 | 4.8 | 3,839–4,536 |
| 4 | 0.980 | 1.04 | 5.4 | 4,538–5,361 |
| 5 | 0.960 | 1.08 | 5.6 | 5,380–6,375 |
| 6 | 0.820 | 2.10 | 4.5 | 6,434–10,091 |
| 7 | 0.700 | 3.96 | 3.6 | 10,115–21,036 |
| 8 | 0.360 | 6.00 | 3.7 | 21,115–38,036 |
| 9 | 0.240 | 6.82 | 3.6 | 38,038–111,470 |

Table 10: FEVER Reflexion token-decile analysis. Low-cost deciles are near-perfect; the high-cost tail has low accuracy and many retries.

**Deep-dive caveats.** All deep-dive tables are from a single seed and post-hoc trajectory tags; they should be interpreted as diagnostic evidence, not definitive causal claims.

| Dec. | Acc. | Trials | Act./trial | Token range |
|---|---|---|---|---|
| 0 | 0.720 | 1.00 | 4.0 | 1,753–5,963 |
| 1 | 0.700 | 1.00 | 6.1 | 5,963–8,917 |
| 2 | 0.660 | 1.00 | 7.0 | 8,942–11,643 |
| 3 | 0.480 | 1.34 | 7.0 | 11,689–16,135 |
| 4 | 0.480 | 1.58 | 7.0 | 16,209–22,047 |
| 5 | 0.380 | 2.16 | 7.0 | 22,165–34,137 |
| 6 | 0.320 | 3.56 | 6.6 | 34,231–62,720 |
| 7 | 0.240 | 5.12 | 6.2 | 62,769–102,973 |
| 8 | 0.160 | 6.16 | 6.5 | 103,003–147,457 |
| 9 | 0.080 | 6.96 | 6.2 | 147,458–197,386 |

Table 11: HotPotQA Reflexion token-decile analysis. Accuracy declines monotonically with token decile.

| Dataset | $r$(tokens, trials) | $r$(tokens, correctness) |
|---|---|---|
| FEVER | 0.939 | -0.738 |
| HotPotQA | 0.939 | -0.493 |

Table 12: Correlation summary from Reflexion deep-dive logs.

| Method | FEVER solved | HotPotQA solved |
|---|---|---|
| ReAct | 317 | 179 |
| Reflexion | 403 | 221 |
| TCAR | 381 | 183 |
| Majority Voting | 379 | 200 |
| CoT-SC | 378 | 189 |

Table 13: Total solved instances by method (deep-dive logs).

| Method | FEVER unique | HotPotQA unique |
|---|---|---|
| ReAct | 3 | 1 |
| Reflexion | 40 | 11 |
| TCAR | 1 | 0 |
| Majority Voting | 6 | 12 |
| CoT-SC | 3 | 3 |

Table 14: Instances solved by exactly one method.

| | ReAct | Refl. | TCAR | MV | CoT-SC |
|---|---|---|---|---|---|
| ReAct | 179 | 174 | 171 | 155 | 155 |
| Refl. | 174 | 221 | 179 | 179 | 177 |
| TCAR | 171 | 179 | 183 | 153 | 154 |
| MV | 155 | 179 | 153 | 200 | 179 |
| CoT-SC | 155 | 177 | 154 | 179 | 189 |

Table 15: HotPotQA pairwise intersection counts of solved instances (Refl. = Reflexion, MV = Majority Voting).

| | ReAct | Refl. | TCAR | MV | CoT-SC |
|---|---|---|---|---|---|
| ReAct | 317 | 316 | 308 | 309 | 302 |
| Refl. | 316 | 403 | 363 | 365 | 360 |
| TCAR | 308 | 363 | 381 | 354 | 354 |
| MV | 309 | 365 | 354 | 379 | 362 |
| CoT-SC | 302 | 360 | 354 | 362 | 378 |

Table 16: FEVER pairwise intersection counts of solved instances (Refl. = Reflexion, MV = Majority Voting).

| task_id | tokens | #trials |
|---|---|---|
| 9581 | 100,669 | 7 |
| 3026 | 76,562 | 5 |
| 5518 | 54,674 | 6 |
| 8028 | 51,122 | 4 |
| 5536 | 50,643 | 5 |
| 6431 | 46,290 | 5 |
| 8144 | 45,552 | 5 |
| 4369 | 44,124 | 4 |
| 5031 | 41,668 | 7 |
| 234 | 39,593 | 4 |
| 9356 | 34,527 | 7 |
| 9250 | 33,906 | 5 |
| 7423 | 33,896 | 3 |
| 6116 | 30,201 | 3 |
| 5373 | 29,538 | 3 |
| 5629 | 28,716 | 4 |
| 2548 | 25,956 | 4 |
| 3020 | 24,309 | 3 |
| 446 | 23,346 | 4 |
| 619 | 22,961 | 3 |

Table 17: FEVER: top 20 Reflexion-unique solved instances by token cost.

| task_id | tokens | #trials |
|---|---|---|
| 2196 | 130,619 | 7 |
| 6460 | 129,749 | 7 |
| 2680 | 82,807 | 4 |
| 7184 | 74,042 | 4 |
| 3542 | 68,313 | 3 |
| 3007 | 54,090 | 4 |
| 3704 | 51,832 | 5 |
| 359 | 51,277 | 3 |
| 6130 | 37,781 | 2 |
| 5608 | 31,122 | 3 |
| 928 | 20,518 | 2 |

Table 18: HotPotQA: all Reflexion-unique solved instances (11 total).

| task_id | tokens | #trials |
|---|---|---|
| 9202 | 111,470 | 7 |
| 5922 | 96,612 | 7 |
| 2168 | 95,048 | 7 |
| 4242 | 78,066 | 7 |
| 2939 | 54,614 | 7 |
| 4425 | 44,632 | 7 |
| 1941 | 42,118 | 7 |
| 3603 | 32,464 | 7 |
| 2186 | 30,474 | 7 |
| 1413 | 30,357 | 7 |
| 9761 | 30,027 | 7 |
| 1952 | 29,429 | 7 |
| 4062 | 29,126 | 7 |

Table 19: FEVER: expensive Reflexion failures that are solved by at least one other method (top by token cost).

| task_id | tokens | #trials |
|---|---|---|
| 672 | 197,386 | 7 |
| 1927 | 174,610 | 7 |
| 6081 | 173,638 | 7 |
| 2517 | 167,548 | 7 |
| 7244 | 165,823 | 7 |
| 5187 | 163,575 | 7 |
| 3693 | 163,544 | 7 |
| 4057 | 160,955 | 7 |
| 6569 | 158,756 | 7 |
| 628 | 158,387 | 7 |
| 1547 | 156,996 | 7 |
| 930 | 125,122 | 7 |
| 6797 | 123,689 | 7 |
| 6796 | 111,991 | 7 |
| 2734 | 86,532 | 7 |
| 1258 | 76,065 | 7 |
| 6485 | 72,425 | 7 |
| 3055 | 61,242 | 7 |
| 4873 | 31,029 | 2 |
| 7333 | 15,681 | 2 |

Table 20: HotPotQA: expensive Reflexion failures that are solved by at least one other method (top by token cost).

| Tag | Exp.-correct (28) | Exp.-incorrect (97) |
|---|---|---|
| exhausted_trials | 5 (18%) | 97 (100%) |
| stuck_same_answer | 0 (0%) | 84 (87%) |
| search_loop | 7 (25%) | 41 (42%) |
| low_div_reflection | 4 (14%) | 36 (37%) |
| forced_finish | 14 (50%) | 30 (31%) |
| expensive | 9 (32%) | 28 (29%) |
| repet._reflection | 0 (0%) | 21 (22%) |
| very_expensive | 1 (4%) | 10 (10%) |
| mostly_forced | 4 (14%) | 7 (7%) |
| clean_success | 9 (32%) | 0 (0%) |

Table 21: FEVER Reflexion failure-mode tags for expensive cases ($\geq$25,502 tokens).

| Tag | Exp.-correct (8) | Exp.-incorrect (117) |
|---|---|---|
| search_loop | 8 (100%) | 115 (98%) |
| forced_finish | 8 (100%) | 88 (75%) |
| very_expensive | 7 (88%) | 92 (79%) |
| exhausted_trials | 3 (38%) | 113 (97%) |
| stuck_same_answer | 0 (0%) | 78 (67%) |
| mostly_forced | 4 (50%) | 52 (44%) |
| low_ans_diversity | 0 (0%) | 32 (27%) |
| expensive | 1 (12%) | 25 (21%) |

Table 22: HotPotQA Reflexion failure-mode tags for expensive cases ($\geq$82,950 tokens).

| Dataset | ReAct acc. | Dominant ReAct tags |
|---|---|---|
| FEVER (n=125) | 5.6% | other_failure (78%), max_actions (18%), forced_finish (17%) |
| HotPotQA (n=125) | 1.6% | max_actions (57%), forced_finish (42%), other_failure (42%) |

Table 23: ReAct performance on the same expensive-Reflexion subsets, indicating these are genuinely hard instances.